# Oracle® Enterprise Single Sign-On Logon Manager

Configuring and Diagnosing
Logon Manager Application Templates

Release 11.1.2

**E27165-03**

March 2013

Oracle Enterprise Single Sign-On Logon Manager: Configuring and Diagnosing Logon Manager Application Templates

Release 11.1.2.1.0

E27165-03

# Table of Contents

# Preface

## Audience

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

Readers of this guide should be familiar with deploying and configuring the Logon Manager Agent and using the Logon Manager Administrative Console. Detailed information about each function described in this guide is available in the Logon Manager Administrative Console help.

## Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support.

For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

## Related Documents

We continually strive to keep our documentation accurate and up to date. For the latest version of this and other documents, visit http://docs.oracle.com/cd/E29306_01/index.htm. For more information, see the other documents in the documentation set for this release:

- **Oracle Enterprise Single Sign-On Suite Plus**
  - *Release Notes*
  - *Installation Guide*
  - *Administrator's Guide*
  - *Secure Deployment Guide*
  - *User's Guide*
- **Oracle Enterprise Single Sign-On Logon Manager**
  - *Deploying Logon Manager with a Directory-Based Repository*
- **Oracle Enterprise Single Sign-On Provisioning Gateway**
  - *Administrator's Guide*
  - *Command Line Interface Guide*
  - *Oracle Identity Manager Connector Guide*
  - *Sun Java System Identity Manager Connector Guide*
  - *IBM Tivoli Identity Manager Connector Guide*
- **Oracle Enterprise Single Sign-On Universal Authentication Manager**
- *Administrator's Guide*
- *User's Guide*

## Conventions

The following text conventions are used in this document:

| Term or Abbreviation | Description |
|---|---|
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

# **Part 1:** Windows Applications

This chapter explains the concepts necessary to understand how and why you should configure Windows application templates to solve specific sign-on scenarios, as well as the recommended best-practice procedures for configuring and testing Windows application forms, and diagnosis and resolution steps for the most common issues that may cause the Agent to erratically detect and/or respond to application forms. It covers the following topics:

- Overview of a Sign-On Event
- Understanding Window Detection
- Understanding Form Response
- Determining the Optimal Configuration for a Form
- Configuring a Form
- Testing the Configuration of a Form
- Publishing a Template to the Repository
- Troubleshooting Window Detection
- Troubleshooting Form Response When Using Control IDs
- Troubleshooting Form Response When Using “SendKeys”
- Troubleshooting Matching
- Troubleshooting a Logon Loop
- Troubleshooting Java Application Issues

> **Tip:** If the troubleshooting steps in this chapter do not resolve your issue, you can troubleshoot further by tracing and logging the activity of Logon Manager and submitting the logged information to Oracle Support for analysis. For this purpose, Oracle provides the Trace Controller utility, from Oracle Support. For information on how to use the utility, see the “Using the Trace Controller Utility” topic in the *Enterprise Single Sign-On Suite Plus Administrator’s Guide*.

## Overview of a Sign-On Event

Logon Manager 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 application windows, the Agent constantly monitors the currently logged in user's Windows message queue and responds as follows:

1. **Agent detects the window.** For each new window it detects, the Agent does the following:
    a. Searches through all of the available Windows application templates to find a match for the values of the identification criteria defined in the template. (These include the window title, window class, and executable name.)
    b. Selects the first template whose configuration matches the criteria presented by the window.
    c. Begins the response process.
2. **Agent responds to the form within the detected window.** 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. (Optional) Performs the actions, such as setting field focus, which might be required by the application to invoke or activate the logon or password change form.
    b. Retrieves the associated credentials from the user's store (if they exist) and populates the appropriate fields. (If the credentials don't exist, the Agent prompts the user to store them.)
    c. Performs the actions necessary to submit the credentials to the application for processing, such as pressing the **Logon** button.
    d. (Optional) Detects any follow-up forms or dialogs, such as new password confirmation, and performs the required action.

## Understanding Window Detection

Whenever the Agent detects the instantiation of a new window (through the Windows message queue), the Agent examines the values of the following characteristics of the window and compares them to the values stored in each available *application template* in order to uniquely identify the window and the form it contains:

- Window title
- Window class
- Executable name (also referred to as a *module name* or an *AppPath key*)

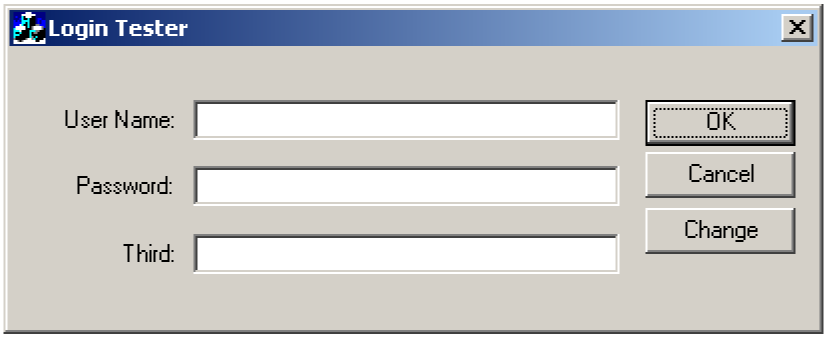
**Note:** An *application template* is a set of configuration options that instruct the Logon Manager Agent how to detect and respond to application windows and the forms they contain.

If a window other than the target window shares the values of these identifying criteria with the target window, the Agent may erroneously respond to such a window in addition to the target window. In such cases, you must use matching to more precisely constrain the allowable values of these criteria so that the resulting configuration uniquely identifies the target window and form to the Agent. For more information on matching, see Using Matching to Improve Detection Accuracy.

If a match is found, the Agent responds to the form as defined in the first matched template, i.e., logs the user on or initiates password change.

## Understanding Form Response

Once the Agent recognizes an application window, it checks for the presence of fields and controls that comprise a "form," such as a logon form, or a password change form.  A logon form, for example, typically contains at least a user name field, a password field, and a button that submits the credentials to the application. An example logon form is shown below.



## Supported Form Types

As of the release date of this document, Logon Manager supports the following types of forms in Windows applications:

- Logon
- Logon success (a message confirming successful logon)
- Logon failure (a message indicating the injected credentials have been rejected)
- Password change
- Password confirmation (a dialog requesting confirmation of the new password)
- Password change success (a dialog confirming successful password change)
- Password change failure (a dialog indicating the new password was rejected)

The same application window may contain different forms depending on the invoked function (i.e., logon or password change); thus, a single template can contain definitions for the multiple forms that the window can display. For most applications, you need to only define the forms to which you want Logon Manager 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.

Logon Manager 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.

## Supported Form Response Methods

Depending on the design of the target application, Logon Manager can use one of the following methods to interact with the fields and controls in the target form.

### Control IDs

This is the default and preferred form interaction method for most Windows applications. This method uses the Windows API to identify and interact with the fields and controls within the form. In a Windows API-compliant application, each field and control exposes a unique Control ID to the operating system. The Agent detects these Control IDs and binds to them specific sign-on functions that signify the purpose of the object represented by the Control ID, such as the password field or the “submit” button.

> **Note:** If some or all of the Control IDs exposed by the application are non-unique or dynamic, the Agent can substitute ordinals – sequential ID numbers assigned to each item in the window from top to bottom, left to right – to uniquely identify the detected fields and controls.

If a field or control does not expose its Control ID, or if there are additional actions required to complete the sign-on event, such as selecting a check box or manually setting focus on a specific field, you may also have to use the “SendKeys” method, in tandem with Control IDs, to interact with the target form.

### “SendKeys”

This method allows the Agent to interact with the target application by emulating user input, such as keystrokes and mouse clicks. Use this method if the Agent is unable to programmatically detect or interact with the target fields and controls. For example, if an application does not expose Control IDs for any of its fields and controls, you will need to send individual keystrokes to populate the fields, mouse clicks or **Tab** keystrokes to toggle between fields, and a final mouse click to engage the **Logon** button.

> **Note:** If you configure a form definition to use the “Control IDs” method and Logon Manager fails to inject credentials programmatically due to application incompatibility, by default Logon Manager will automatically fall back to the “SendKeys” method and retry injection, which could result in credentials being injected into the wrong window or application. You can disable this fallback behavior by following the steps in Disabling Automatic “SendKeys” Fallback.

### Control IDs with “SendKeys”

This method is a “best-of-both-worlds” combination of the two above methods and is the preferred way of solving sign-on scenarios that require actions that cannot be performed programmatically. Control IDs are used to interact with the form wherever possible, while keystrokes and mouse clicks are sent to accomplish tasks such as setting field focus, selecting a check box, and other actions that the Agent cannot perform programmatically within the target application.

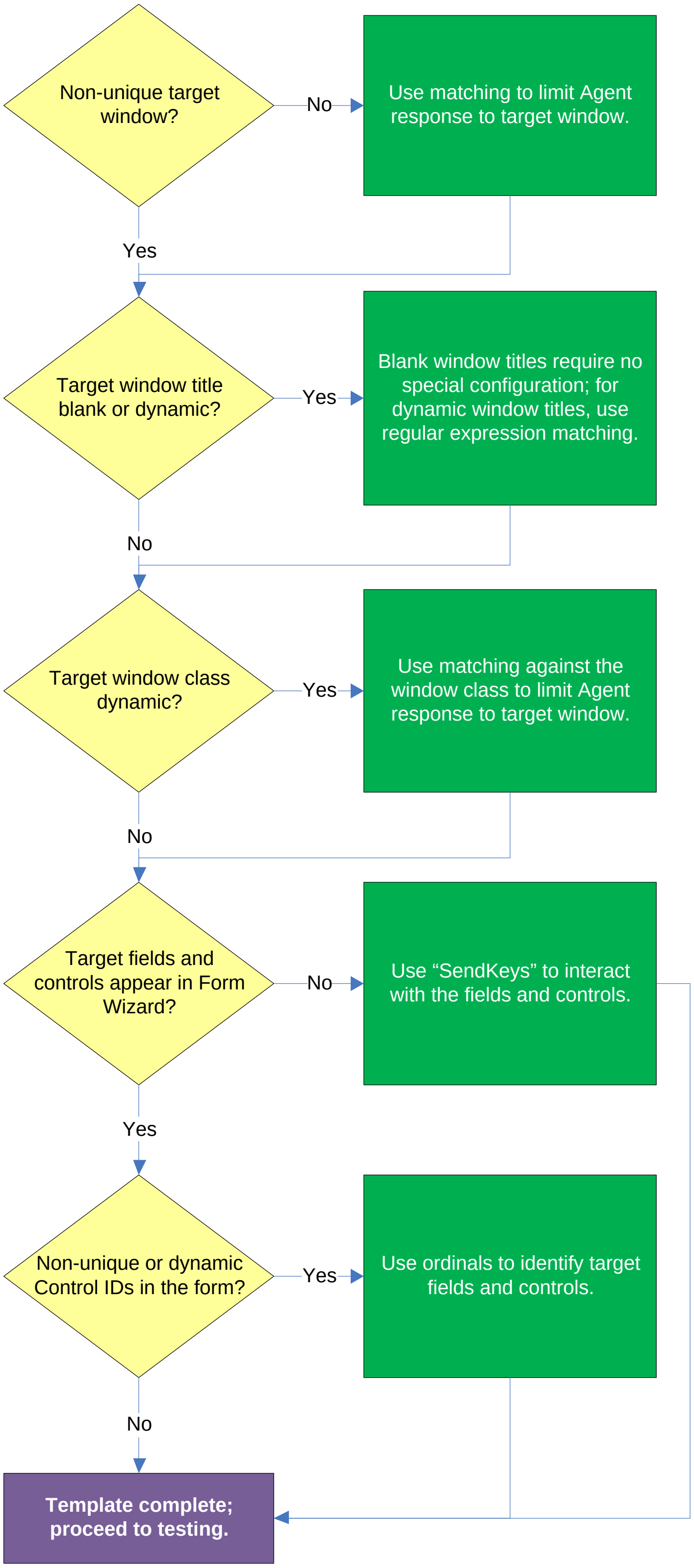
> **Note:** To achieve this “mixed” mode, configure the form to initially use the “SendKeys” response method, then configure the desired “SendKeys” actions; while configuring the actions, enable the **Inject directly into control** option for actions that you wish to retain the “Control IDs” programmatic response method.

For example, if fields must be populated in a specific order due to cascading validation (i.e., the password field becomes active only once the user name field has been populated), you would use Control IDs to inject credentials into the fields, but send a **Tab** keystroke via “SendKeys” between each field injection to advance field focus.

# 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


Non-unique target window?
No
Use matching to limit Agent response to target window.
Yes
Target window title blank or dynamic?
Yes
Blank window titles require no special configuration; for dynamic window titles, use regular expression matching.
No
Target window class dynamic?
Yes
Use matching against the window class to limit Agent response to target window.
No
Target fields and controls appear in Form Wizard?
No
Use "SendKeys" to interact with the fields and controls.
Yes
Non-unique or dynamic Control IDs in the form?
Yes
Use ordinals to identify target fields and controls.
No
Template complete; proceed to testing.

### Non-unique target window?

If the Agent cannot uniquely distinguish the target window from others, (i.e., another window whose window title, class, and/or module name are identical to those of the target window), the Agent might erroneously respond to such a window *in addition* to the target window. For more information on the causes of this issue and steps for resolving it, see Troubleshooting Form Detection.

### Target window title blank or dynamic?

Blank window titles require no special configuration; dynamic text in the window title can be matched using either regular expressions or environment variables. For more information, see Using Matching to Improve Detection Accuracy.

### Target window class dynamic?

If the window class of the target window is partially or fully dynamic, you must use matching to allow the Agent to uniquely identify the target window by its title. For more information, see Using Matching to Improve Detection Accuracy.

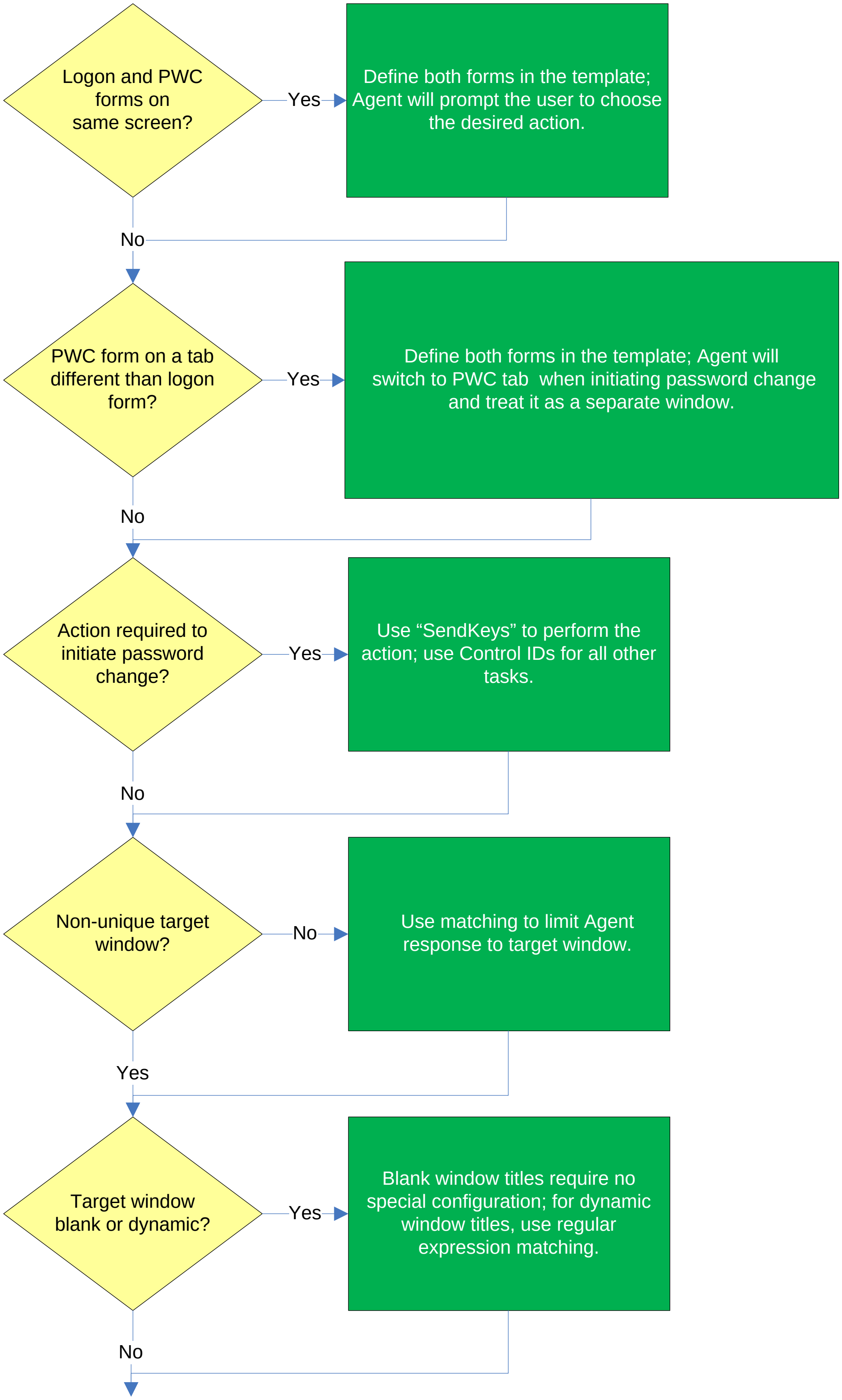
### Target fields and controls appear in the Form Wizard?

In most situations, the Form Wizard successfully detects and displays the credential fields and controls present in the target window. If no fields or controls are visible in the Form Wizard, or if the items listed do not correspond to any of the fields or controls in the target window, you may need to use "SendKeys" to interact with that field or control. For more information, see Supported Form Response Methods.

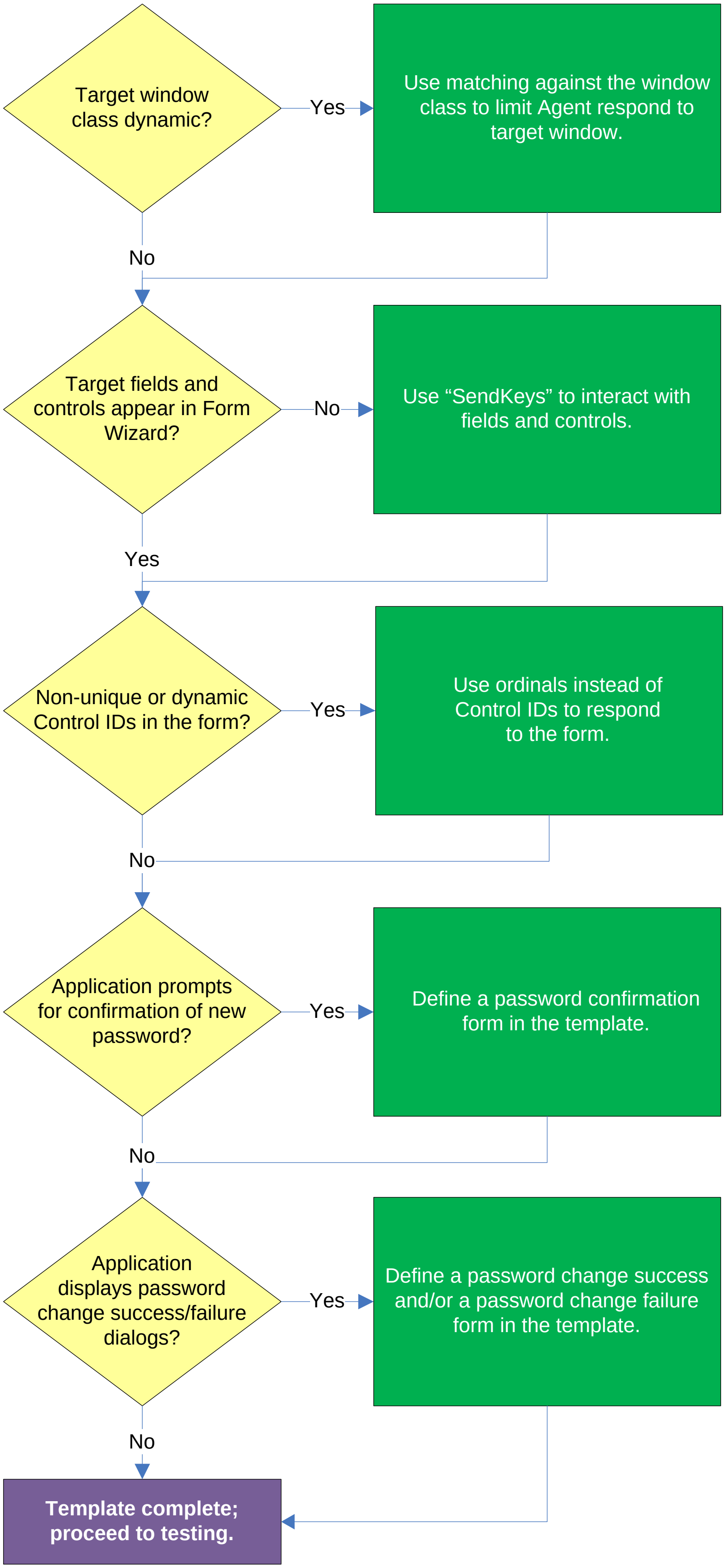
### Non-unique or dynamic Control IDs in the form?

Once you have identified the credential fields and controls that you wish to enable for sign-on, check that their Control IDs are unique. If you discover duplicates, Logon Manager's response to the application might be unreliable; in such cases, use ordinals to interact with the fields and controls in question. For more information, see Supported Form Response Methods.

## Determining the Optimal Configuration for a Password Change Form

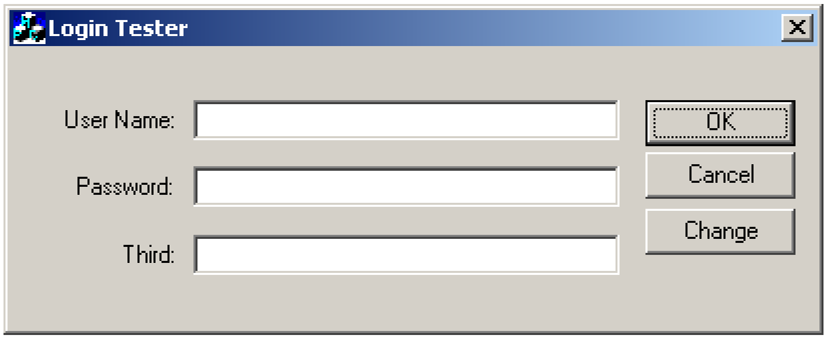


Logon and PWC forms on same screen?
Yes
Define both forms in the template; Agent will prompt the user to choose the desired action.
No
PWC form on a tab different than logon form?
Yes
Define both forms in the template; Agent will switch to PWC tab  when initiating password change and treat it as a separate window.
No
Action required to initiate password change?
Yes
Use “SendKeys” to perform the action; use Control IDs for all other tasks.
No
Non-unique target window?
No
Use matching to limit Agent response to target window.
Yes
Target window blank or dynamic?
Yes
Blank window titles require no special configuration; for dynamic window titles, use regular expression matching.
No

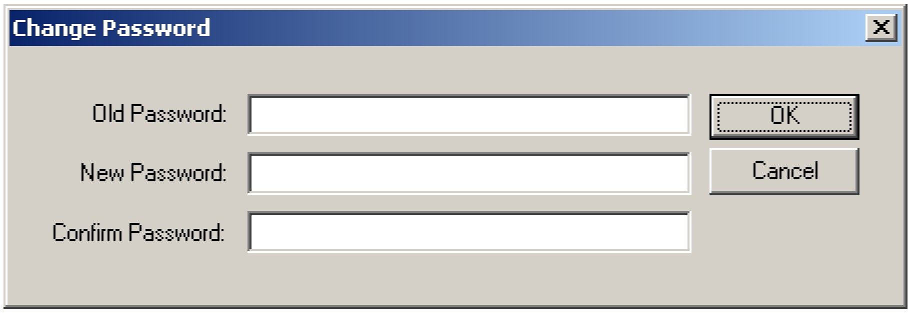

Target window class dynamic?
Yes
Use matching against the window class to limit Agent respond to target window.
No
Target fields and controls appear in Form Wizard?
No
Use “SendKeys” to interact with fields and controls.
Yes
Non-unique or dynamic Control IDs in the form?
Yes
Use ordinals instead of Control IDs to respond to the form.
No
Application prompts for confirmation of new password?
Yes
Define a password confirmation form in the template.
No
Application displays password change success/failure dialogs?
Yes
Define a password change success and/or a password change failure form in the template.
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. For example, the logon screen shown below contains a **Change** button.



This **Change** button, in turn, invokes the password change screen.



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.

### Password change form on a tab different than logon form?

Define the logon and password change forms in the template. The Agent will automatically switch to the password change tab when password change is initiated, and treat the password change form tab as a separate window.

**Note:** Only applications that implement tabs in full compliance with the WinAPI specification are supported.

### Action required to initiate password change?

If the form requires an action to initiate password change, such as selecting a checkbox, and the Agent is unable to perform that action programmatically, you may need to use "SendKeys" to complete the action by sending keystrokes and/or mouse clicks to the application. For more information, see Supported Form Response Methods.

### Non-unique target window?

If the Agent cannot uniquely distinguish the target window from other, similar windows, (i.e., when another window is instantiated whose window title, class, and module name are identical to those of the target window), the Agent might erroneously respond to such a window *in addition* to the target window. For more information on the causes of this issue and steps for resolving it, see Troubleshooting Form Detection.

### Target window title blank or dynamic?

Blank window titles require no special configuration; dynamic text in the window title can be matched using either regular expressions or environment variables. For more information, see Using Matching to Improve Detection Accuracy.

### Target window class dynamic?

If the window class of the target window is partially or fully dynamic, you must use matching to allow the Agent to uniquely identify the target window by its title. For more information, see Using Matching to Improve Detection Accuracy.

### Target fields and controls appear in the Form Wizard?

In most situations, the Form Wizard successfully detects and displays the credential fields and controls present in the target window. If no fields or controls are visible in the Form Wizard, or if the items listed do not correspond to any of the fields or controls in the target window, you may need to use "SendKeys" to interact with that field or control. For more information, see Supported Form Response Methods.

### Non-unique or dynamic Control IDs in the form?

Once you have identified the credential fields and controls that you wish to enable for sign-on, check that their Control IDs are unique. If you discover duplicates, Logon Manager's response to the application might be unreliable; in such cases, use ordinals to interact with the fields and controls in question. For more information, see Supported Form Response Methods.

### 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 dialog, configure the password change form, then configure a "Confirm Password" form that will respond to the confirmation dialog. For instructions, see Configuring a 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. For instructions, see Configuring a Form.

## Configuring a Form

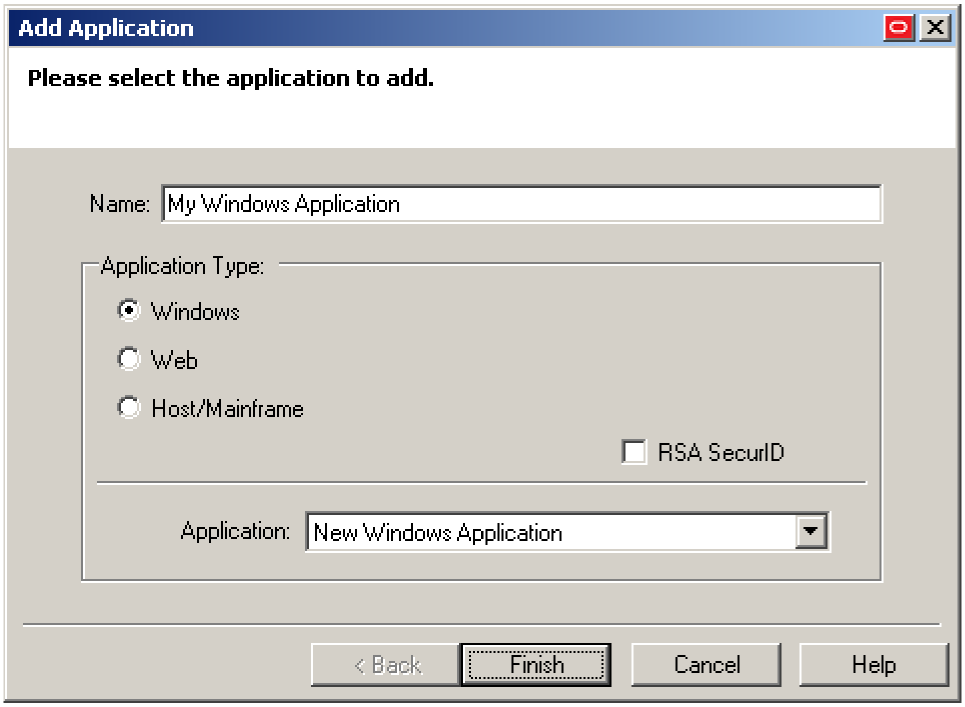
The procedures in this section use concepts and terminology explained earlier in this guide. When performing the procedures in this section, refer to Determining the Optimal Configuration for a Form to make configuration decisions that best suit the target application.

### Configuring a Form Using Control IDs

To configure a Windows application form that will use the Control IDs response method, do the following:

1. Open the Logon Manager Administrative Console. By default, the shortcut is located at **Start → Programs → Oracle → ESSO-LM Administrative 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 the logon 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.



Add Application

Please select the application to add.

Name: My Windows Application

Application Type:

● Windows

○ Web

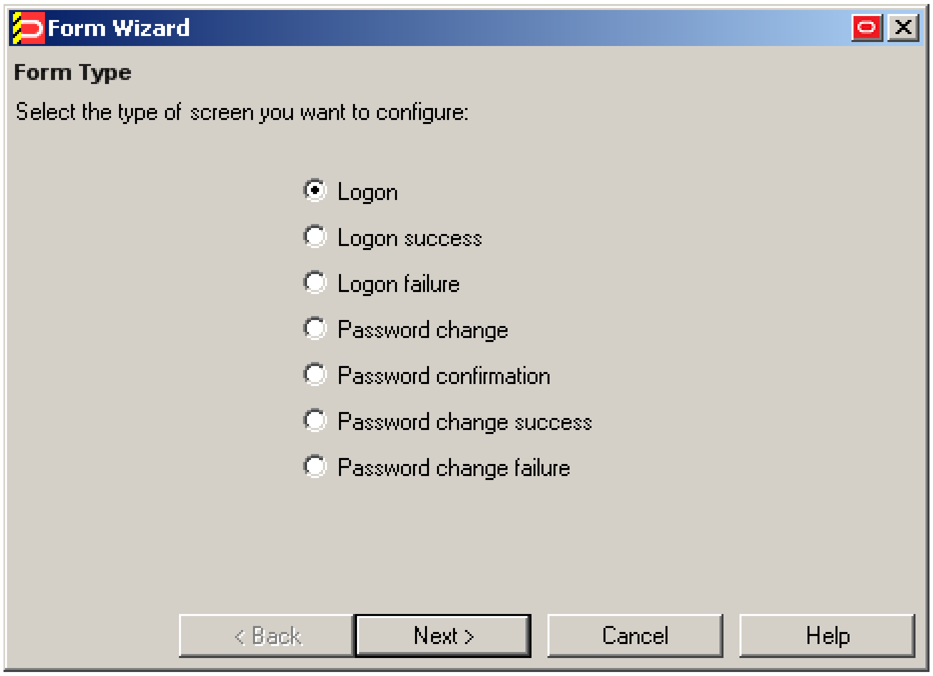
○ Host/Mainframe

□ RSA SecurID

Application: New Windows Application

- If you want to add a logon 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 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**.



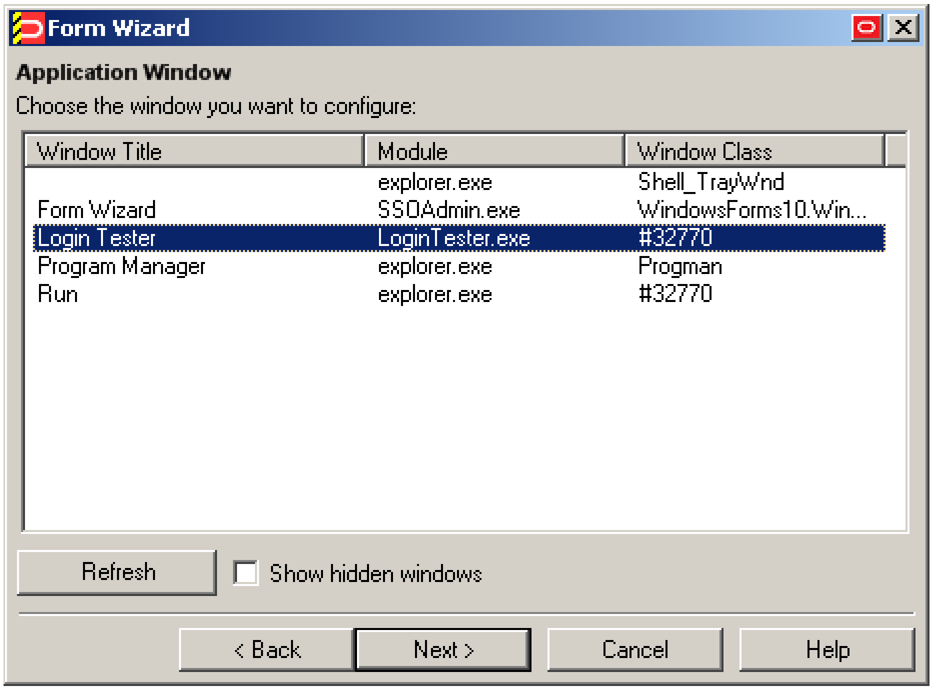
Form Wizard

Form Type

Select the type of screen you want to configure:

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

b. In the “Application Window” screen, select the window that contains the target form. A thick blue border appears around the chosen window to indicate your selection.



Application Window

Choose the window you want to configure:

| Window Title | Module | Window Class |
|---|---|---|
| | explorer.exe | Shell_TrayWnd |
| Form Wizard | SSOAdmin.exe | WindowsForms10.Win... |
| Login Tester | LoginTester.exe | #32770 |
| Program Manager | explorer.exe | Progman |
| Run | explorer.exe | #32770 |

**Note:** If you see two or more windows that share the same value for any of the parameters, the Agent might erroneously respond to all of those windows, instead of only the target window. For more information on this issue, see Troubleshooting Window Detection.

c. In the “Credential Fields” screen, select and configure the target fields and controls from among the objects in the list as follows:

i. (Optional) If your form requires the injection of the same data into multiple fields (e.g., injecting the password into a **Password** and a **Confirm Password** field), enable the **Allow multiple field designation** option.

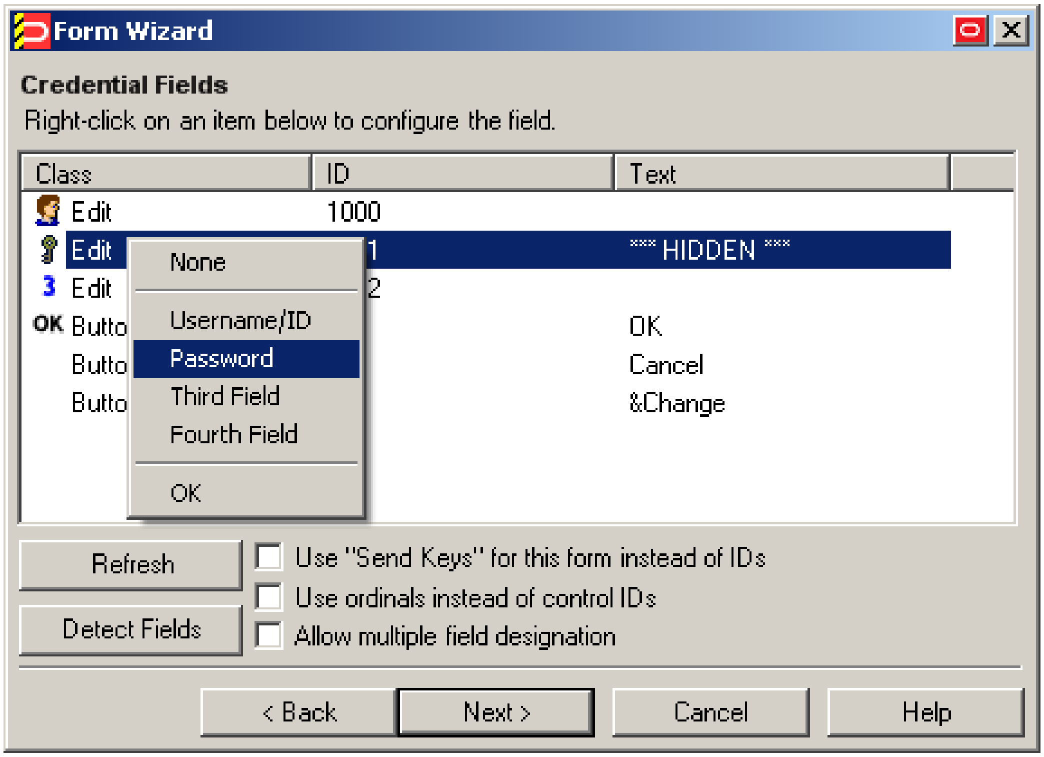
ii. Right-click each desired field or control and select its function from the context menu.

**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, Logon Manager will still send a “submit” action to the application after injecting credentials.

iii. If the application requires that Logon Manager interacts with its fields and controls using the “SendKeys” method, which emulates user input such as keystrokes and mouse clicks, select **Use “SendKeys” for this form instead of IDs**. To determine whether your application requires this option, see Understanding Form Response.)

iv. If the application requires that Logon Manager addresses its fields and controls via ordinals rather than Control IDs, select **Use ordinals instead of control IDs**. (To determine whether your application requires this option, see Understanding Form Response.)



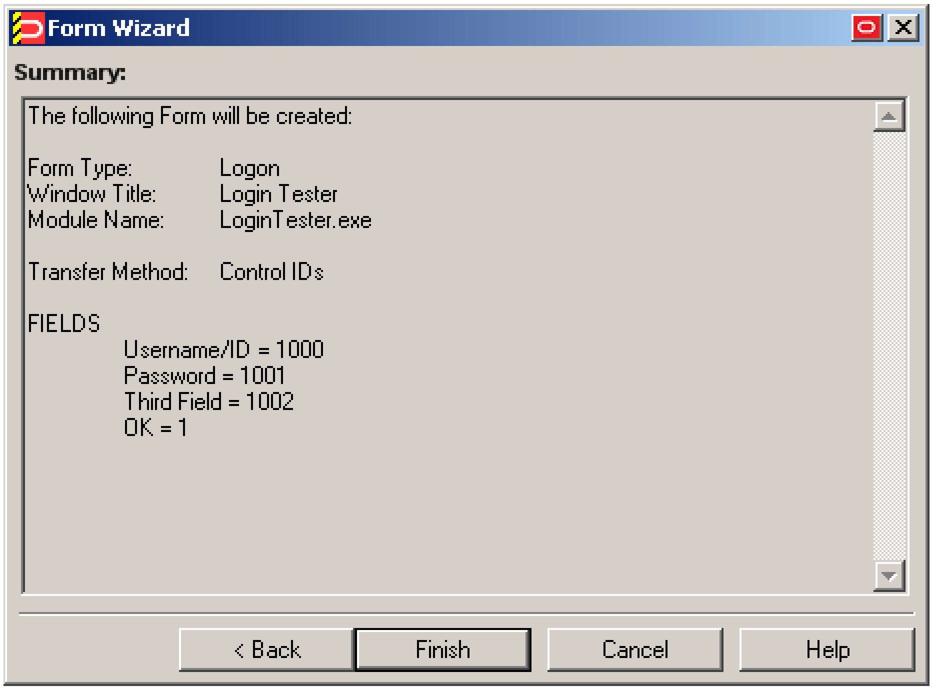
**Credential Fields**

Right-click on an item below to configure the field.

| Class | ID | Text |
|---|---|---|
| Edit | 1000 | |
| Edit | 1 | *** HIDDEN *** |
| 3 Edit | 2 | |
| OK Butto | | OK |
| Butto | | Cancel |
| Butto | | &Change |

☐ Use "Send Keys" for this form instead of IDs
☐ Use ordinals instead of control IDs
☐ Allow multiple field designation

d. In the “Summary” screen, review your configuration choices. If you want to change any of the options you selected, click **Back**, otherwise, click **Finish**.

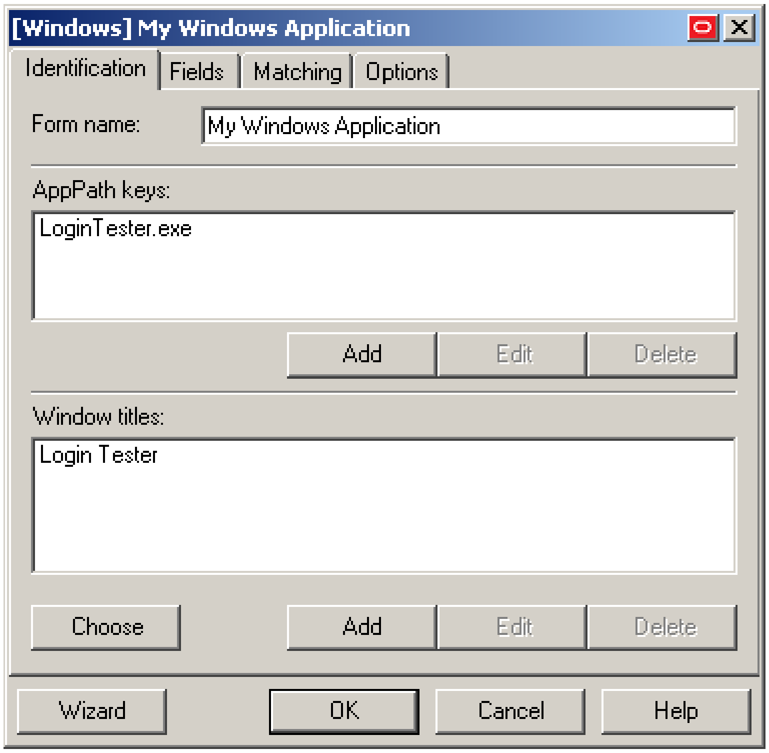


Form Wizard

Summary:

```
The following Form will be created:

Form Type:          Logon
Window Title:       Login Tester
Module Name:        LoginTester.exe

Transfer Method:    Control IDs

FIELDS
            Username/ID = 1000
            Password = 1001
            Third Field = 1002
            OK = 1
```

e. In the template properties dialog that appears, click **OK** to save the new form definition.



Form name: My Windows Application

AppPath keys: LoginTester.exe

Window titles: Login Tester

> **Note:** If you want to configure additional options for the template and you are familiar with their locations and functionalities, you may do so now, if you desire. Configuration procedures for options present in this dialog are included later in this guide.

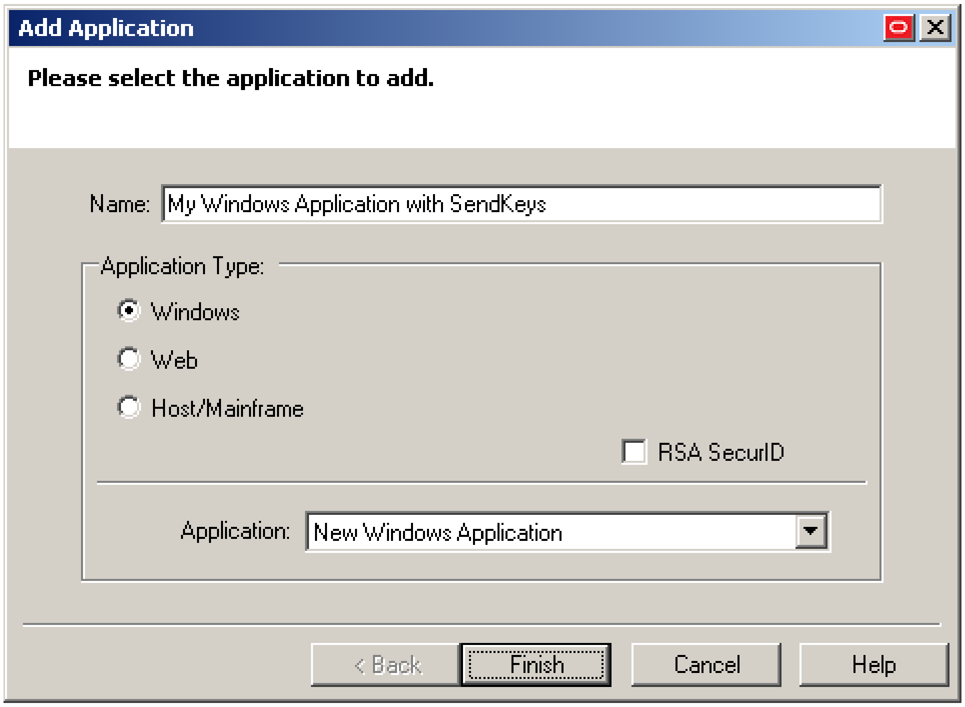
4. Publish your changes to the repository as described in Publishing a Template to the Repository, if applicable.

## Configuring a Form Using “SendKeys”

To configure a Windows application form that will use the SendKeys response method, do the following:

1. Open the Logon Manager Administrative Console. By default, the shortcut is located at **Start → Programs → Oracle → Logon Manager 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 the logon 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.



Add Application

Please select the application to add.

Name: My Windows Application with SendKeys

Application Type:

● Windows

○ Web

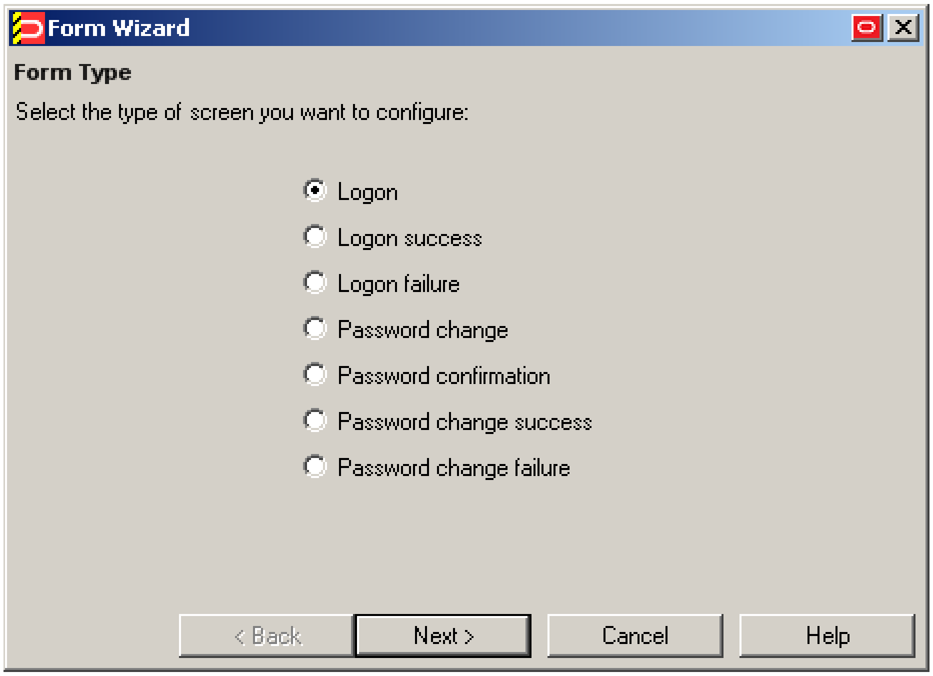
○ Host/Mainframe

□ RSA SecurID

Application: New Windows Application

    - If you want to add a logon 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 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**.



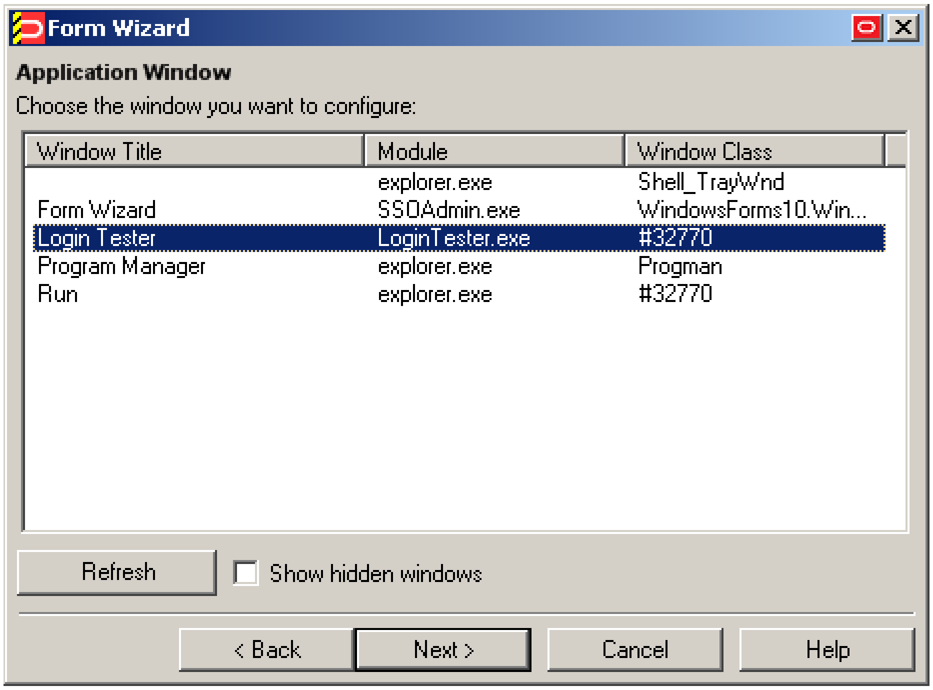
Form Wizard

**Form Type**

Select the type of screen you want to configure:

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

b. In the “Application Window” screen, select the window that contains the target form. A thick blue border appears around the chosen window to indicate your selection.



**Application Window**

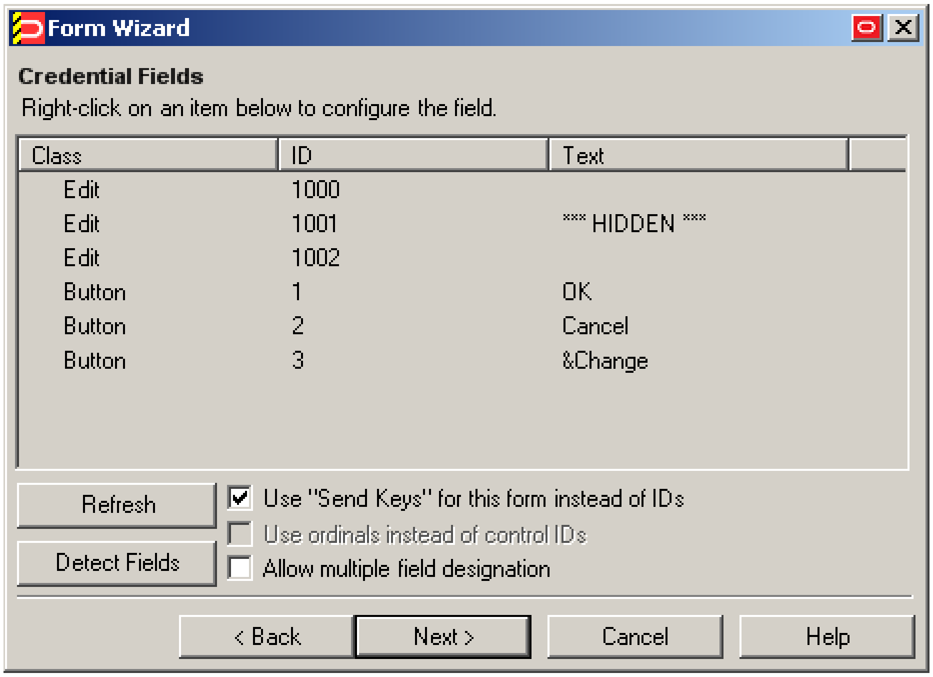
Choose the window you want to configure:

| Window Title | Module | Window Class |
|---|---|---|
| | explorer.exe | Shell_TrayWnd |
| Form Wizard | SSOAdmin.exe | WindowsForms10.Win... |
| Login Tester | LoginTester.exe | #32770 |
| Program Manager | explorer.exe | Progman |
| Run | explorer.exe | #32770 |

☐ Show hidden windows

**Note:** If you see two or more windows that share the same value for any of the parameters, the Agent might erroneously respond to all of those windows, instead of only the target window. For more information on this issue, see Troubleshooting Window Detection.

c. In the “Credential Fields” screen, select the **Use “Send Keys” for this form instead of IDs** check box and click **Next**.



**Credential Fields**

Right-click on an item below to configure the field.

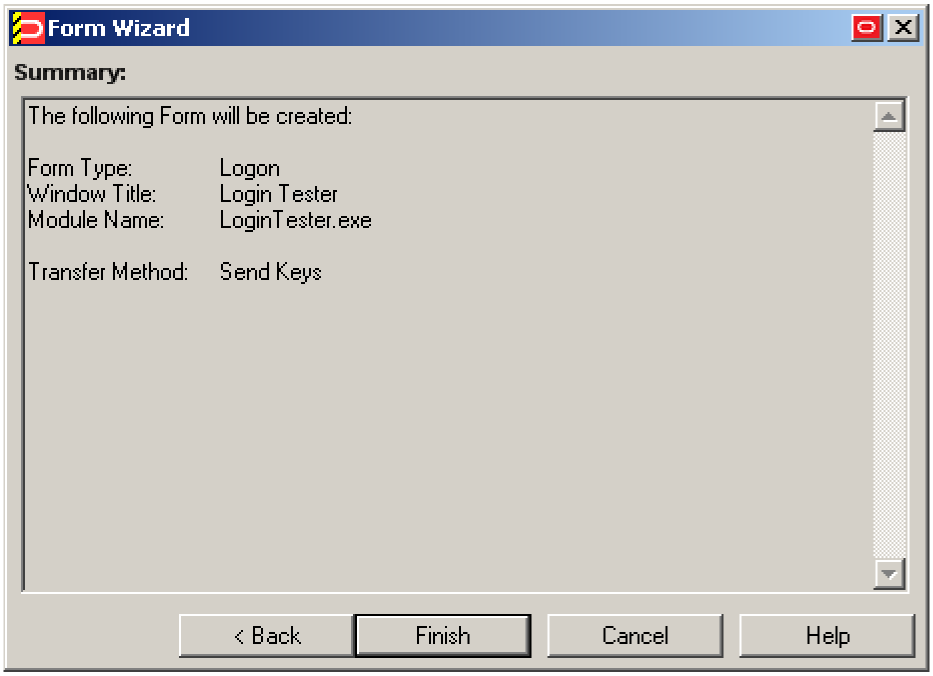
| Class | ID | Text |
|---|---|---|
| Edit | 1000 | |
| Edit | 1001 | *** HIDDEN *** |
| Edit | 1002 | |
| Button | 1 | OK |
| Button | 2 | Cancel |
| Button | 3 | &Change |

☑ Use "Send Keys" for this form instead of IDs

☐ Use ordinals instead of control IDs

☐ Allow multiple field designation

d. In the “Summary” screen, review your configuration choices. If you want to change any of the options you selected, click **Back**, otherwise, click **Finish**.

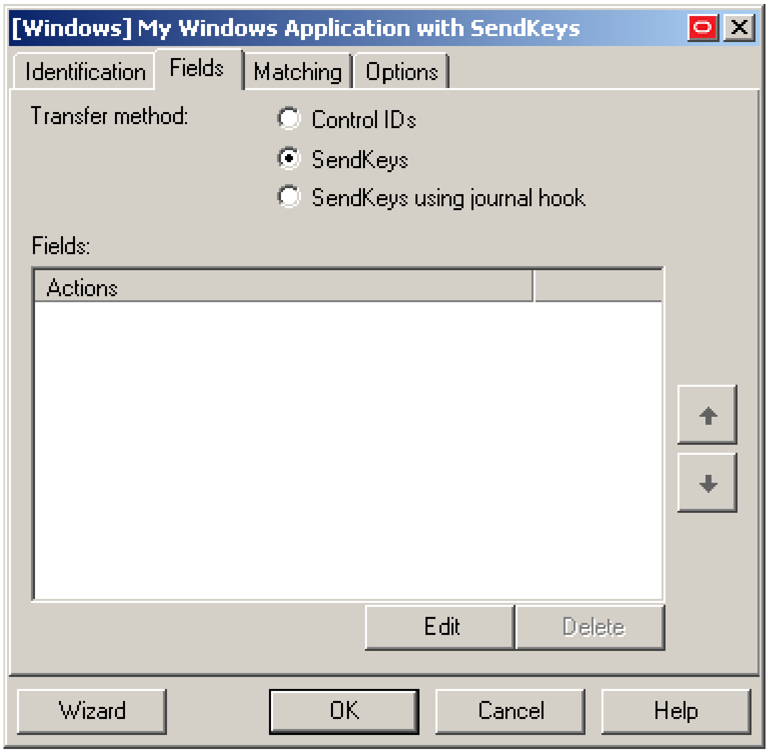


Summary:

The following Form will be created:

| | |
|---|---|
| Form Type: | Logon |
| Window Title: | Login Tester |
| Module Name: | LoginTester.exe |
| Transfer Method: | Send Keys |

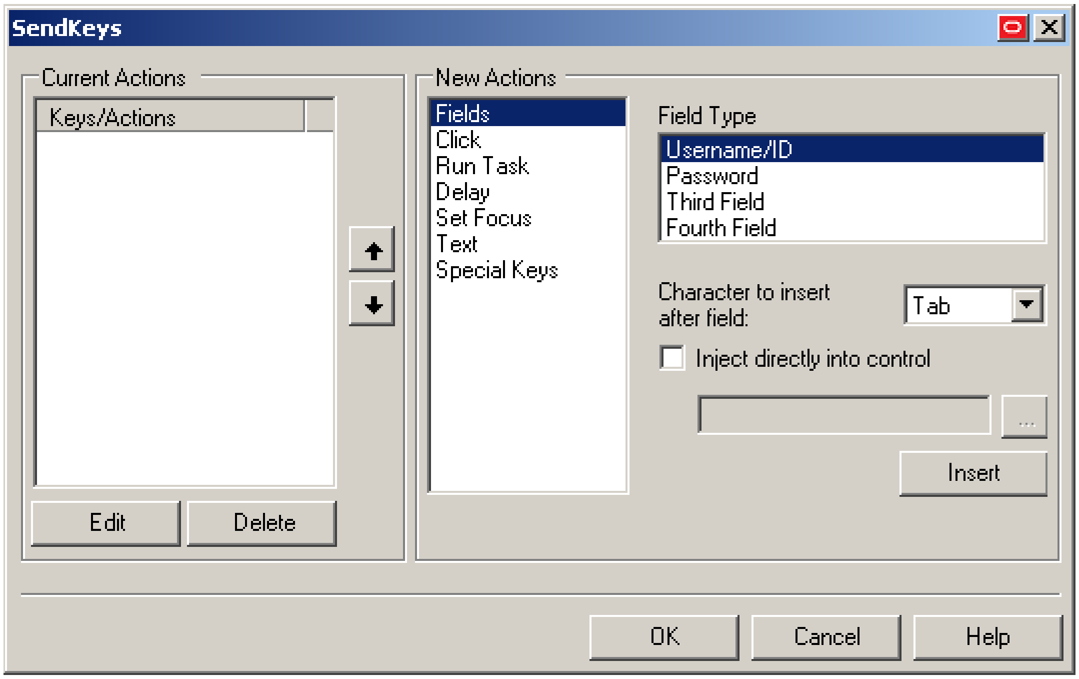
e. In the template properties dialog that appears, select the **Fields** tab.



Transfer method:
○ Control IDs
● SendKeys
○ SendKeys using journal hook

Fields:

f. Click **Edit** to display the SendKeys editor dialog.



g. Configure your SendKeys actions as desired. For detailed descriptions of each option, see the Console help.

> **Note:** To configure an action as programmatic (i.e., using the Control IDs response method) while the form is in "SendKeys" mode, enable the **Inject directly into control** option for the action.

h. When you have finished, click **OK** to save your changes and dismiss the SendKeys editor dialog.

> **Note:** If you want to configure additional options for the template and you are familiar with their locations and functionalities, you may do so now, if you desire. Configuration procedures for options present in this dialog are included later in this guide.

i. Click **OK** in the form properties dialog to save your form configuration.

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

## Using Matching to Improve Response Accuracy

Matching is a mechanism that allows Logon Manager to uniquely distinguish a window and the form it contains from other windows and forms that may be encountered in a session. The term "matching" refers to the comparison of the values of certain parameters, such as window title or a field label, to the value stored in a template.

For example, an application might instantiate dialog boxes that serve different purposes, such as informative messages, wizards, and logon forms, but share the same window title and class; in such a case, you would configure matching against unique elements within the form to limit Agent response to that specific form.

Logon Manager allows you to match against the values of the following criteria:

- Window title, including blank and partially or fully dynamic values
- Window class, including constriction to specific values
- Executable name (also referred to as a module name)
- A field, control, or a text string

### Matching for a Blank or Dynamic Value

You can use regular expressions supported by the Microsoft .NET Framework to formulate the text pattern that the Agent should recognize as a match for a criterion. Wildcards, such as ? (single character) and * (any combination of characters, excluding space), are the most commonly used regular expressions.

For example:

- `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.
- `Afx:400000:0:10011:0:*` will match against all window classes whose last six digits are variable. `Afx:400000:0:10011:0:130927` would be a match in this case.

If the target criterion, (for example, a window title) contains one or more system variables, such as the currently logged in user name, you can use the variable name as part of the matching text pattern. For example, the following pattern will result in a match against a window title that begins with `Password Expired` – and includes the currently logged on user's name in uppercase:

```
Password Expired - %UC%%DOMAINUSER%
```

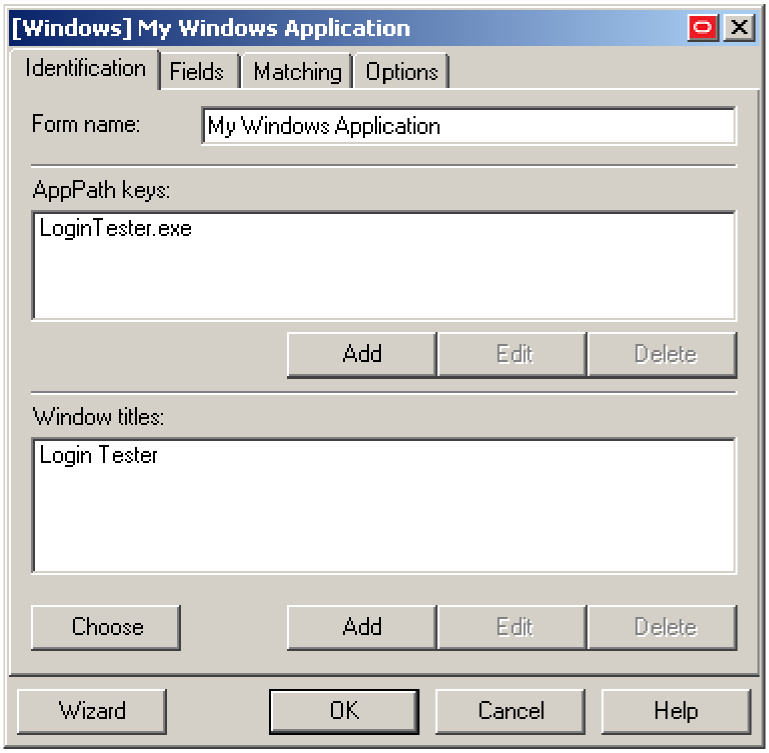
To match against:

- **A partially or fully dynamic value**, use regular expressions, environment variables, and static text, as appropriate, to create a text string that will cause a successful match against the target parameter value.
- **Against a blank value**, specify the null regular expression ^$ as the value.

> **Note:** Blank window titles are supported automatically; they do not require any special configuration.

Before completing the procedures in this section, do the following:

1. Launch the Logon Manager Administrative Console.
2. In the tree in the left-hand pane, expand the **Applications** node and select the desired template.
3. In the right-hand pane, select the **General** tab.
4. Click **Edit** to bring up the form properties dialog.


[Windows] My Windows Application

Form name: My Windows Application

AppPath keys: LoginTester.exe

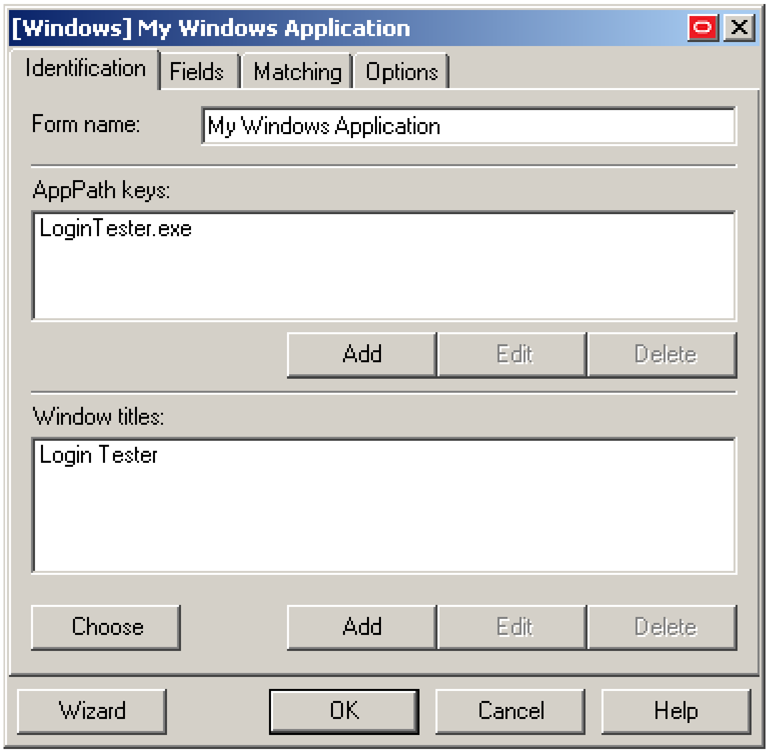
Window titles: Login Tester

5. Continue to the desired procedure:
   - Matching Against a Window Title
   - Matching Against a Window Class
   - Matching Against an Executable Name
   - Matching Against a Field, Control, or Text String

## Matching Against a Window Title

To match against one or more static or dynamic window titles, do the following:

1. In the template properties dialog, select the **Identification** tab.



[Windows] My Windows Application

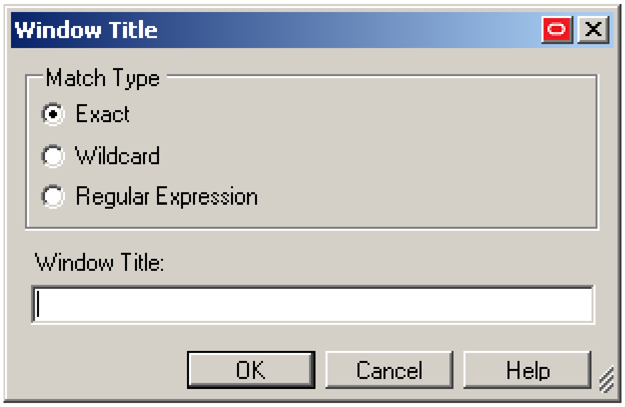
Form name: My Windows Application

AppPath keys: LoginTester.exe

Window titles: Login Tester

2. In the “Window Titles” section, do one of the following:
    - To add a new matching rule, click **Add**.
    - To modify an existing matching rule, select the rule in the list and click **Edit**.

3. In the “Window Title” dialog, do the following:



Window Title

Match Type

● Exact

○ Wildcard

○ Regular Expression

Window Title:

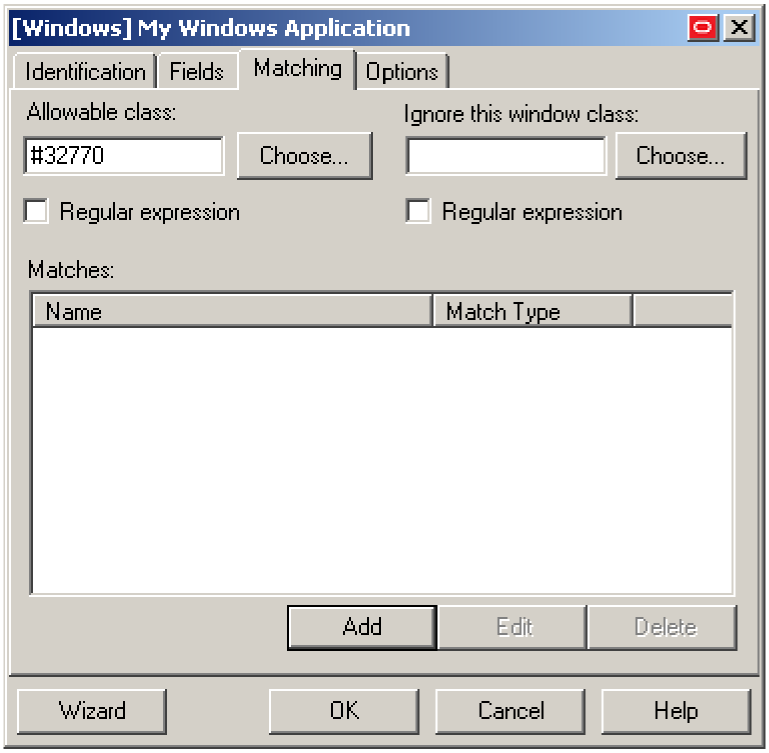
   a. Select the desired matching type:
      - **Exact** matches the string strictly as entered. Use with static window titles.
      - **Wildcard** allows matching using a combination of text and a wildcard. Use with blank or dynamic window titles.
      - **Regular expression** allows matching using a combination of text and regular expressions. Use with dynamic window titles.

      For more information, see Matching for a Blank or Dynamic Value.
   b. Enter the desired text string against which the Agent will perform the match.
   c. Click **OK** to add the new matching rule.
4. Click **OK** to close the properties dialog.

### Matching Against a Window Class

To match against one or more static or dynamic window classes, do the following:

1. In the form properties dialog, select the **Matching** tab.



Allowable class: #32770

☐ Regular expression

Ignore this window class:

☐ Regular expression

Matches:

| Name | Match Type | |
|---|---|---|

2. Specify the allowable window classes as follows:
   - If you don't know the exact value(s) of the class(es) you want to specify, do the following:
      i. In the **Allowable Class** field, click **Choose**.
      ii. In the dialog that appears, select the target window.
      iii. Click **OK**. The class of the selected window populates the **Allowable Class** field.
   - If you know the exact value(s) of the class(es) you want the Agent to recognize, enter them as a comma-delimited list into the **Allowable Class** field. If you want to match against one or more dynamic window classes, select the **Regular Expression** check box and include the desired regular expressions in the list.

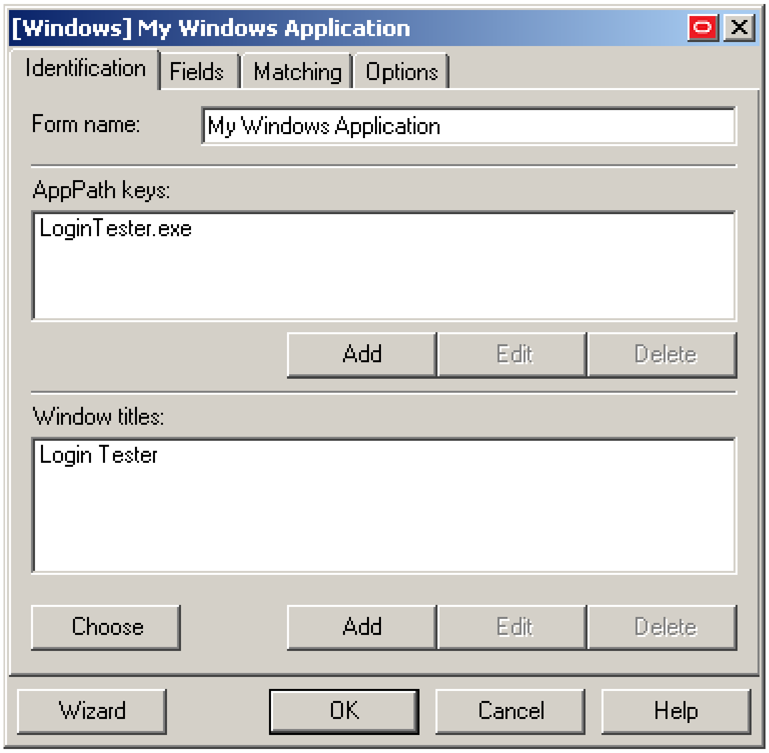
3. Specify the window classes you want the Agent to ignore (i.e., never respond to) as follows:
   - If you don’t know the exact value(s) of the class(es) you want to specify, do the following:
     i. In the **Ignore this window class** field, click **Choose**.
     ii. In the dialog that appears, select the target window.
     iii. Click **OK**. The class of the selected window populates the **Ignore this Window Class** field.
   - If you know the exact value(s) of the class(es) you want the Agent to recognize, enter them as a comma-delimited list into the **Ignore this Window Class** field. If you want to match against one or more dynamic window classes, select the **Regular Expression** check box and include the desired regular expressions in the list.
4. Click **OK** to close the properties dialog.

### Matching Against an Executable Name

To match against a specific executable name, do the following:

> **Note:** Matching against executable names supports exact matching only. Wildcards or regular expressions are not supported.

1. In the form properties dialog, select the **Identification** tab.



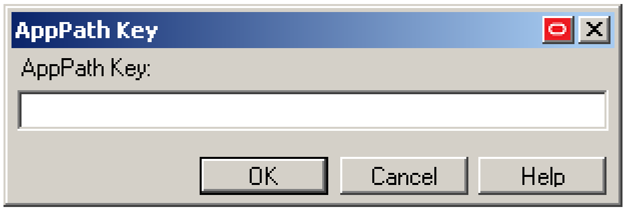
[Windows] My Windows Application

Form name: My Windows Application

AppPath keys: LoginTester.exe

Window titles: Login Tester

2. In the “AppPath Keys” section, do one of the following:
   - To add a new matching rule, click **Add**.
   - To modify an existing matching rule, select the rule in the list and click **Edit**.
3. In the “AppPath Key” dialog, enter the desired value and click **OK**.



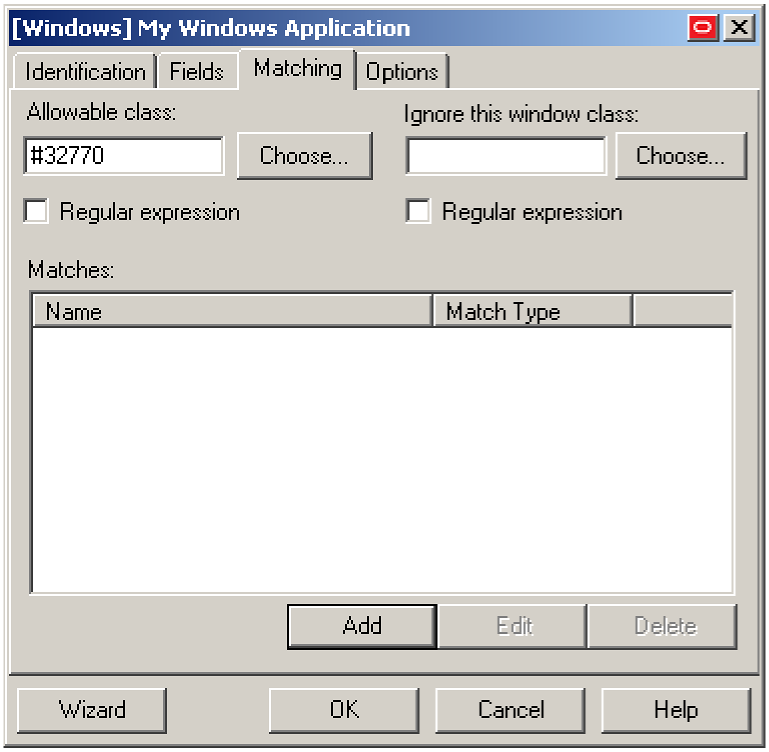
4. Click **OK** to close the template properties dialog.

### Matching Against a Field, Control, or Text String

Use the **Matching** tab to map user credentials of the currently selected logon to other logon, password-change or password-confirmation forms (referred to here as target forms) within the same application. The Agent uses the match criteria you supply to distinguish among similar forms that use the same credential data. This lets the Agent apply a single set of user credentials appropriately to these multiple forms. You can use also use matching to identify forms that the Agent should ignore.

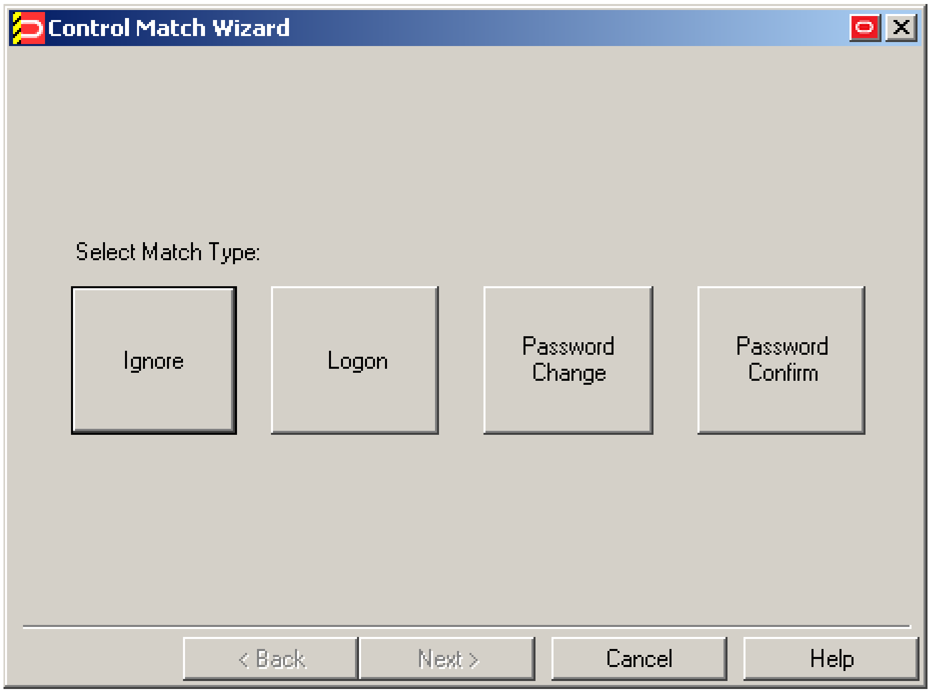
To match for a specific field or control, do the following:

1. In the template properties dialog, select the **Matching** tab.

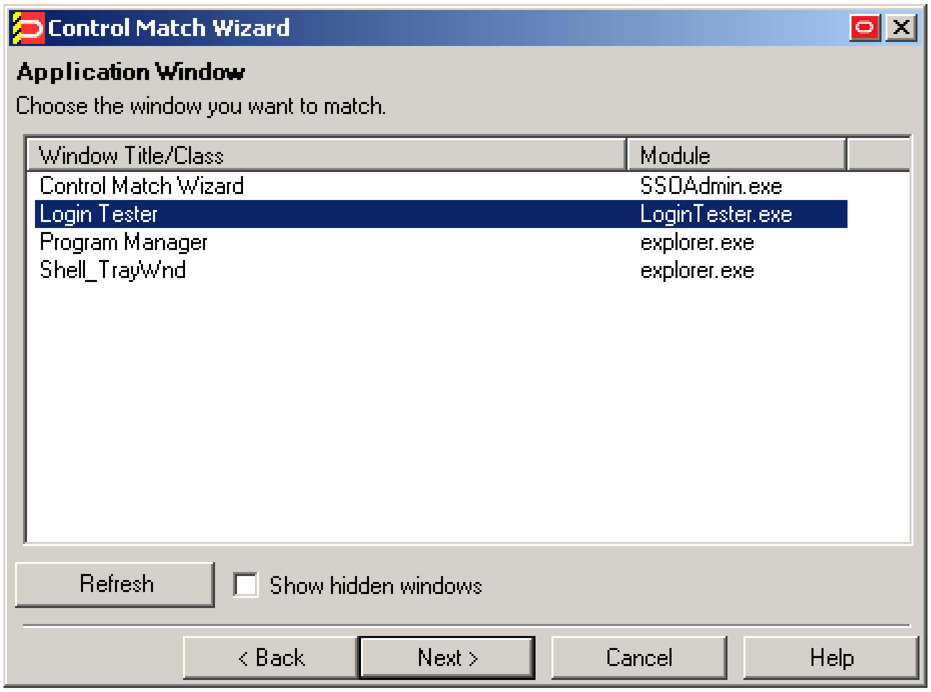


Allowable class: #32770

2. In the “Matches” section, do one of the following:
   - To add a new matching rule, click **Add**.
   - To modify an existing matching rule, select the rule in the list and click **Edit**.
3. In the Control Match Wizard that appears, select the desired matching type.



4. In the “Application Window” screen, select the target window.

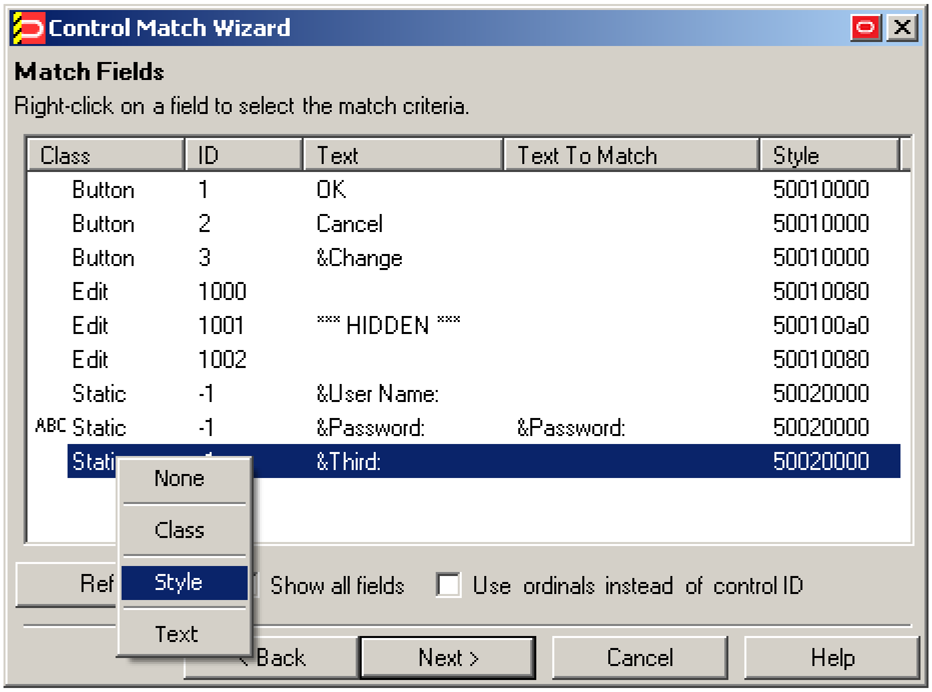


**Application Window**

Choose the window you want to match.

| Window Title/Class | Module |
|---|---|
| Control Match Wizard | SSOAdmin.exe |
| Login Tester | LoginTester.exe |
| Program Manager | explorer.exe |
| Shell_TrayWnd | explorer.exe |

5. In the “Match Fields” screen that appears, configure the desired match rules. For each field, control, or text string that you want to match against, do the following:
   a. In the list, select the desired item and right-click it.
   b. Select the desired match criterion from the context menu:
      - **Class:** instructs the Agent to match against the numeric class value of this item.
      - **Style:** instructs the Agent to match against the numeric style value of this item.
      - **Text:** instructs the Agent to match against the text presented by this item. When prompted, enter the desired match string into the dialog box that appears and click **OK**.
   c. Click **Next**.



**Match Fields**

Right-click on a field to select the match criteria.

| Class | ID | Text | Text To Match | Style |
|---|---|---|---|---|
| Button | 1 | OK | | 50010000 |
| Button | 2 | Cancel | | 50010000 |
| Button | 3 | &Change | | 50010000 |
| Edit | 1000 | | | 50010080 |
| Edit | 1001 | *** HIDDEN *** | | 500100a0 |
| Edit | 1002 | | | 50010080 |
| Static | -1 | &User Name: | | 50020000 |
| ABC Static | -1 | &Password: | &Password: | 50020000 |
| Stati | | &Third: | | 50020000 |

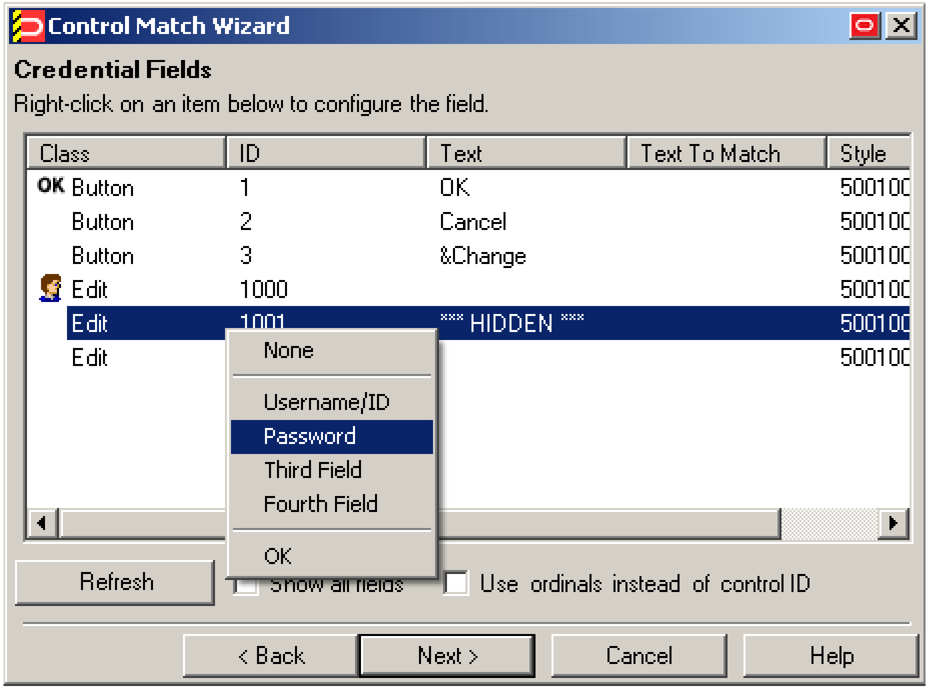
6. In the “Credential Fields” screen, select and configure the credential fields and controls that the Agent will use to complete the logon upon a successful match:
   a. Right-click each desired field or control and select its function from the context menu.

| | |
|---|---|
| **Note:** | If a “submit” button (usually labeled **OK**, **Logon**, etc.) is not visible in the list, Logon Manager will still send a “submit” action to the application after injecting credentials. |

   b. If the application requires that Logon Manager interacts with its fields and controls using the “SendKeys” method, which emulates user input such as keystrokes and mouse clicks, select **Use “SendKeys” for this form instead of IDs**. (To determine whether your application requires this option, see Understanding Form Response.)

c. If the application requires that Logon Manager addresses its fields and controls via ordinals rather than Control IDs, select **Use ordinals instead of control IDs**. (To determine whether your application requires this option, see Understanding Form Response.)

d. Click **Next**.



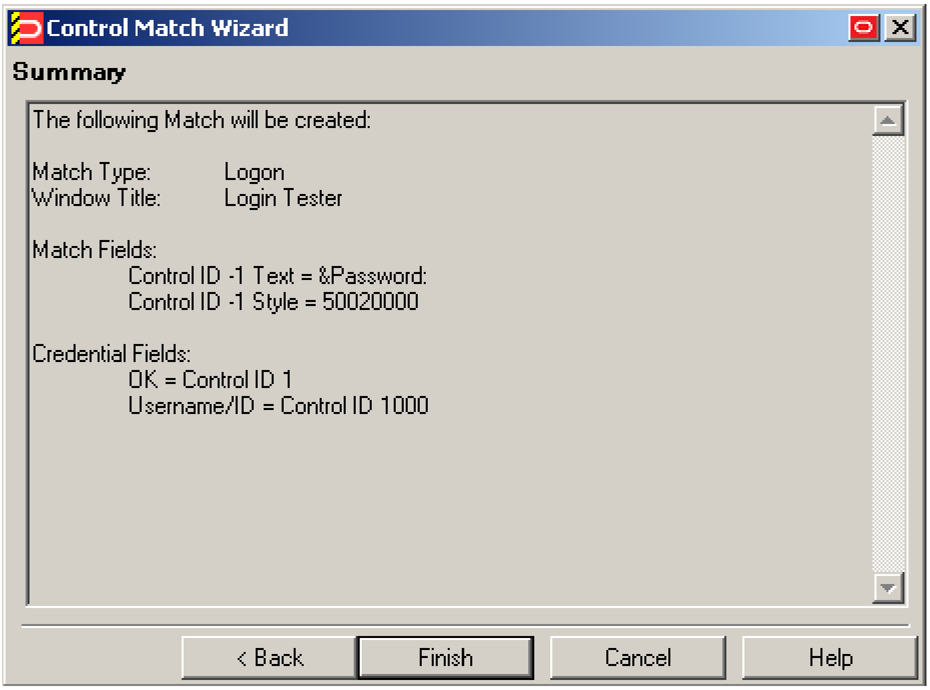
Control Match Wizard

**Credential Fields**

Right-click on an item below to configure the field.

| Class | ID | Text | Text To Match | Style |
|---|---|---|---|---|
| OK Button | 1 | OK | | 500100 |
| Button | 2 | Cancel | | 500100 |
| Button | 3 | &Change | | 500100 |
| Edit | 1000 | | | 500100 |
| Edit | 1001 | *** HIDDEN *** | | 500100 |
| Edit | | | | 500100 |

7. In the “Summary” screen, review your configuration choices. If you want to change any of the options you selected, click **Back;** otherwise, click **Finish**.



Control Match Wizard

**Summary**

```
The following Match will be created:

Match Type:        Logon
Window Title:      Login Tester

Match Fields:
          Control ID -1 Text = &Password:
          Control ID -1 Style = 50020000

Credential Fields:
          OK = Control ID 1
          Username/ID = Control ID 1000
```

## Configuring an Application as a Service Logon

If an application window changes its contents after it has been initially detected by Logon Manager, you must configure it as a service logon.

> **Note:** Enabling this option will cause Logon Manager to actively poll the application's window for updates, consuming additional CPU time (compounding with each additional window being polled), and should only be used if all other detection options have been exhausted.

To configure the application as a service logon, do the following:

1. Open the desired template.
2. Select the **Miscellaneous** tab and check the **Service Logon** check box.
3. Obtain the application's window class:
    a. Select the **General** tab,
    b. In the list of form definitions, select the logon form and click **Edit**.
    c. In the form properties dialog, select the **Matching** tab and note down the value present in the **Allowable Class** field – this is the window class detected in the application by Logon Manager.
4. Save your changes and push the updated template to the repository, if applicable.
5. Add the application's window class to the list of window classes the Agent will recognize as system services:
    a. Load your global Agent settings set.
    b. In the left-hand tree, navigate to **Global Agent Settings → End-User Experience → Windows Apps**.
    c. Select the check box next to the **Supported Window Classes for Services** option, if it is not already selected.
    d. In the field next to the above option, add the window class you noted down in step 3c to the list of existing classes, separated by a semicolon.
6. Save your changes and push the updated **Supported Window Classes for Services** setting to the repository as part of your administrative overrides.

## Disabling Automatic "SendKeys" Fallback

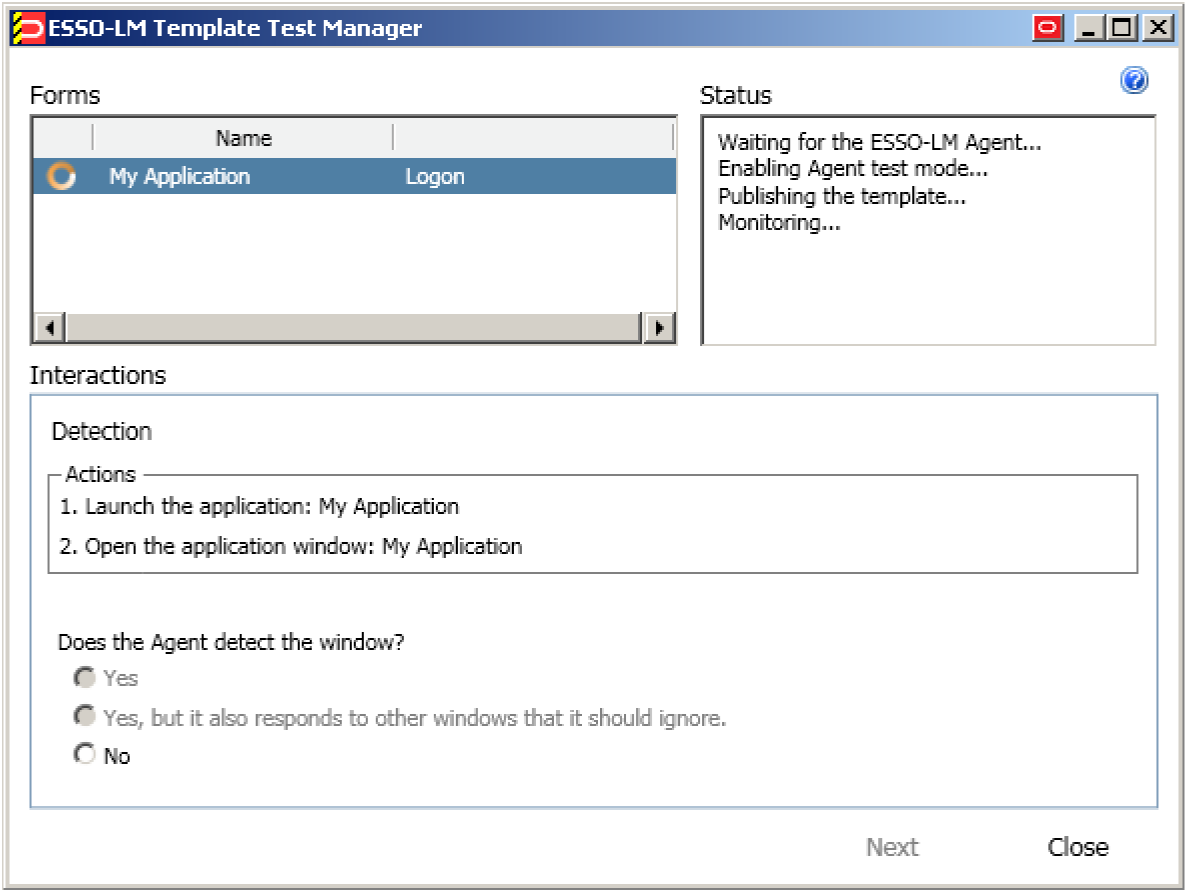
If Logon Manager is unable to respond to an application programmatically (i.e., using the "Control IDs" method), by default it will automatically fall back to the "SendKeys" response method and retry injection. In certain scenarios, this automatic fallback behavior might be undesirable – for example, Logon Manager might inadvertently inject credentials into the wrong window or application, exposing the credentials to the end-user. To disable this automatic fallback behavior, do one of the following:

- To disable automatic "SendKeys" fallback for a specific form definition, disable the **Fall back to SendKeys if direct injection fails** option in the **Options** tab in that form's properties dialog.
- To disable automatic "SendKeys" fallback globally, set the **Allow fallback from control IDs to SendKeys** option under **Global Agent Settings → <Target Settings Set> → User Experience → Application Response** to **No** and publish your changes to the repository.

## Testing the Configuration of a Form

Once you have created and configured a form, complete the following steps to test it before publishing:

1. In the left-hand tree, expand the **Applications** node and navigate to the target template.
2. Right-click the target template and select **Test** from the context menu.
3. In the “Logon Manager Template Test Manager” window that appears, do the following:
    a. In the **Forms** pane, select the target form.
    b. Follow the instructions displayed in the **Interactions** pane.



Waiting for the ESSO-LM Agent...
Enabling Agent test mode...
Publishing the template...
Monitoring...

Detection

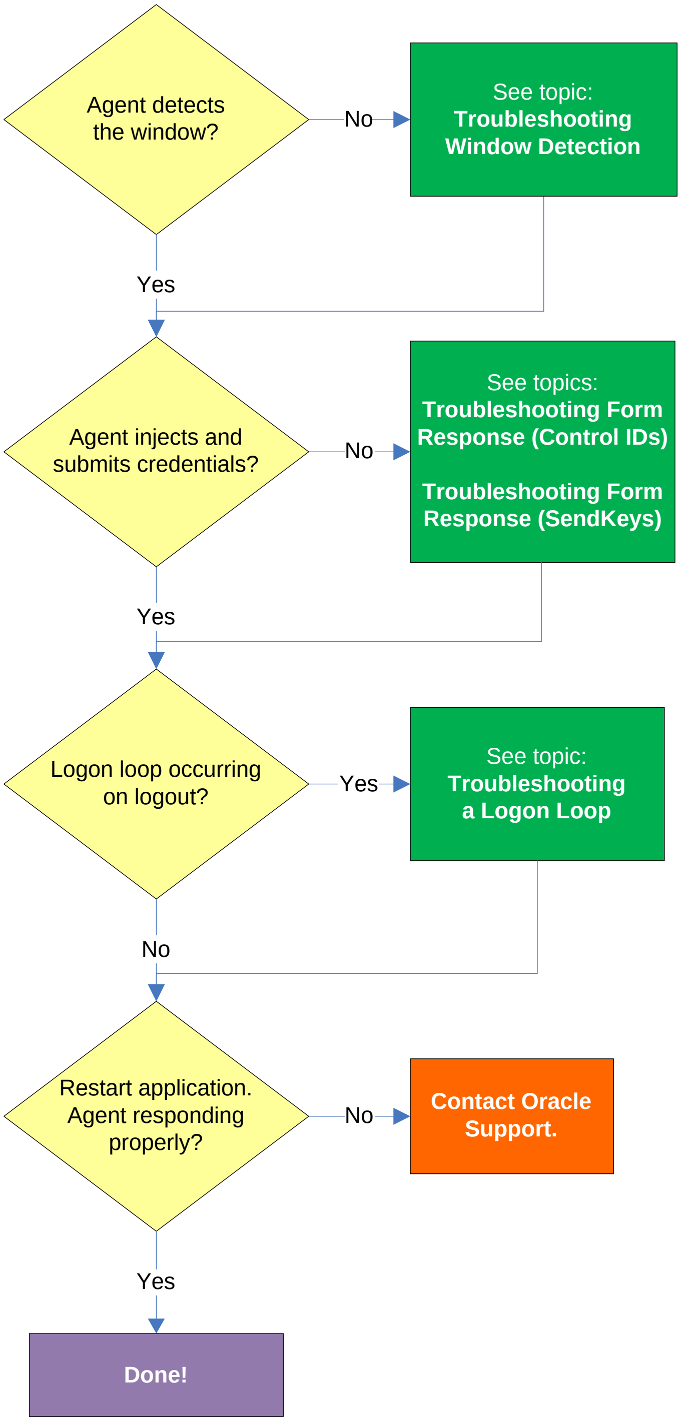
Actions
1. Launch the application: My Application
2. Open the application window: My Application

Does the Agent detect the window?

○ Yes
○ Yes, but it also responds to other windows that it should ignore.
○ No

c. Do one of the following:
    - If the Agent responds to the application as desired and the test has completed successfully, click **Finish**.
    - If the Agent is not responding to the application as desired, click **Close** and follow the troubleshooting flowcharts in this section to determine and correct the problem, then repeat steps 1-3 to test the corrected configuration.

## Testing the Configuration of a Logon Form


Agent detects the window?
No
See topic: Troubleshooting Window Detection
Yes
Agent injects and submits credentials?
No
See topics: Troubleshooting Form Response (Control IDs)
Troubleshooting Form Response (SendKeys)
Yes
Logon loop occurring on logout?
Yes
See topic: Troubleshooting a Logon Loop
No
Restart application. Agent responding properly?
No
Contact Oracle Support.
Yes
Done!

### Agent detects window?

Once the Agent has been provided with the template, it will automatically respond to the target application, unless the automatic response feature has been explicitly disabled. If the Agent fails to respond to the application, see Troubleshooting Window 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 Form Response When Using Control IDs and Troubleshooting Form Response When Using "SendKeys".

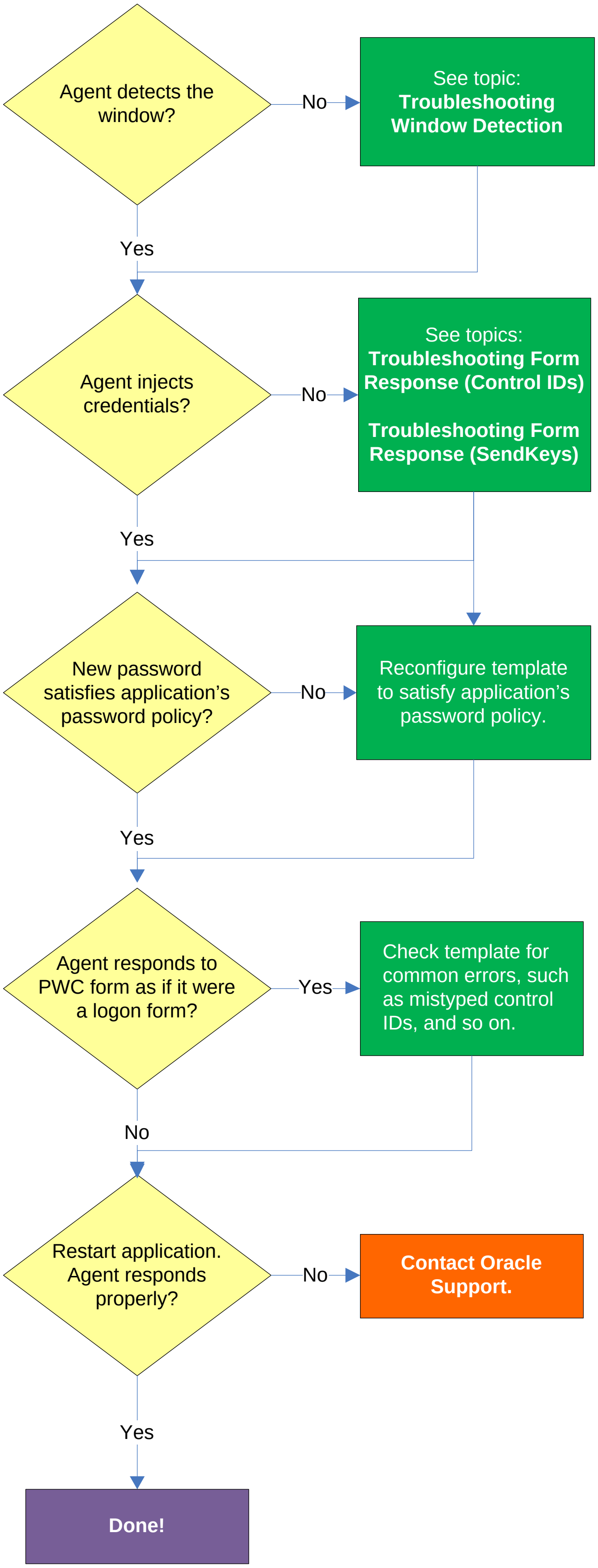
### Logon loop occurring on logout?

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.

### Agent responding properly after application is restarted?

If the target application is shut down and restarted, the Agent should respond to the application and log the user on. If logon does not occur, it is possible that the application is running in the system space instead of the user space and thus requires active polling instead of passive message queue monitoring by the Agent. If this is the case, you must follow the instructions in Configuring an Application as a Service Logon.

## Testing the Configuration of a Password Change Form


Agent detects the window?
No
See topic: **Troubleshooting Window Detection**
Yes
Agent injects credentials?
No
See topics: **Troubleshooting Form Response (Control IDs)**
**Troubleshooting Form Response (SendKeys)**
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 control IDs, and so on.
No
Restart application. Agent responds properly?
No
**Contact Oracle Support.**
Yes
**Done!**

### Agent detects the window?

Once the Agent has been provided with the template, it will automatically respond to the target application, unless the automatic response feature has been explicitly disabled. If the Agent fails to respond to the application, see Troubleshooting Window 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 Form Response When Using Control IDs and Troubleshooting Form Response When Using "SendKeys".

### New password satisfies application's password policy?

If the new password generated by Logon Manager 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 form type, incorrect field and control definitions, and so on.
- Check whether the password change form has a dynamic window title or class, and configure the template accordingly.
- If you are using matching, check whether you are using the correct matching type and examine your matching strings for errors.

## Publishing a Template to the Repository

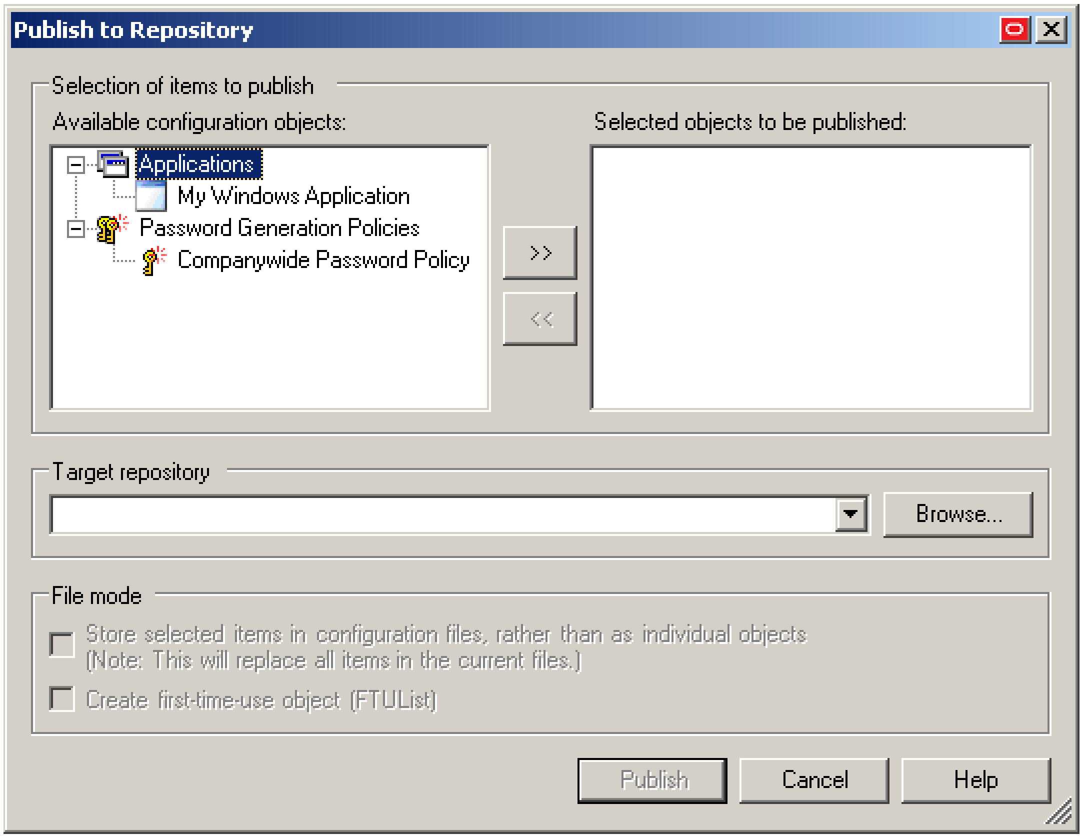
Once you have successfully tested your application template, you can distribute it to end-user machines by publishing it to the selected target container within your repository, either in a directory-style hierarchy (default), or as a flat configuration file.

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

**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:

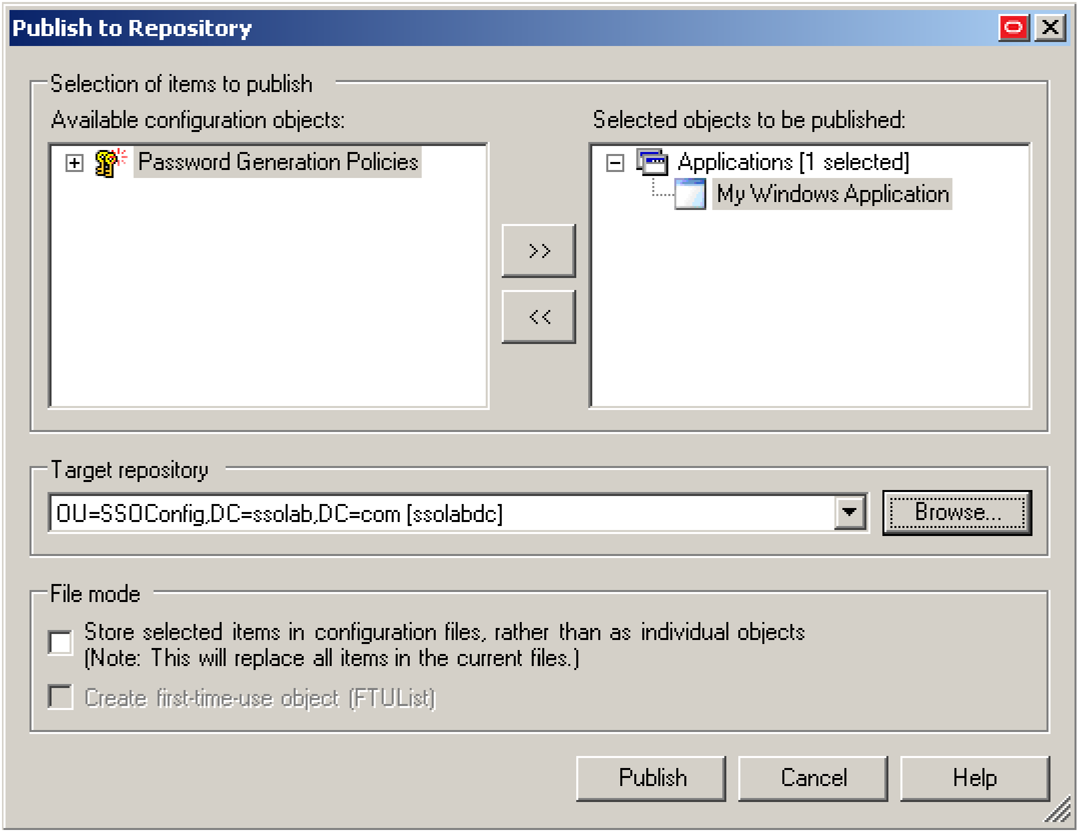
1. Launch the Logon Manager Administrative Console.
2. Right-click the **Applications** node and select **Publish…** from the context menu.
   The "Publish to Repository" dialog appears.


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

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 Windows Application
>>
<<
Target repository
OU=SSOConfig,DC=ssolab,DC=com [ssolabdc]
Browse...
File mode
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)
Publish
Cancel
Help

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

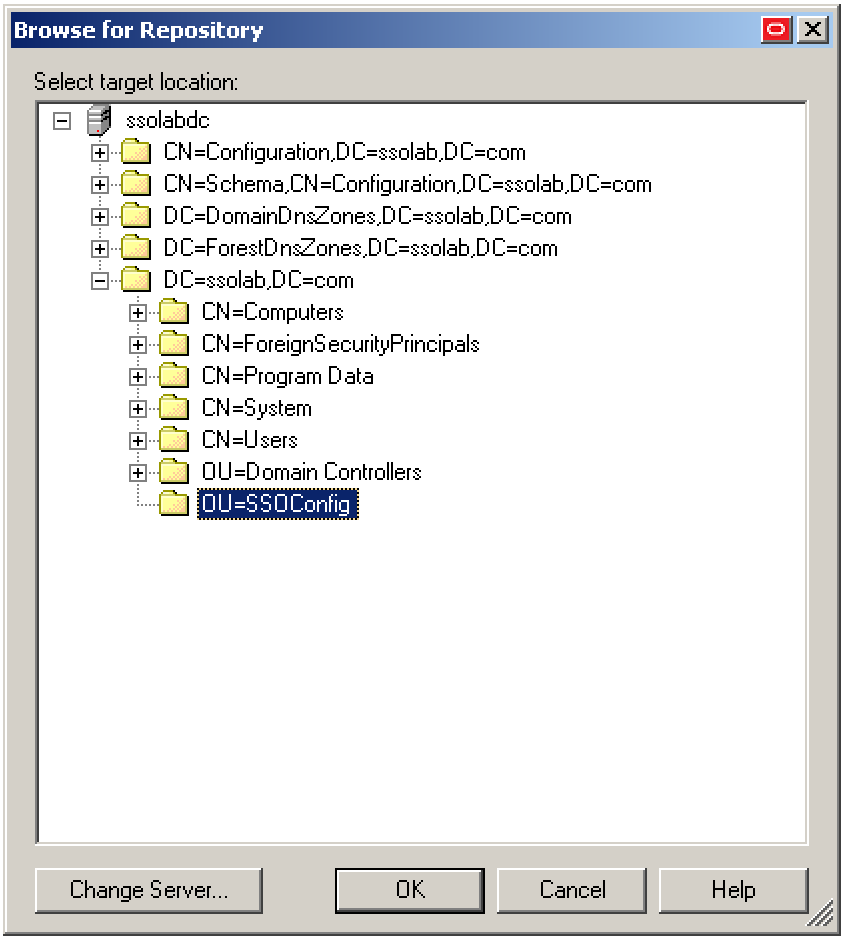
   - If you have previously published to the desired container, select it from the **Target Repository** drop-down list.

   - 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.



**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.

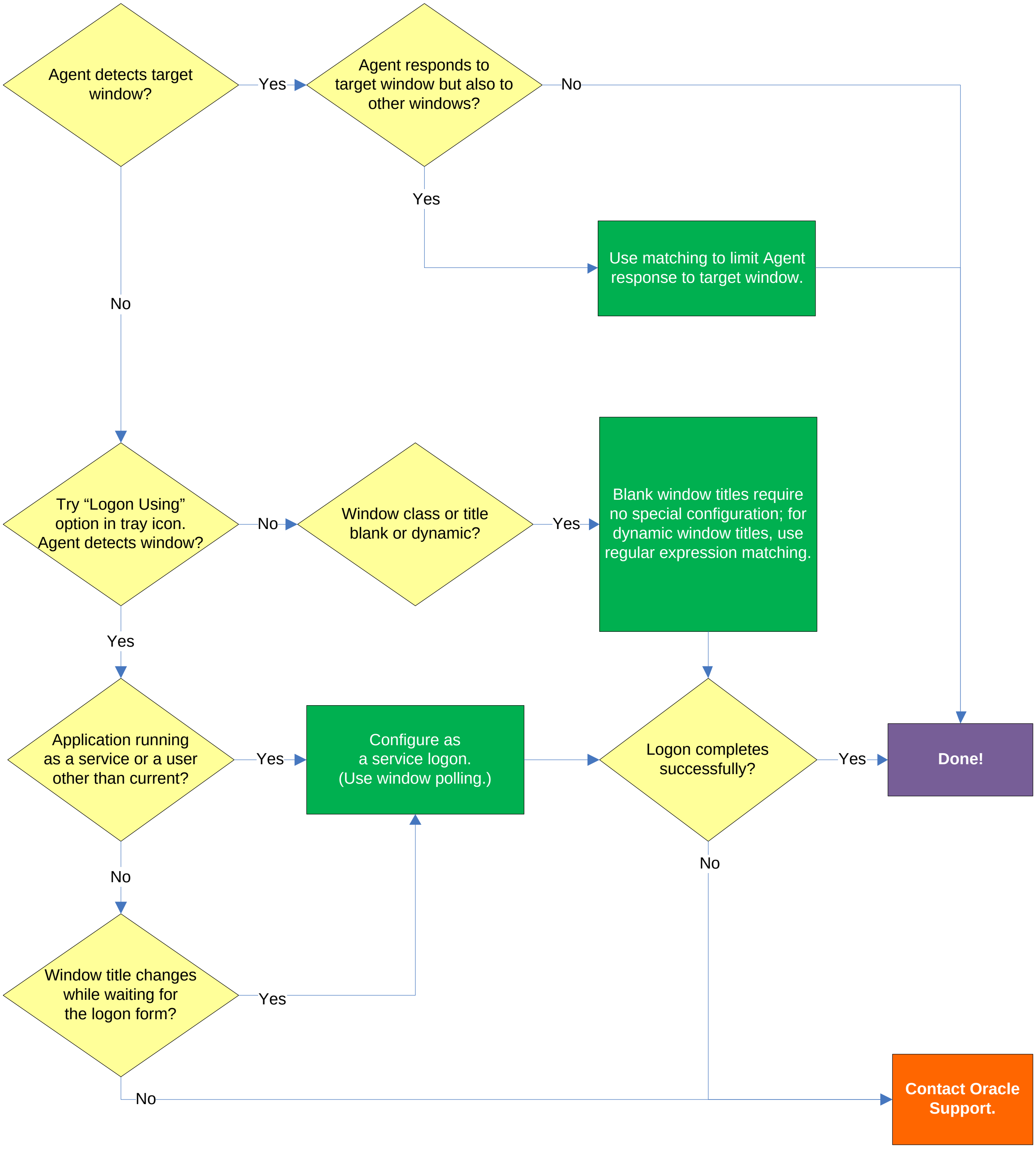
8. Click **Publish**. The Console publishes the selected objects to the target repository.

> **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 Logon Manager Administrative Console help.

## Troubleshooting Window Detection

Use the steps below to diagnose erratic window detection.


Agent detects target window?
Yes
Agent responds to target window but also to other windows?
No
Yes
Use matching to limit Agent response to target window.
No
Try "Logon Using" option in tray icon. Agent detects window?
No
Window class or title blank or dynamic?
Yes
Blank window titles require no special configuration; for dynamic window titles, use regular expression matching.
Yes
Application running as a service or a user other than current?
Yes
Configure as a service logon. (Use window polling.)
Logon completes successfully?
Yes
Done!
No
No
Window title changes while waiting for the logon form?
Yes
No
Contact Oracle Support.

### Agent detects the window?

If the Agent does not detect the window, first ensure that the **Auto-Recognize** option in the **Options** tab of the target form definition’s properties dialog is enabled; if it is, proceed to the next step.

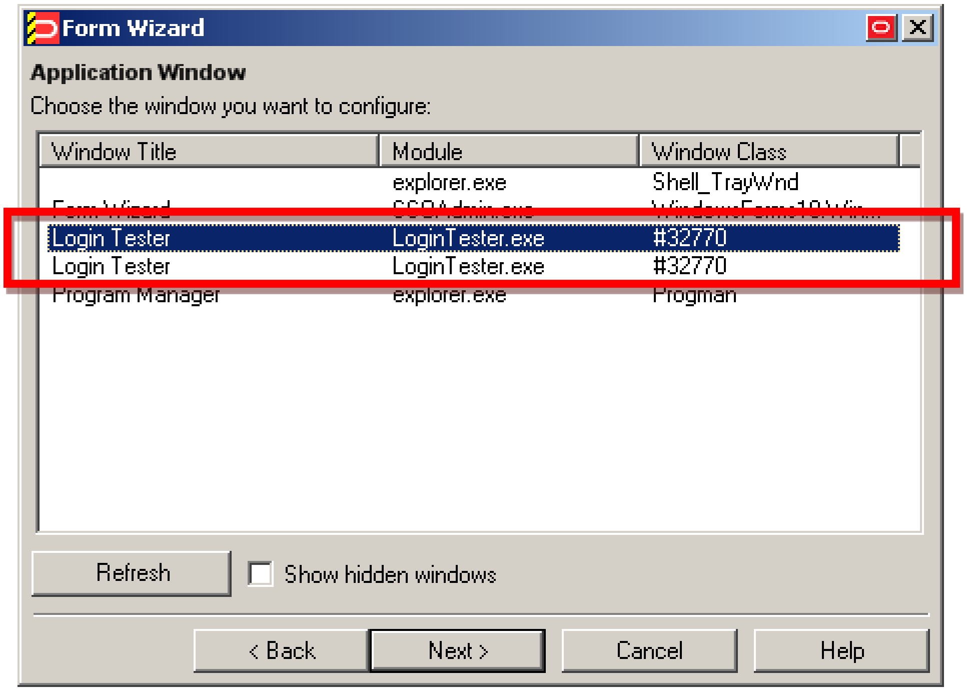
### Agent responds to target window but also to other windows?

Since the Agent will respond to all windows that possess the characteristics defined in the template, a default form configuration (created by simply completing the Form Wizard) may result in undesired response to windows that should be otherwise ignored. It is critical to configure the template to be as specific as possible so that the Agent can uniquely identify the target window.

> **Note:** This situation is sometimes referred to as a “duplicate event model” because the Agent is facing multiple sign-on events which it cannot uniquely distinguish from one another.

To decide whether more granular response control is necessary, examine the list in the Form Wizard window for duplicate window titles, module names, and window classes when creating a template. In the following example, two instances of the “Login Tester” application share their module (parent process) names and window class values, and will thus appear as the same application to the Agent.

In such cases, you must use matching to place more specific constraints on the values of the criteria that uniquely identify the window and form to the Agent.



**Application Window**

Choose the window you want to configure:

| Window Title | Module | Window Class |
|---|---|---|
| | explorer.exe | Shell_TrayWnd |
| Form Wizard | SSOAdmin.exe | WindowsForms10.Win... |
| Login Tester | LoginTester.exe | #32770 |
| Login Tester | LoginTester.exe | #32770 |
| Program Manager | explorer.exe | Progman |

### Agent detects window when using the “Logon Using Logon Manager” tray icon option?

Manually invoke window detection by using the **Logon Using Logon Manager** option from the Agent’s system tray icon, then do one of the following:

- If the Agent detects the window, you may have to configure the application as a service logon; continue to the next step.
- If the Agent does not detect the target window even when you manually invoke detection by using the **Logon Using Logon Manager** option from the Agent’s tray icon, review the template for common configuration errors, such as a mistyped window title or class; also, determine whether the window title and/or class are dynamic, and reconfigure the template as appropriate.

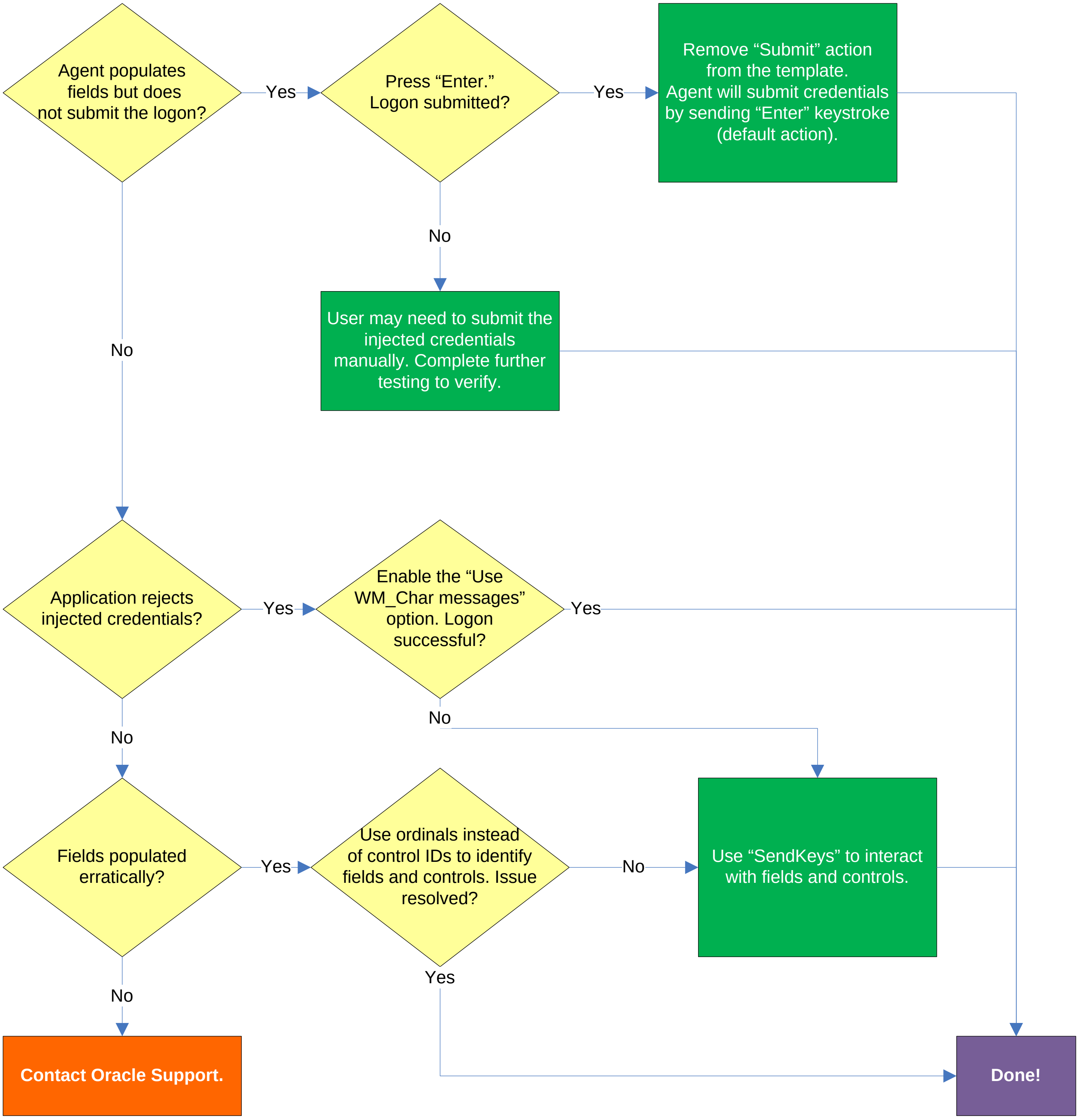
### Application running as a service or a user other than the current?

If the application is running in the system space (under the SYSTEM account), rather than in the user space (under the currently logged in user’s account), or the application has been launched under a user account different from the currently logged in user, you must configure it as a service logon. This allows the Agent to actively poll the application instead of passively responding to events in the currently logged on user’s Windows message queue. For instructions, see Configuring an Application as a Service Logon.

### Window title changes after detection?

If the title of the target window changes after the Agent has detected the window but before it begins responding to the window (for example, if the window title changes when the logon form is invoked), you must configure the application as a service logon. This allows the Agent to actively poll the application instead of passively responding to events in the currently logged on user’s Windows message queue. For instructions, see Configuring an Application as a Service Logon.

## Troubleshooting Form Response When Using Control IDs


Agent populates fields but does not submit the logon?
Yes
Press "Enter." Logon submitted?
Yes
Remove "Submit" action from the template. Agent will submit credentials by sending "Enter" keystroke (default action).
No
User may need to submit the injected credentials manually. Complete further testing to verify.
No
Application rejects injected credentials?
Yes
Enable the "Use WM_Char messages" option. Logon successful?
Yes
No
No
Fields populated erratically?
Yes
Use ordinals instead of control IDs to identify fields and controls. Issue resolved?
No
Use "SendKeys" to interact with fields and controls.
Yes
No
Contact Oracle Support.
Done!

### Agent populates fields but does not submit the logon?

If the credentials are inserted by not automatically submitted, first check that the **Auto-Submit** option is enabled for the application. If **Auto Submit** is enabled but the Agent still does not submit them to the application, press **Enter** after the Agent has populated the fields and see whether the credentials are submitted. If the credentials are submitted, remove the **Submit** action from the template – this will allow the Agent to use its default submit action, the **Enter** keystroke.

### Application rejects injected credentials?

If the application rejects the submitted credentials, enable WM_CHAR-style messaging in the application template – this causes the Agent to use an alternate API to interact with the fields and controls in the window. To do so, select the **General** tab in the template, select the target form, click **Edit**, and select the **Use WM_CHAR messages to fill controls** check box in the **Options** tab of the form properties dialog.

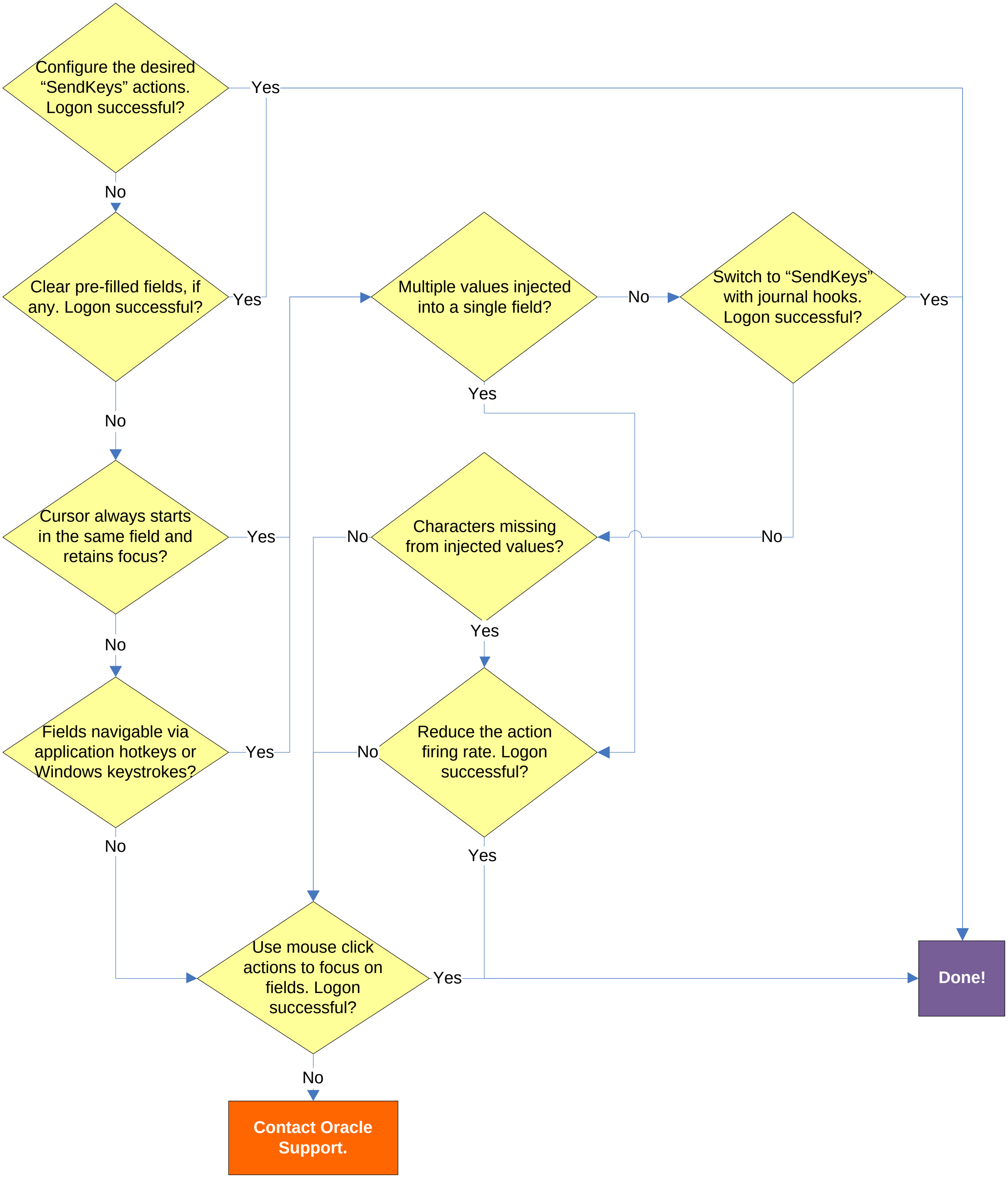
### Fields populated erratically?

If the Agent populates the fields erratically, i.e., inserts wrong, truncated, garbled, or blank values, one or more of the target Control IDs might be dynamic. In such case, use ordinals instead of Control IDs to uniquely identify fields and controls within the form. Ordinals are sequential ID numbers assigned by the Agent to each object in the window, from top to bottom, left to right, which allow the Agent to uniquely identify the detected fields and controls separately from Control IDs.

To switch a form from using Control IDs to using ordinals, select the form in the **General** tab in the template, click **Edit**, and click **Wizard** in the form properties dialog. Then, follow the steps in Basic Configuration to re-create the form definition but select the **Use ordinals instead of control IDs** check box when you arrive at the "Credential Fields" screen.  The new configuration choices you make in the wizard will overwrite the existing form definition.

If the issue persists even when using ordinals, consider using the "SendKeys" method of interacting with the application. For more information, see Supported Form Response Methods.

## Troubleshooting Form Response When Using “SendKeys”

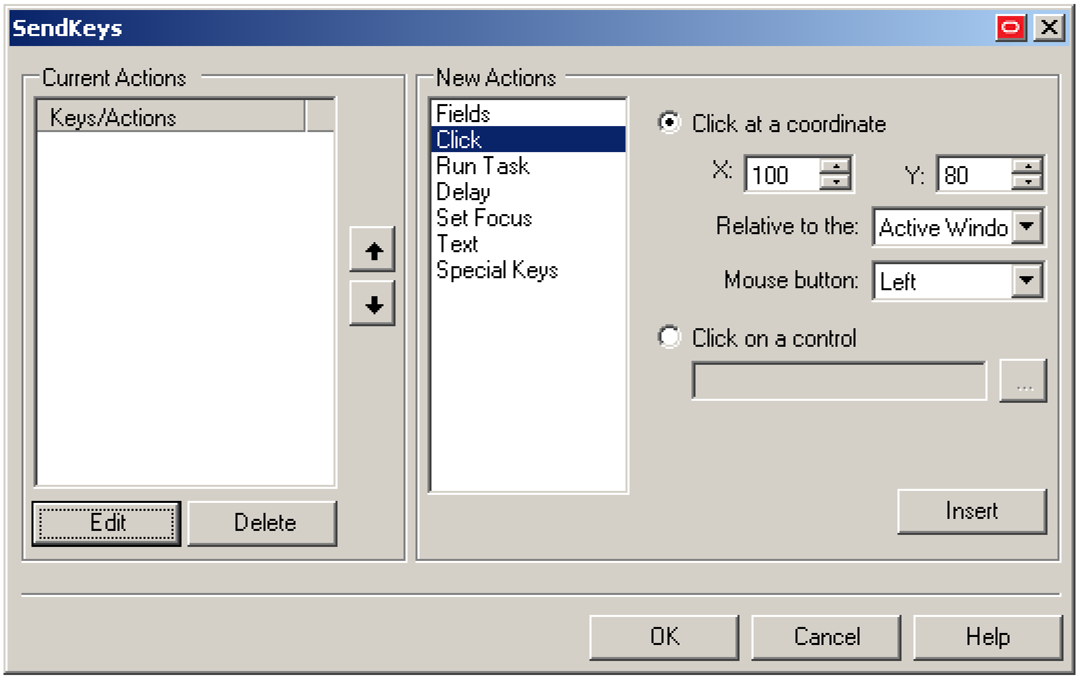

Configure the desired “SendKeys” actions. Logon successful?
Yes
No
Clear pre-filled fields, if any. Logon successful?
Yes
No
Cursor always starts in the same field and retains focus?
Yes
No
Fields navigable via application hotkeys or Windows keystrokes?
Yes
No
Multiple values injected into a single field?
No
Yes
Switch to “SendKeys” with journal hooks. Logon successful?
Yes
No
Characters missing from injected values?
No
Yes
Reduce the action firing rate. Logon successful?
No
Yes
Use mouse click actions to focus on fields. Logon successful?
Yes
No
Contact Oracle Support.
Done!

### Pre-filled fields cause erroneous logon?

Some applications might pre-fill the logon fields when the logon form is displayed – for example, the user name field might be pre-filled with the name of the last successfully logged on user. You may have to send one or more **Backspace** or **Delete** key strokes to clear such a pre-filled field before injecting credentials into it.

### Cursor always starts in the same field and retains focus?

If the cursor does not always start in the same field and the field loses focus before the Agent populates it, see if the application permits you to navigate to the field through a specific hotkey combination (such as **Alt+U**) or by using standard Windows keys, such as **Tab**, arrows, and so on. If you cannot use keystrokes to navigate to the field, use a mouse click action whose coordinates point to within the target field to allow the Agent to "click" within the field, as shown in the example below.



● Click at a coordinate

X: 100 Y: 80

Relative to the: Active Windo

Mouse button: Left

○ Click on a control

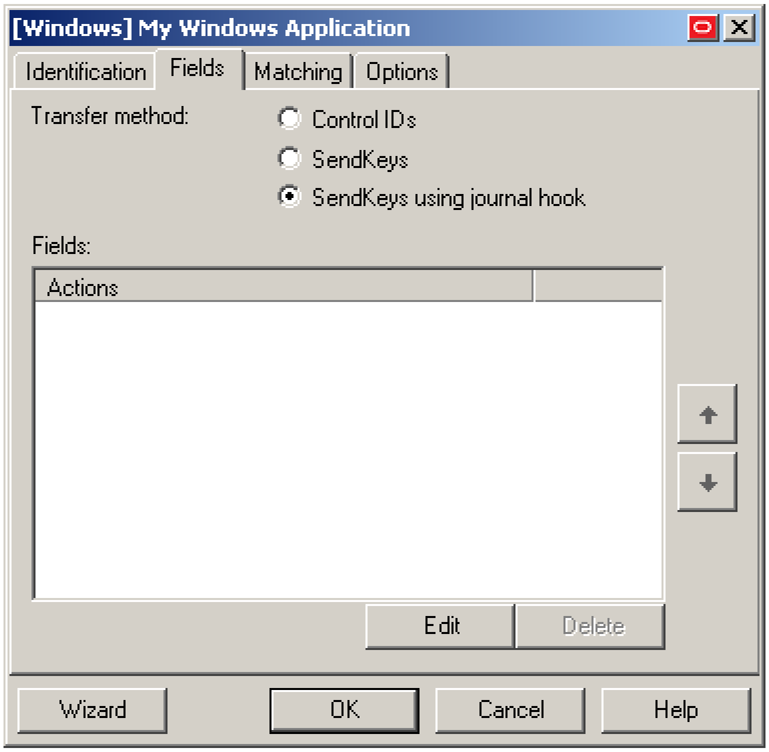
### Multiple values injected into a single field?

If the Agent is inserting multiple values (e.g., both the user name and the password) into a single field, it might be firing the "SendKeys" actions too quickly for the application to respond properly. In such cases, experiment with slowing down the action firing rate until credential injection is reliable. To do so, either insert a "Delay" action in between other actions, or set  the **SendKeys event interval** global Agent setting under **End-User Experience → Response** to either **Use for slow system** or **Use for very slow system**.

### Switching to “SendKeys” with journal hooks restores reliable injection?

If the Agent continues injecting multiple values into a single field after you have tried the suggestions in the previous step, switch the form interaction method to “SendKeys” with journal hooks. To do so, select the **General** tab in the application template, select the desired form, click **Edit**, select the **Fields** tab, and set the **Transfer Method** option to **SendKeys using Journal Hook**.

> **Note:** The “SendKeys” with journal hooks option causes the Agent to use an alternate API to send keystrokes and mouse clicks to the application; it is typically the most effective in Citrix environments.



Transfer method:
○ Control IDs
○ SendKeys
● SendKeys using journal hook

### Characters missing from injected values?

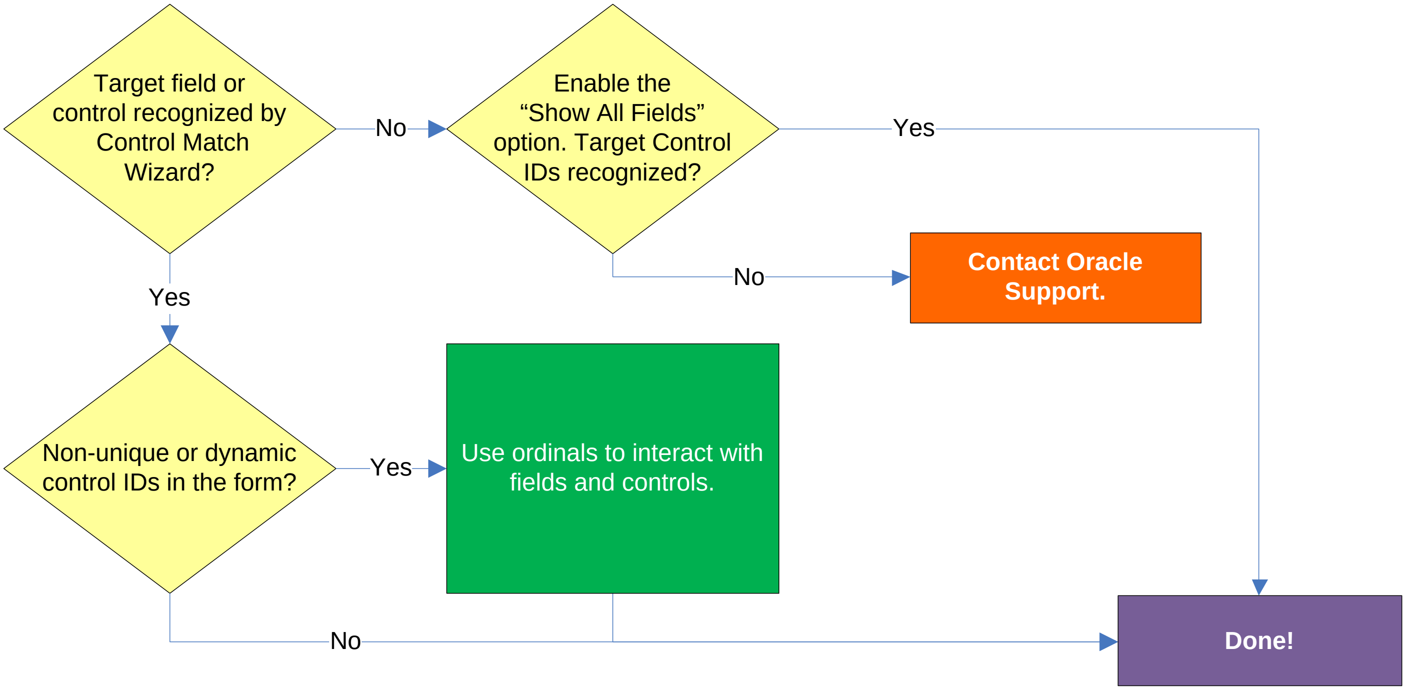
If you find that individual characters are omitted from the injected field values, the Agent might be firing the “SendKeys” actions too quickly for the application to accept them properly. In such cases, experiment with slowing down the action firing rate until credential injection is reliable. To do so, either insert a “Delay” action in between other actions or set  the **SendKeys event interval** global Agent setting under **End-User Experience → Response** to either **Use for slow system** or **Use for very slow system**.

### Using mouse click actions to focus on fields results in successful logon?

If logon is still unsuccessful, consider using mouse click actions to focus on all fields and controls within the form. Be aware that because each mouse click action requires exact coordinates of the field or control you want to click, those coordinates must remain static in order for the mouse click to succeed. Thus, mouse click actions might not be reliable for windows whose contents shift when the window is resized or displayed in different resolutions.

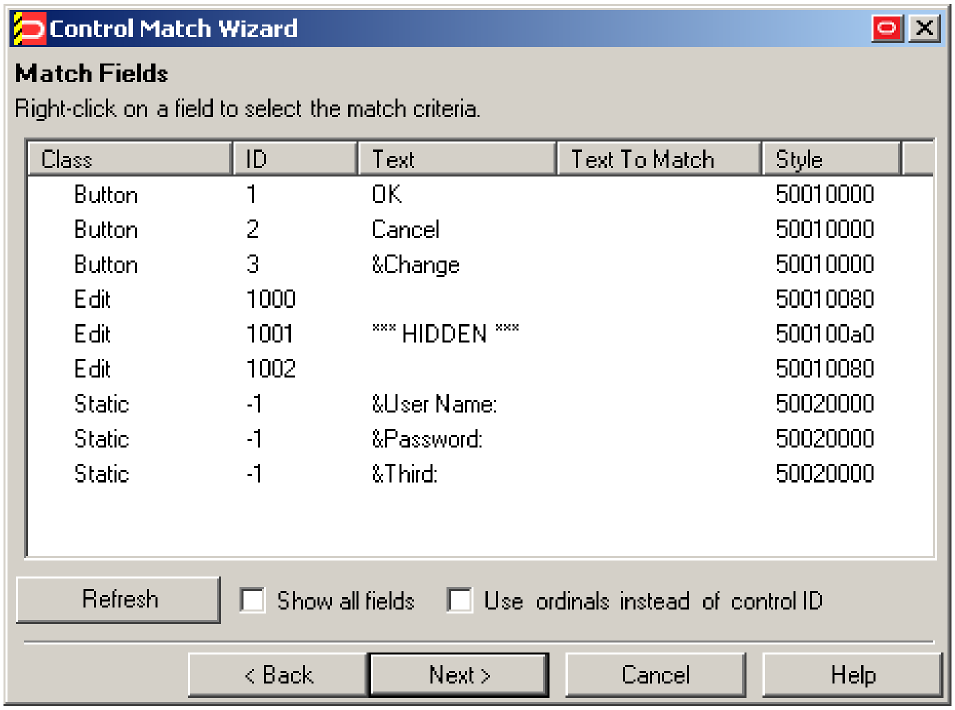
If none of the above steps resolve your issue, contact Oracle Support for assistance.

## Troubleshooting Matching


Target field or control recognized by Control Match Wizard?
No
Enable the "Show All Fields" option. Target Control IDs recognized?
Yes
No
Contact Oracle Support.
Yes
Non-unique or dynamic control IDs in the form?
Yes
Use ordinals to interact with fields and controls.
No
Done!

### Target field or control recognized by Control Match Wizard?

If the field or control targeted for matching do not appear in the Control Match Wizard, even after enabling the **Show all fields** option, matching might not be possible. In such cases, please contact [Oracle Support](#) for assistance.



**Match Fields**

Right-click on a field to select the match criteria.

| Class | ID | Text | Text To Match | Style |
|---|---|---|---|---|
| Button | 1 | OK | | 50010000 |
| Button | 2 | Cancel | | 50010000 |
| Button | 3 | &Change | | 50010000 |
| Edit | 1000 | | | 50010080 |
| Edit | 1001 | *** HIDDEN *** | | 500100a0 |
| Edit | 1002 | | | 50010080 |
| Static | -1 | &User Name: | | 50020000 |
| Static | -1 | &Password: | | 50020000 |
| Static | -1 | &Third: | | 50020000 |

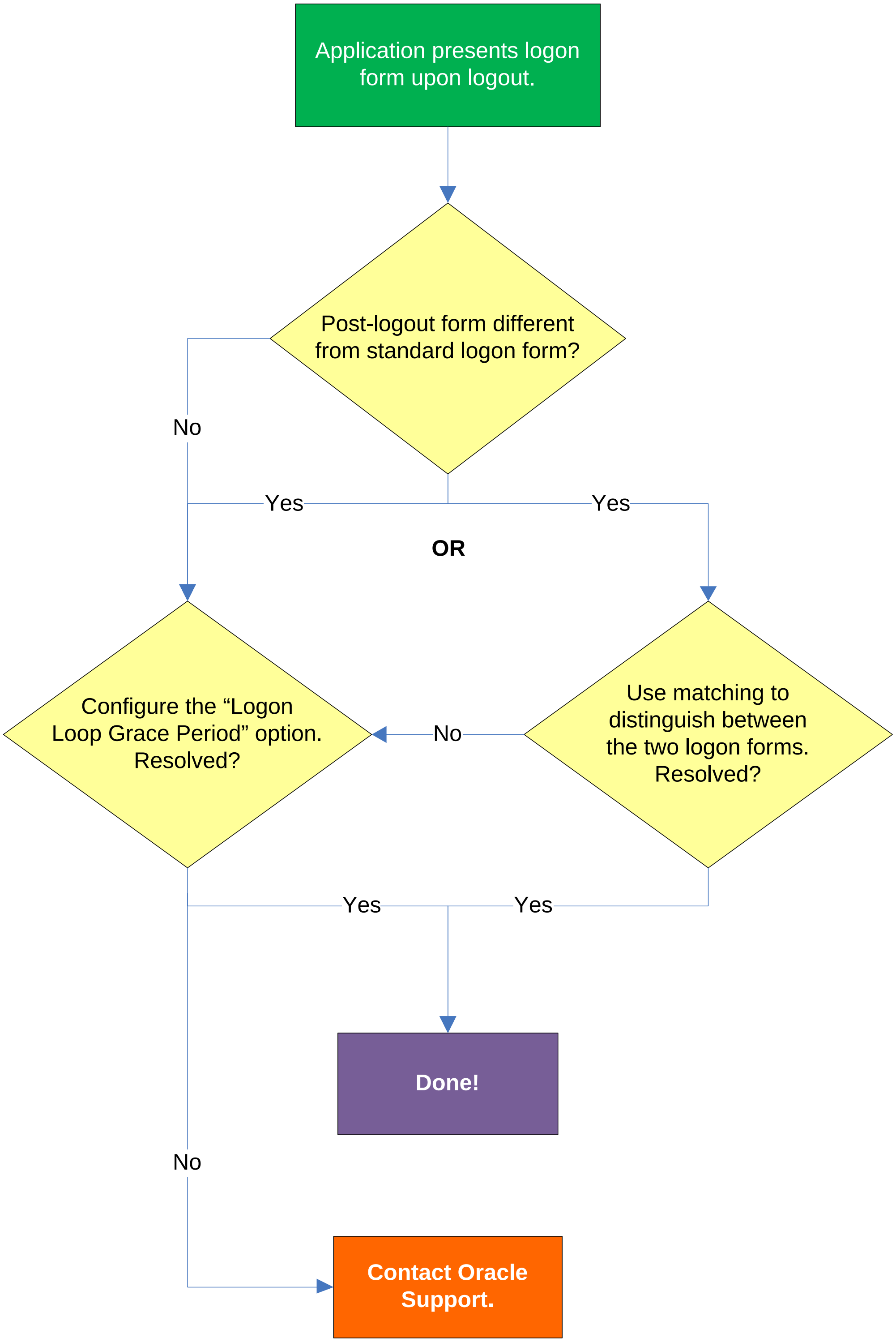
### Non-unique or dynamic Control IDs in the form?

If one or more of the target Control IDs are non-unique or dynamic, use ordinals instead of Control IDs to uniquely identify fields and controls within the form. Ordinals are sequential ID numbers assigned by the Agent to each object in the window, from top to bottom, left to right, which allow the Agent to uniquely identify the detected fields and controls separately from Control IDs.

To switch a form from using Control IDs to using ordinals, select the form in the **General** tab in the template, click **Edit**, and click **Wizard** in the form properties dialog. Then, follow the steps in Basic Configuration to re-create the form definition but select the **Use ordinals instead of control IDs** check box when you arrive at the "Credential Fields" screen.  The new configuration choices you make in the wizard will overwrite the existing form definition.

## Troubleshooting a Logon Loop

Some applications display their logon form upon logout, which causes Logon Manager 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 Logon Manager from logging on to an application within set time period since the last logon.


Application presents logon form upon logout.
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 different from standard logon form?

Oracle recommends that you consider the “Logon Loop Grace Period” as well as matching if the logon form presented upon logout is sufficiently different from the application’s standard logon form. (For more information on matching, see Using Matching to Improve Response Accuracy.) If the forms cannot be uniquely distinguished, use the “Logon Loop Grace Period” feature described below.

### 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:

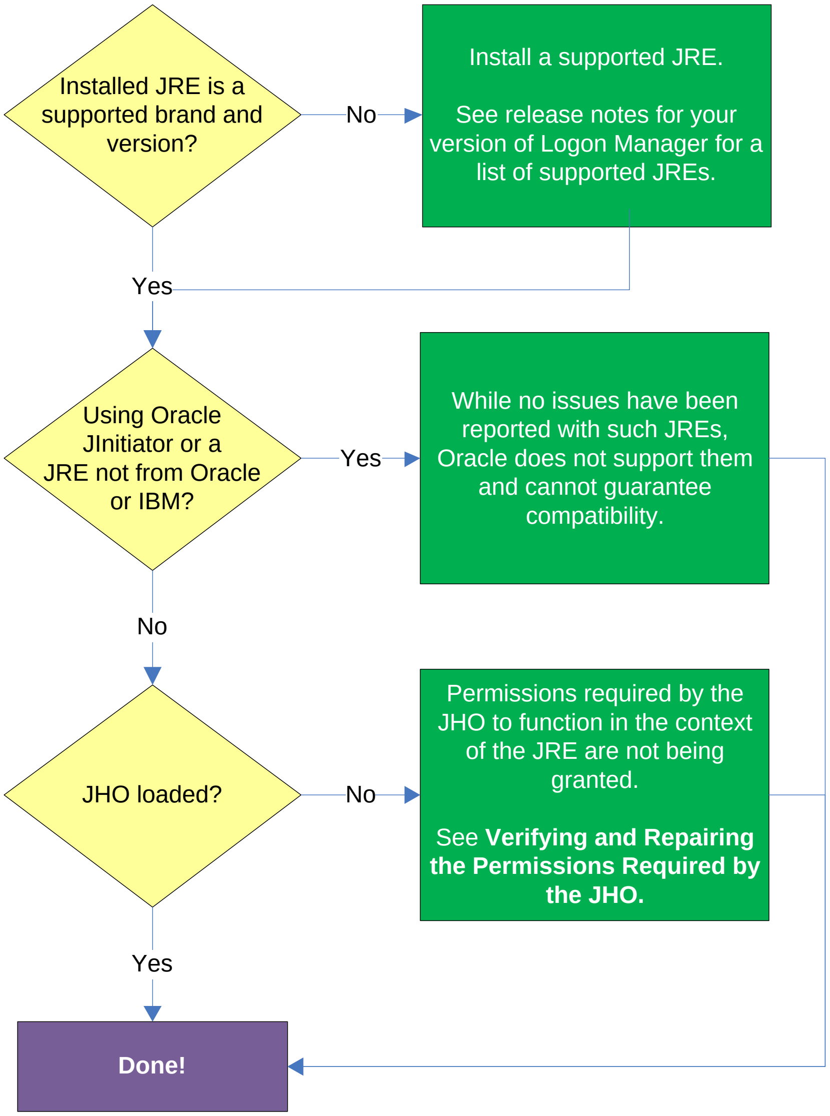
1. In the Logon Manager Administrative Console, open the desired template and select the **Miscellaneous** tab.
2. In the **Logon Loop Grace Period** field, select the desired mode of operation from the drop-down list:
   - **Prompt** – if the Agent detects the application’s logon form while the grace 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.

**Note:** If you have configured the logon grace period timer, and logon loop is still occurring for a specific form definition, make sure that the **Adhere to logon loop grace period** option in the form definition’s **Options** tab is enabled.

If this does not resolve the logon loop for the application, contact Oracle Support for 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 Logon Manager for a list of supported JREs.
Yes
Using Oracle JInitiator or a JRE not from Oracle or IBM?
Yes
While no issues have been reported with such JREs, Oracle does not support them and cannot guarantee compatibility.
No
JHO loaded?
No
Permissions required by the JHO to function in the context of the JRE are not being granted.
See **Verifying and Repairing the Permissions Required by the JHO.**
Yes
Done!

### Installed JRE is a supported brand and version?

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

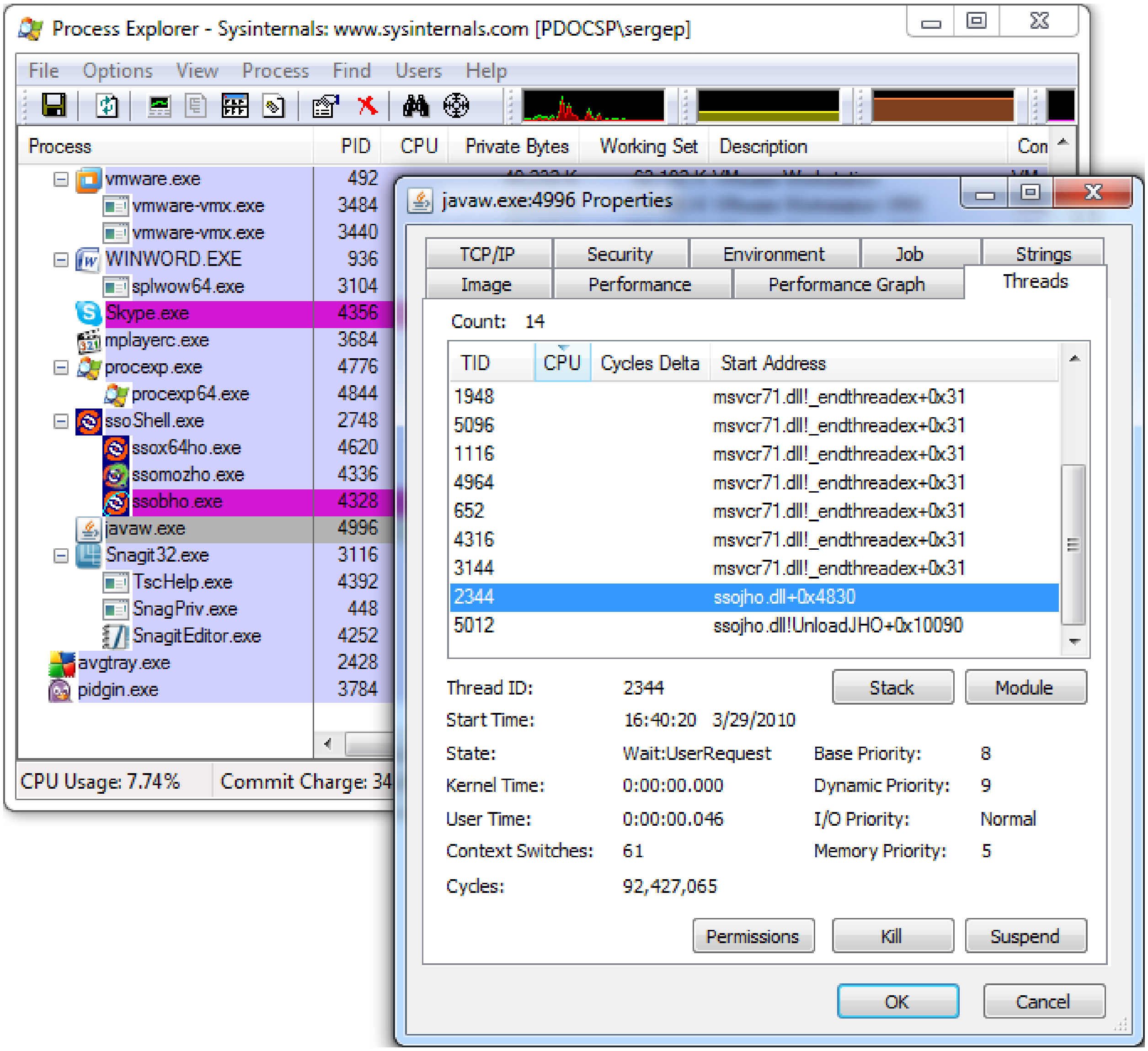
### Using Oracle JInitiator or another JRE not made by Oracle or IBM?

While no issues have been reported when deploying Logon Manager with Oracle JInitiator or non-Oracle/non-IBM JREs, Oracle does not support nor warrant the proper functioning of Logon Manager with such JREs.

## JHO loaded?

In certain situations, a configuration issue might prevent the JHO from loading when the Java application 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 | CPU | Private Bytes | Working Set | Description | Com |
|---|---|---|---|---|---|---|
| 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 the JHO is not loading for the target JRE, check that the permissions required by the JHO are being granted through the user’s `.java.policy` file (located in their home directory) as described in the next section.

## Verifying and Repairing the Permissions Required by the JHO

The JHO requires that a set of specific permissions is granted to it via the user's `.java.policy` file so that it can properly operate in the context of the target JRE. If these permissions are not in effect, Logon Manager will be unable to detect or respond to Java applications.

### Verifying That the Required Permissions Are Being Granted

Every time Logon Manager starts, the permissions are copied from a template `.java.policy` file (located in `%INSTALLDIR%\v-GO SSO\Helper\Java`) to the user's `.java.policy` file located in the user's home directory. These permissions are then automatically granted to the JHO every time a Java application is launched.

If you suspect that the required permissions are not being granted to the JHO, check your JRE's log after starting Logon Manager for any missing permission exceptions referencing the JHO. If such exceptions are present, the required permissions are not being granted. This can be caused by one of the following:

- The user's `.java.policy` file does not contain the permissions required by the JHO,
- The target JRE's security settings (usually stored in the JRE's `java.security` file) do not reference the user's `.java.policy` file.

### Restoring the Missing JHO Permissions

If the user's `.java.policy` file is missing the required JHO permissions, the file's ACL might be preventing Logon Manager from writing the permissions to it; in such case, check the file's ACL to make sure that write privileges are not being denied.

If the ACL is correct and the required permissions are not present in the user's `.java.policy` file, do one of the following:

- If the user's `.java.policy` file contains other permissions or configuration information that you want to preserve, edit the file and manually add the missing JHO permissions.
- If the user's `.java.policy` file does not contain other permissions or configuration information that you wish to preserve, simply delete it and restart Logon Manager; the file will be recreated and the required permissions inserted automatically.

**Note:** As JREs are updated by their vendors, it is possible that a future release introduces one or more new permissions that will be required by the JHO but which could not be accounted for at the time of Logon Manager's release. In such case, examine your JRE's error log for any missing permission exceptions after launching Logon Manager and a target Java application and add those permissions to Logon Manager's `.java.policy` template file so that they can be granted via the user's `.java.policy` file the next time Logon Manager starts.

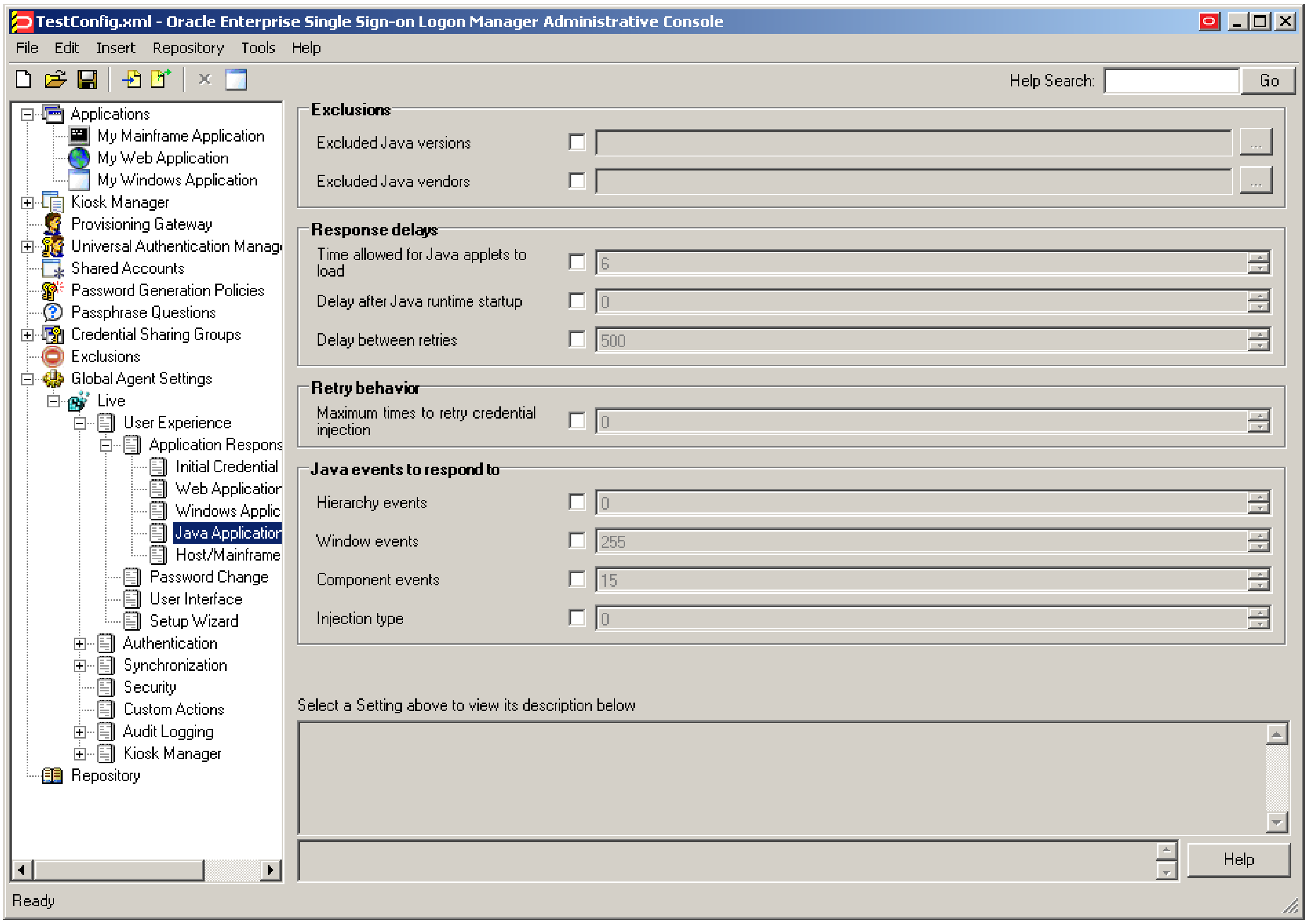
## Configuring the Behavior of the Java Helper Object (JHO)

Logon Manager provides global Agent settings that allow you to control the following aspects of the JHO's behavior:

- Exclude specific JRE versions
- Exclude specific JRE vendors
- Define a response delay to account  for the loading of Java applets
- Define  a response delay to account for the initialization of the JRE
- Define a delay between response retries
- Define a maximum number of response retries
- Define specific hierarchy, window, component and injection type events that the JHO will recognize or ignore

These settings are located in the tree in the Logon Manager Administrative Console at the following location:

**Global Agent Settings → Live → User Experience → Application Response → Java Applications**



**Exclusions**

Excluded Java versions □

Excluded Java vendors □

**Response delays**

Time allowed for Java applets to load □ 6

Delay after Java runtime startup □ 0

Delay between retries □ 500

**Retry behavior**

Maximum times to retry credential injection □ 0

**Java events to respond to**

Hierarchy events □ 0

Window events □ 255

Component events □ 15

Injection type □ 0

Select a Setting above to view its description below

To learn more about these settings and how to configure them, see the Console help. For even more granular control, Logon Manager also provides registry settings described in the next sections. Oracle recommends these settings be used only by experienced administrators.

### Manually Restricting the JHO to Specific JREs

Aside from the Console settings described earlier, Logon Manager provides registry settings that allow you to create extremely granular inclusion and exclusion rules that Logon Manager 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).<br>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:<br>`JhoIncludeHostName1=.*javaw?\.exe`<br>No-value default:<br>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.<br>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:<br>`JhoIncludeHostCommandLine1=.*Login\.jar.*`<br>No-value default:<br>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).<br>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:<br>`JhoExcludeHostName1=.*javaw?\.exe`<br>No-value default:<br>Blank (i.e., nothing will cause the JHO to ignore the application) |
| `JhoExcludeHostCommandLineN` | Specifies the command used to launch the application.<br>The command string usually includes the full path and name of the JVM executable, the JAR file, and any required command-line parameters.<br>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:<br>`JhoExcludeHostCommandLine1=.*Login\.jar.*`<br>No-value default:<br>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.<br>Default value:<br>`1.2` (the earliest JRE version supported by the JHO) |
| `JhoMaximumJavaVersion` | Specifies the highest allowed JRE/JDK version.<br>Default value:<br>Blank (i.e., no upper JRE version limit is imposed) |

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

Aside from the settings provided in the Console, Logon Manager provides registry settings that control the event response behavior of the JHO at an even more granular level. You can use these settings to troubleshoot and optimize Logon Manager performance when responding to Java applications by modifying the JHO'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:<br>`HIERARCHY_EVENT_CHANGED = 0x1`<br>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:<br>• `WINDOW_EVENT_OPENED = 0x1`<br>• `WINDOW_EVENT_CLOSED = 0x2`<br>• `WINDOW_EVENT_ACTIVATED = 0x4`<br>• `WINDOW_EVENT_DEACTIVATED = 0x8`<br>• `WINDOW_EVENT_CLOSING = 0x10`<br>• `WINDOW_EVENT_ICONIFIED = 0x20`<br>• `WINDOW_EVENT_DEICONIFIED = 0x40`<br>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:<br>• `COMPONENT_EVENT_SHOWN = 0x1`<br>• `COMPONENT_EVENT_HIDDEN = 0x2`<br>• `COMPONENT_EVENT_ADDED = 0x4`<br>• `COMPONENT_EVENT_REMOVED = 0x8`<br>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:<br>• `INJECT_TYPE_DEFAULT = 0`<br>• `INJECT_TYPE_METHOD = 1`<br>• `INJECT_TYPE_ACCESSIBLE = 2`<br>• `INJECT_TYPE_NONACCESSIBLE = 3`<br>• `INJECT_TYPE_ROBOT = 4`<br>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:<br>• `INJECT_TYPE_METHOD` (if an appropriate set method had been found for the control)<br>• `INJECT_TYPE_ACCESSIBLE` (if the control supports accessibility)<br>• `INJECT_TYPE_NONACCESSIBLE`<br>• `INJECT_TYPE_ROBOT`<br>**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 Logon Manager:

- `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)`

# **Part 2:** Web Applications

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 chapter, as well as the recommended best-practice procedures for configuring and testing Windows application forms, and diagnosis and resolution steps for the most common issues that may cause the Agent to erratically detect and/or respond to Web forms. This part covers the following topics:

- Overview of a Sign-On Event
- Understanding Web Form Detection
- Understanding Form Response
- Determining the Optimal Configuration for a Form
- Configuring a Form
- Testing the Configuration of a Form
- Publishing a Template to the Repository
- Troubleshooting Web Form Detection
- Troubleshooting Form Response when Using Control IDs
- Troubleshooting Form Response When Using "SendKeys"
- Troubleshooting Detection Matching
- Troubleshooting a Logon Loop
- Troubleshooting Java Application Issues

**Tip:** If the steps in this chapter do not resolve your issue, you can troubleshoot further by tracing and logging the activity of Logon Manager 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 "Using the Trace Controller Utility" topic in the *Enterprise Single Sign-On Suite Plus Administrator's Guide*.

## Overview of a Sign-On Event

Logon Manager 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, Logon Manager monitors the running Web browser(s) and detects whenever a Web page has completed loading in the browser.

For each completely loaded page, Logon Manager does the following:

1. **Detects the target Web page and form.** Whenever a new URL is accessed through the browser being monitored, Logon Manager:
    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 logon or 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), Logon Manager uses helper objects that hook into the browser(s) and provide Logon Manager 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: `SSOWEBHO.EXE`

Whenever the Web browser completes loading a page, the respective helper object notifies Logon Manager 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, Logon Manager begins detection.
- If the current page is refreshed, or if another page begins loading, Logon Manager 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, Logon Manager 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, Logon Manager 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 Logon Manager 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, Logon Manager supports the following types of forms in Web applications:

- Logon
- Logon success (a message confirming successful logon)
- Logon failure (a message indicating the injected credentials have been rejected)
- 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 Logon Manager to respond.

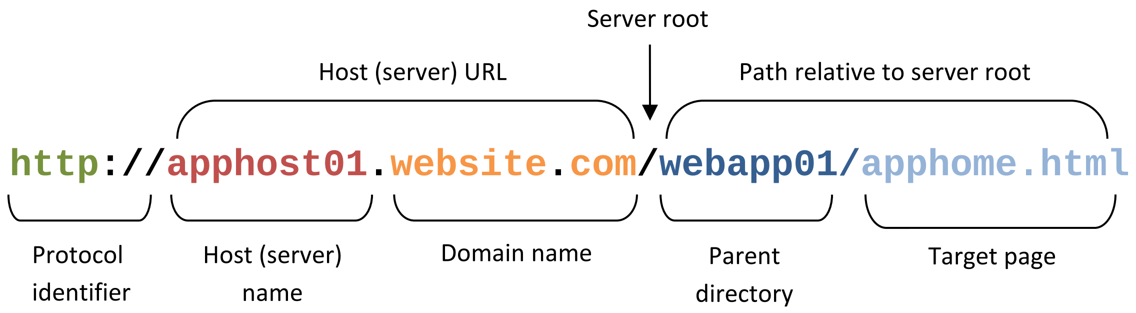
> **Note:** Defining a form comprises providing unique identification criteria, specifying the action to take when the form is detected, and the specific way in which the action (e.g., injecting credentials) should be performed.

Logon Manager 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 Logon Manager 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

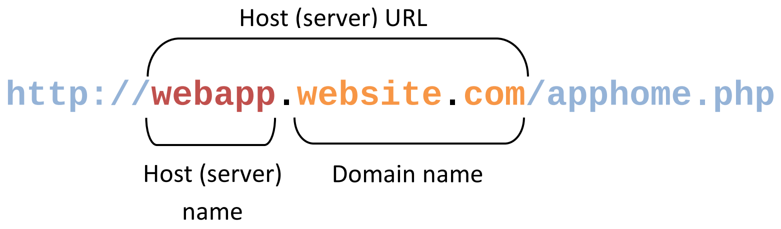
`…apphome.html?s=bd4cecd0d934870&view=full&sidebar=off`

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. Logon Manager allows for matching against dynamic URLs via regular expressions, as well as a global Agent setting that determines the URL matching precision used by Logon Manager 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



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:

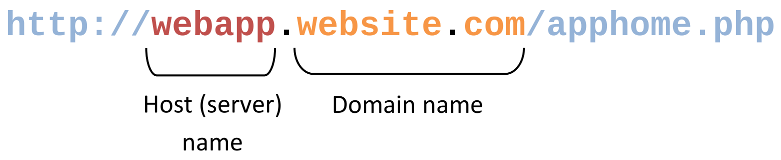
`http://*.website.com/apphome.php`

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 Logon Manager will match against three consecutive segments of the host URL – in our example, it will match the entire host name exactly as it appears:



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 Logon Manager 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, Logon Manager 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 Logon Manager 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 Logon Manager'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.

Logon Manager 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 Logon Manager 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

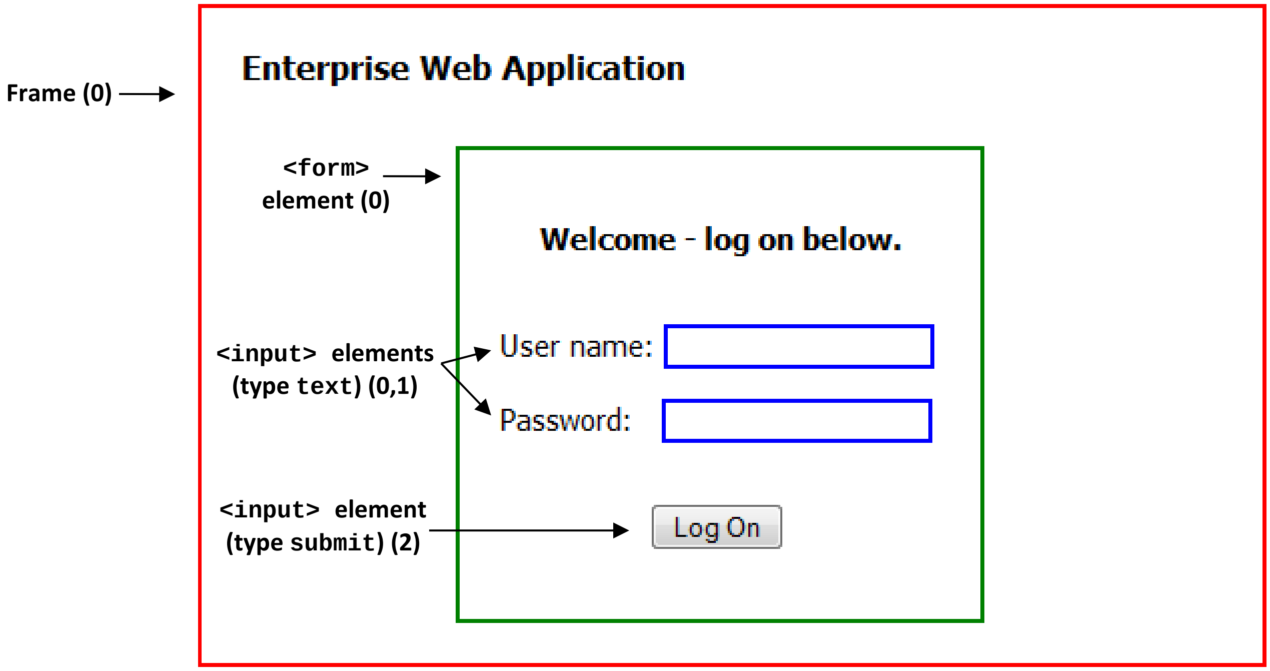
Once Logon Manager has positively matched the detected URL to a template, it examines the page's Document Object Model (DOM) in order to detect the fields configured in the form definition.

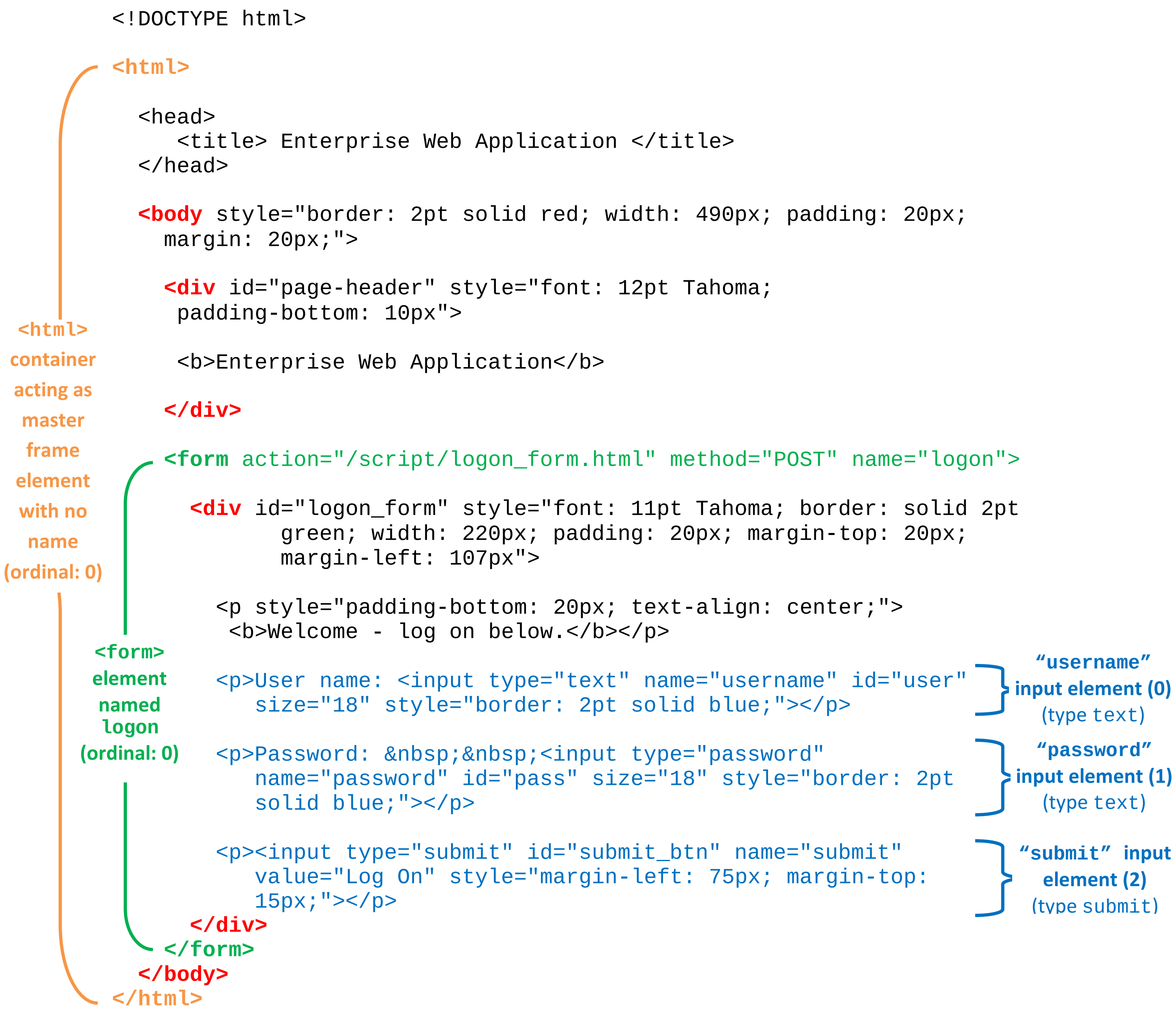
For each field and control, the Agent looks at the name or ordinal of:

- 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, Logon Manager 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>
```

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 Logon Manager by:

- Value of the element's name attribute, or, if not specified,
- Ordinal, an index number assigned by Logon Manager 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 Logon Manager 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 Logon Manager

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, Logon Manager recognizes the following element types:

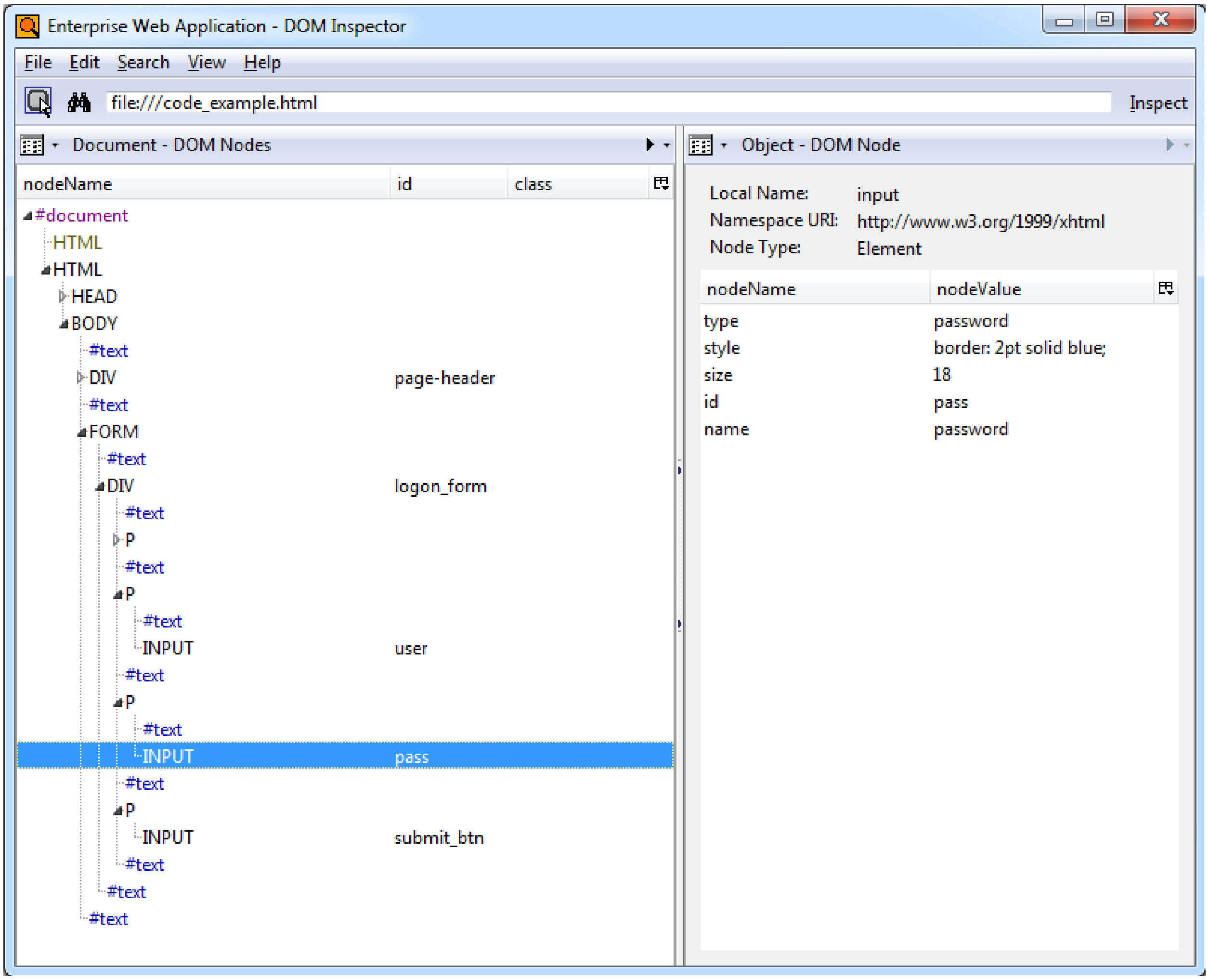
| Logon Manager Field Type | Input Element Types Accepted by the Logon Manager Field Type |
|---|---|
| • **User name/ID**<br>• **Password**<br>• **Third Field**<br>• **Fourth Field** | • **Text** – a text field identified by the `<input>` tag of `type="text"`. Text entered into this field is unmasked.<br>• **Password** – a text field identified by the `<input>` tag of `type="password"`. Text entered into this field is masked.<br>• **Select-one** – a list field identified by the `<select>` tag that permits the selection of a single list item as its value.<br>• **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.)<br>• **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.<br>• **Image** –a hyperlinked image identified by the `<input>` tag of `type="image"` that executes the referenced link when clicked.<br>• **IMG** –an image identified by the `<img src=` tag and wrapped in either an anchor (<A HREF=) tag or JavaScript `onClick` action which executes the submit URL when clicked.<br>• **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

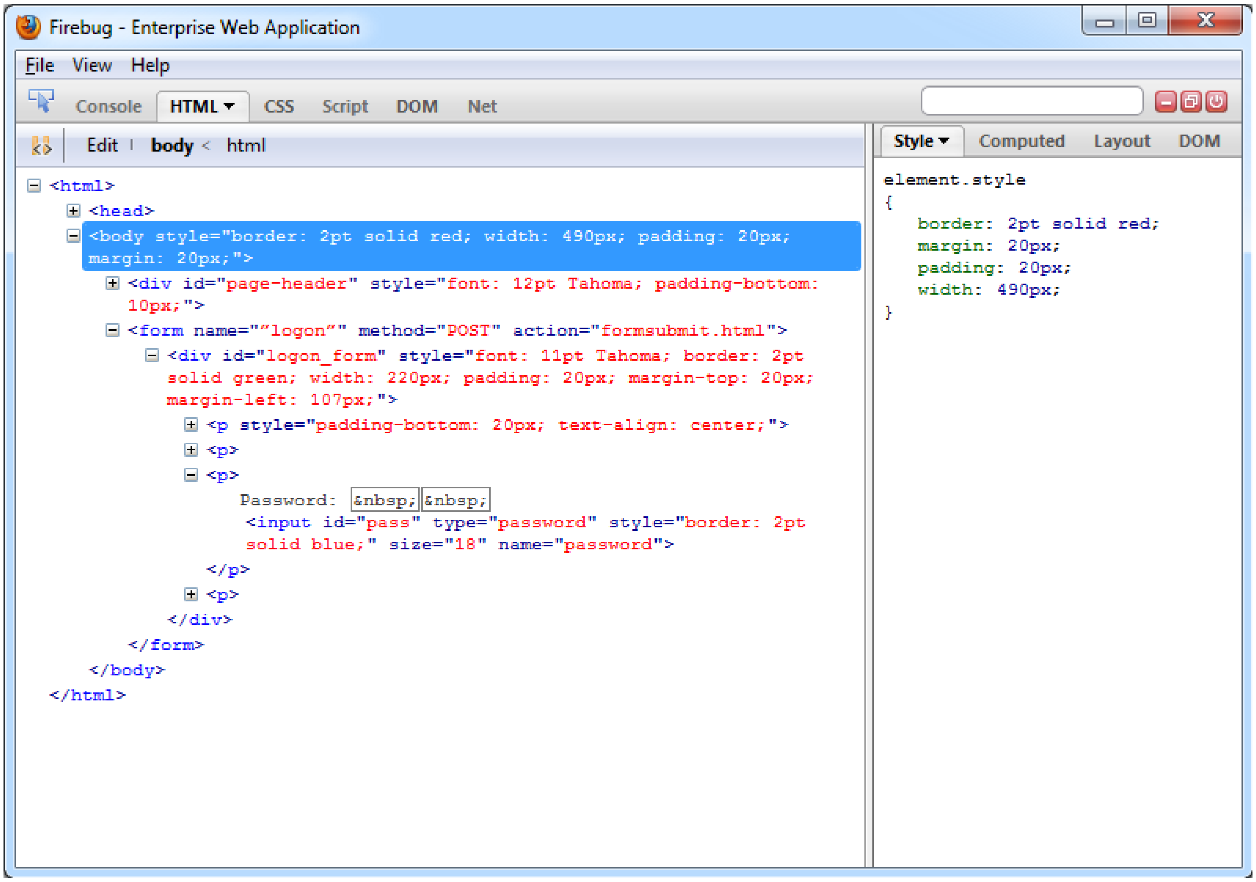
| nodeName | nodeValue |
|---|---|
| type | password |
| style | border: 2pt solid blue; |
| size | 18 |
| id | pass |
| name | password |

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 Logon Manager 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 Logon Manager 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.

Logon Manager 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

Logon Manager 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, Logon Manager 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, Logon Manager 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 Logon Manager 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

Logon Manager 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 Logon Manager to ignore the form once the success message is displayed so that Logon Manager 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 Logon Manager to ignore the form, and
- An **AND** match rule for any text on the form that will result in a positive match and cause Logon Manager 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, Logon Manager 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 ATL framework to formulate the text pattern that Logon Manager 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:

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

<table>
<tr><th>Expression</th><th>Description</th></tr>
<tr><td>+</td><td>Indicates that the preceding expression matches one or more times.<br><br>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.<br><br>Example: Given the input <abc><def>, <.*?> matches <abc><br>while <.*> matches <abc><def>.</td></tr>
<tr><td>\</td><td>Escape character that forces the next character to be interpreted literally.<br><br>Example: [0-9]+ matches one or more digits, but [0-9]\+ matches a digit followed by a plus character).<br><br>If \ is followed by a number *n*, it matches the *n*th match group (starting from 0).<br><br>Example: <{.*?}>.*?</\0> matches <head>Contents</head>.<br><br>The **** is also used for abbreviations, the most common of which are shown below:<br><br>
<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.<br><br>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 Logon Manager has successfully identified the Web form and its input fields, it uses the appropriate helper object to populate the target fields and actuate the submit control in the form. Depending on how the Web page is coded, you will want to use one of the response methods described below.

### Control IDs

This is the default and preferred form response method for most Web applications. This method allows Logon Manager to use the browser's API to programmatically inject the appropriate credentials into

the target fields, and engage the submit control.

However, if a Web page is written in a way that hampers or even prohibits Logon Manager's ability to programmatically interact with the form's fields and controls, and/or if there are additional actions required to complete the sign-on event, such as selecting a check box or manually setting focus on a specific field, you may also have to use the "SendKeys" method, in tandem with the "Control IDs" method, to interact with the form.

### "SendKeys"

This method allows the Agent to interact with the target form by emulating user input, such as keystrokes and mouse clicks. Use this method if the Agent is unable to programmatically interact with the form's fields and controls. For example, if the logon form is implemented using Adobe Flash, you will need to send mouse-clicks to select the target fields, send keystrokes to populate them, then send a final mouse-click to engage the submit control.

### Control IDs with "SendKeys"

This method is a "best-of-both-worlds" combination of the two above methods and is the preferred way of solving sign-on scenarios that require actions that cannot be performed programmatically. Control IDs are used to interact with the form wherever possible, while keystrokes and mouse clicks are sent to accomplish tasks such as setting field focus, selecting a check box, and other actions that the Agent cannot perform programmatically within the target application.

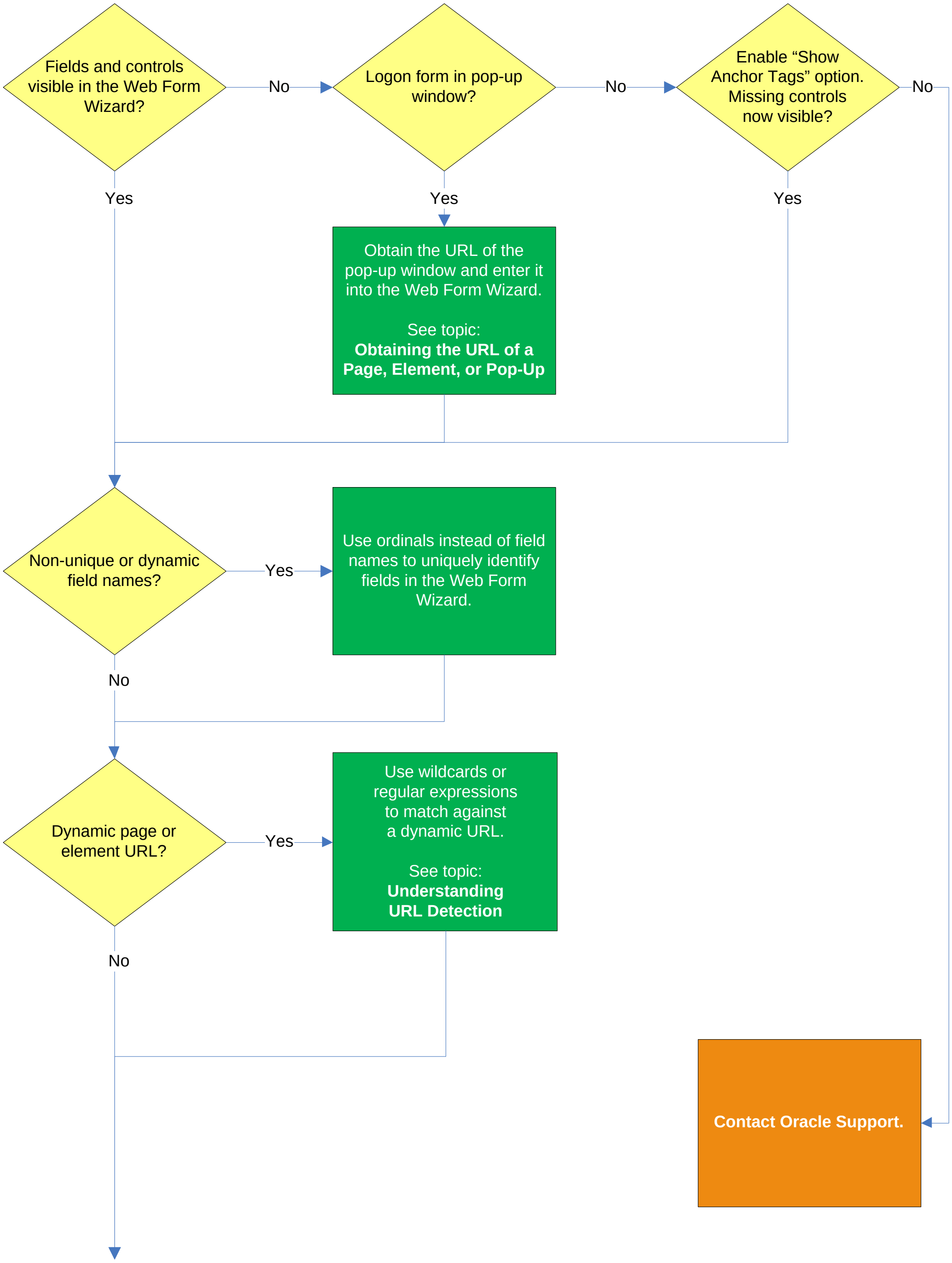
For example, if fields must be populated in a specific order due to cascading validation (i.e., the password field becomes active only once the user name field has been populated), you would use the "Control IDs to inject credentials into the fields, but send a **Tab** keystroke via "SendKeys" between each field injection to advance field focus.

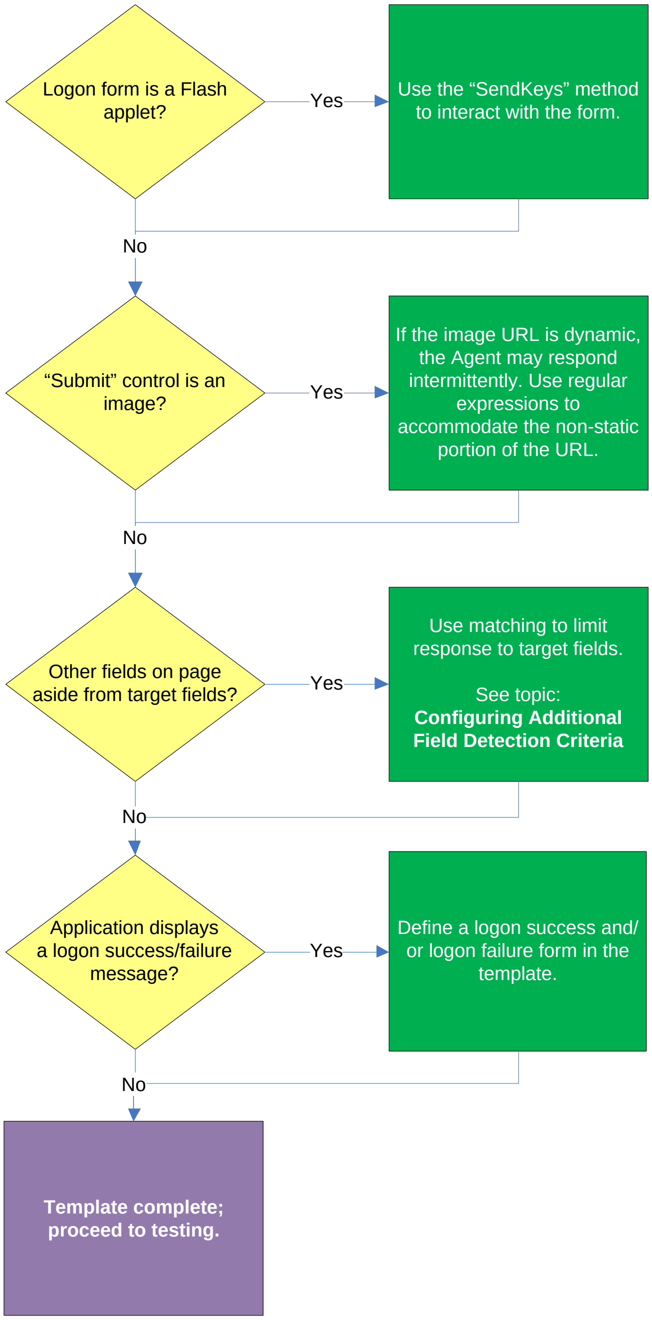
**Note:** To achieve this "mixed" mode, configure the form to initially use the "SendKeys" response method , then configure the desired "SendKeys" actions; while configuring the actions, enable the **Inject directly into control** option for actions that you wish to retain the "Control IDs" programmatic response method.

# 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
**Contact Oracle Support.**


Logon form is a Flash applet?
Yes
Use the “SendKeys” method to interact with the form.
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
Application displays a logon success/failure message?
Yes
Define a logon success and/ or logon failure form in the template.
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 Logon Manager. 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.

### Logon form is a Flash applet?

If the logon form has been implemented in Adobe Flash, Logon Manager will not be able to programmatically interact with its fields and controls; in such cases, use the "SendKeys" method to interact with the form.

### "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, Logon Manager 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.

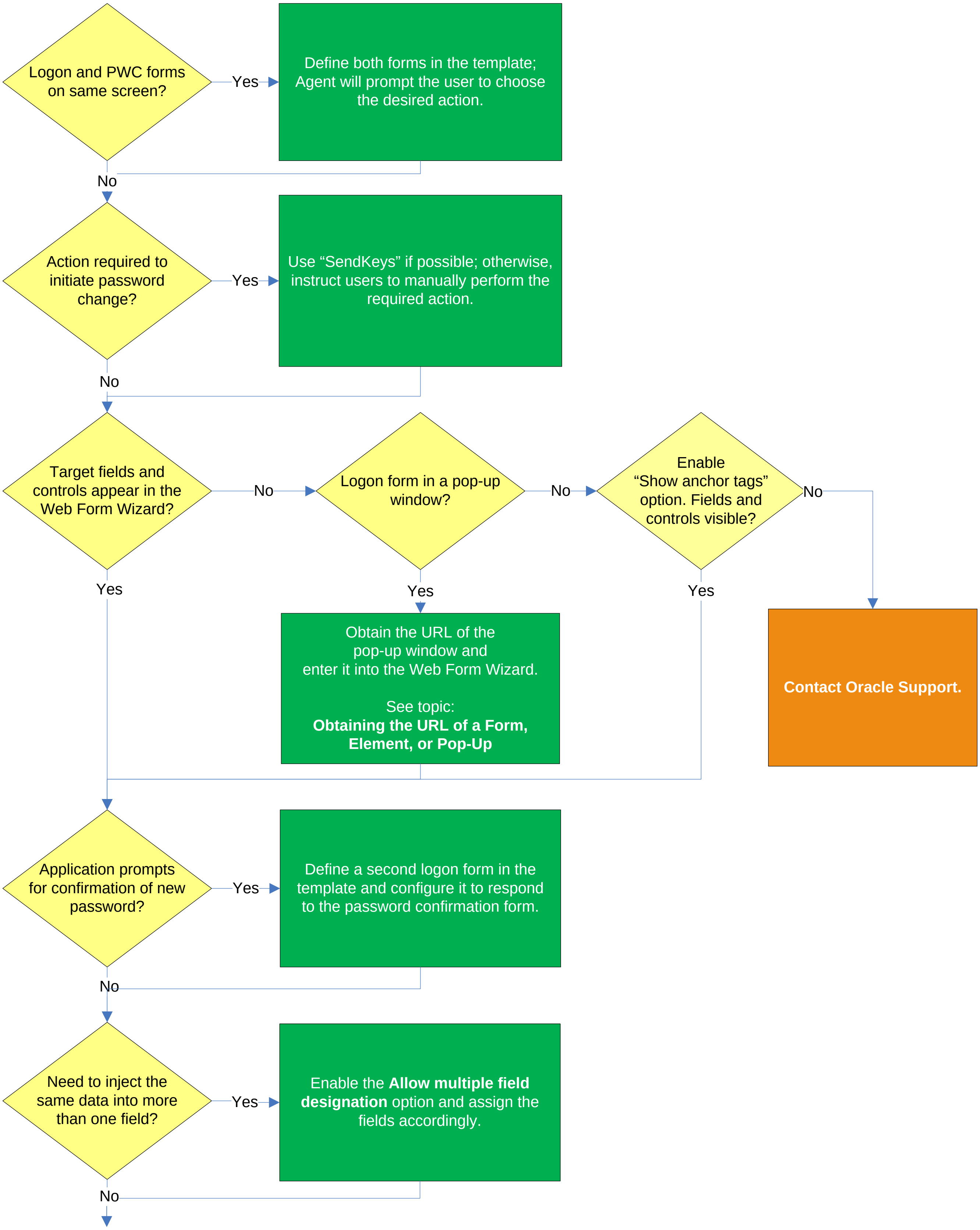
### Other fields on the page besides target fields?

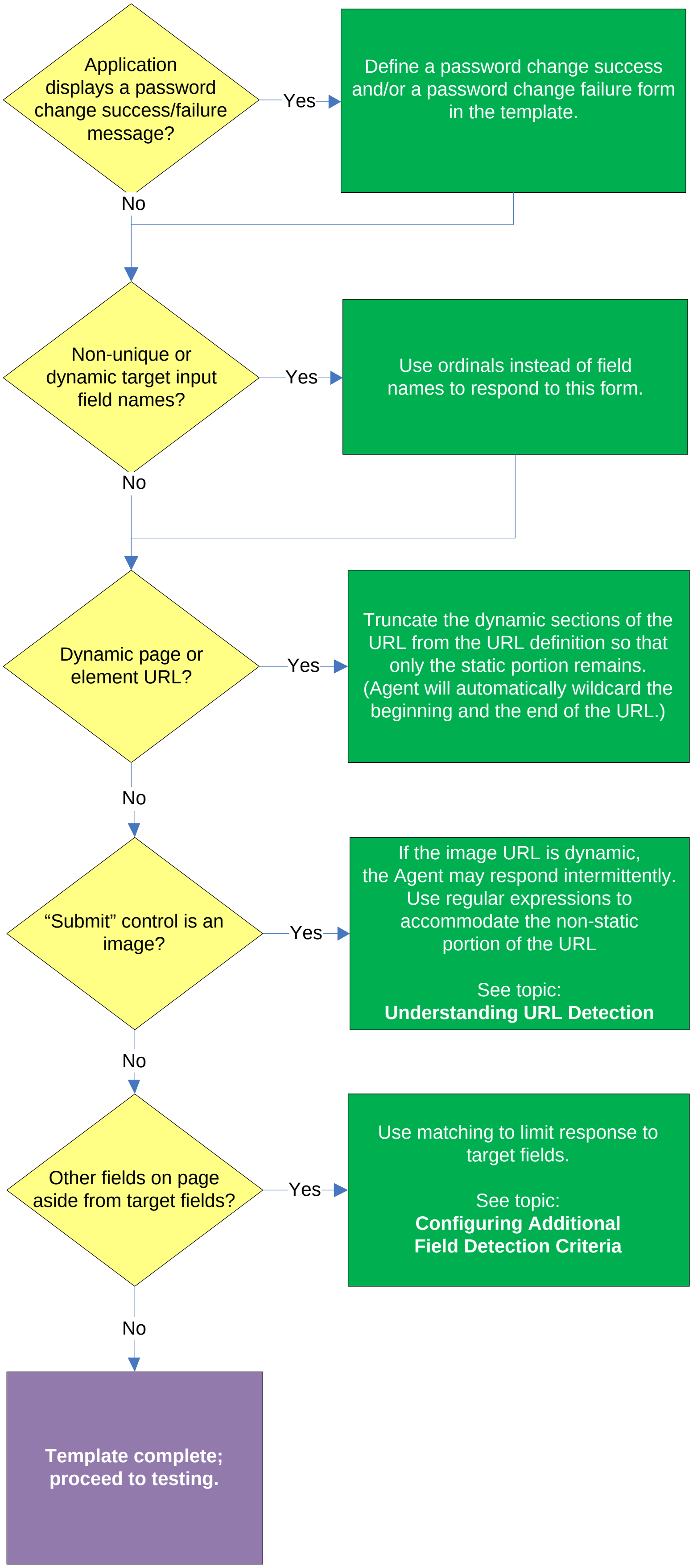
If there are other input fields and controls on the page aside from the target fields, use matching to limit Agent response to the desired fields. For more information, see Configuring Additional Field Detection Criteria.

### Application displays a logon success and/or failure message?

Some applications display a message indicating whether logon was a success or a failure.
In such cases, define a logon success and/or a logon failure form in the application template.

## 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
Use "SendKeys" if possible; otherwise, 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
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**
**Contact Oracle Support.**
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
Need to inject the same data into more than one field?
Yes
Enable the **Allow multiple field designation** option and assign the fields accordingly.
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 dialog.

### 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, Logon Manager may not be able to complete that action programmatically. In such cases,, use the "SendKeys" method to interact with the target control if possible; otherwise, instruct users to perform the require action manually.

### 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.

### Need to inject the same data into more than one field?

If you need to inject the same data into more than one field, .e.g., the form contains a "Password" and a "Confirm Password" field and you want the Agent to inject the new password into both of them, enable the **Allow Multiple Field Designation** option in the Web Form Wizard and assign the fields accordingly.

### Application displays a password change success and/or failure message?

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 Logon Manager. 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, Logon Manager 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.

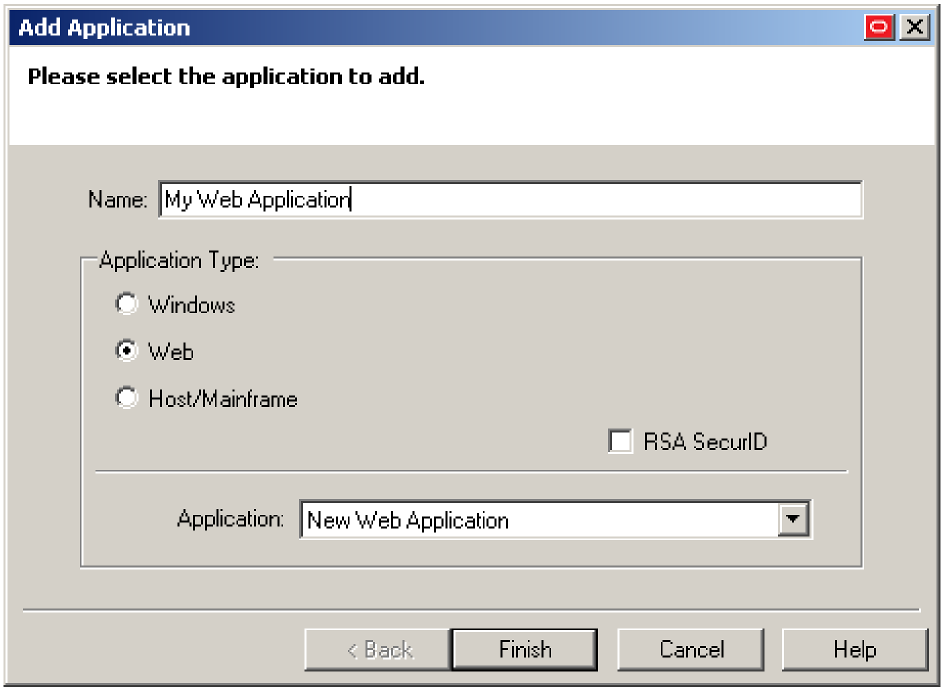
## Basic Configuration

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

**Note:** You can begin the configuration process more quickly by clicking the Logon Manager button
in the target application's title bar (if enabled) and selecting Create Template from the context menu that appears.

1. Open the Logon Manager Administrative Console. By default, the shortcut is located in **Start → Programs → Oracle → Logon Manager 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:

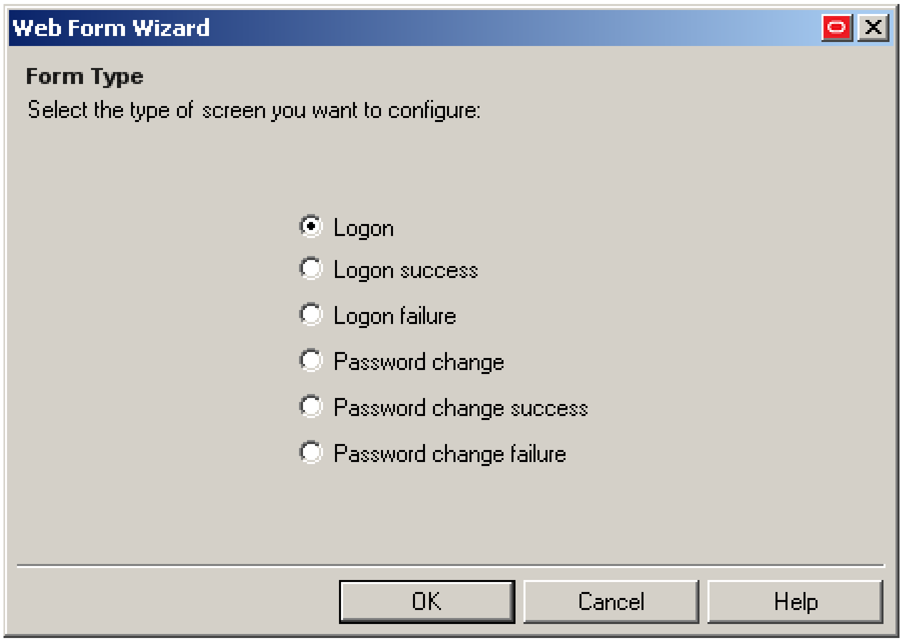
○ Windows
● Web
○ Host/Mainframe

□ 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**.

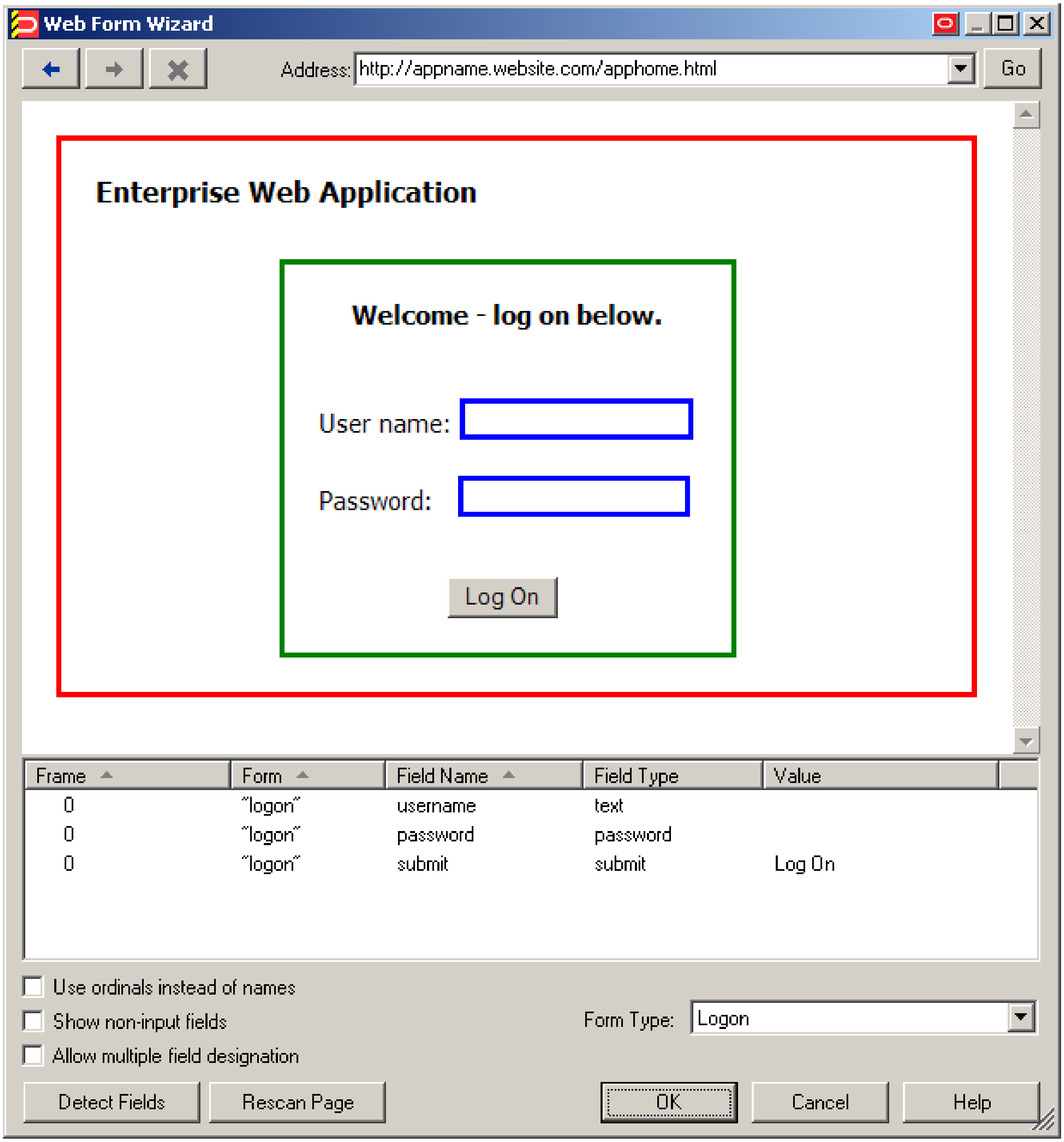


**Form Type**

Select the type of screen you want to configure:

● Logon
○ Logon success
○ Logon failure
○ 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.



Address: http://appname.website.com/apphome.html

| 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 non-input fields

□ Allow multiple field designation

Form Type: Logon

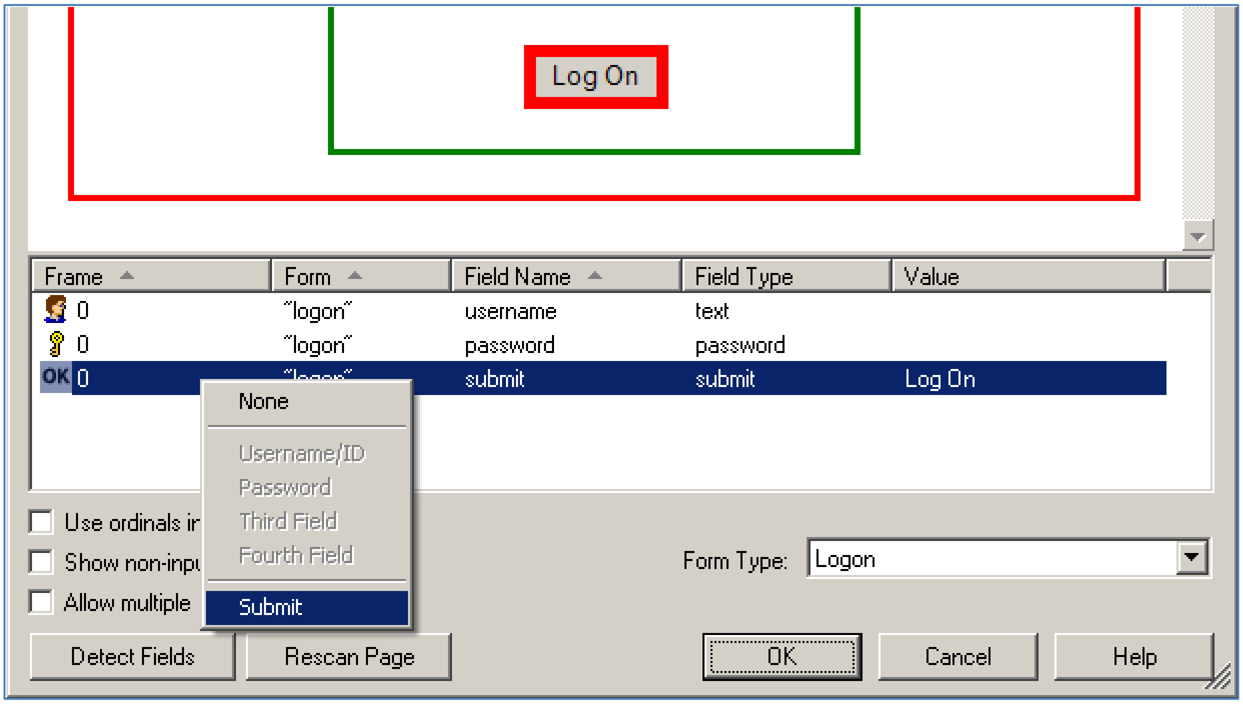
**Note:** If you see two or more fields that share the same name, the Agent might respond to the form erroneously when configuration is complete. In such cases enable the Use ordinals instead of names option to uniquely identify the input fields to the Agent via ordinals (consecutive index numbers). 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.
>
> **Note:** By default, all fields and controls use the "Control IDs" response method. If you want to use the "SendKeys" response method for one or more fields, define them as described in this procedure, then follow the instructions in Configuring a Field or Control Using the "SendKeys" Method.

i. (Optional) If your form requires the injection of the same data into multiple fields (e.g., injecting the password into a **Password** and a **Confirm Password** field), enable the **Allow multiple field designation** option.

ii. 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 non-inpu

□ Allow multiple

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, Logon Manager will still send a "submit" action to the application after injecting credentials. This action is equivalent to the user manually pressing the **Enter** key.

iii. If the submit control is missing, it might be implemented as an anchor (i.e., `<A HREF=`) or an image (i.e., `<IMG SRC=`). In such cases, enable the **Show anchor tags** and/or **Show non-input fields** option, respectively, to expose such a control.

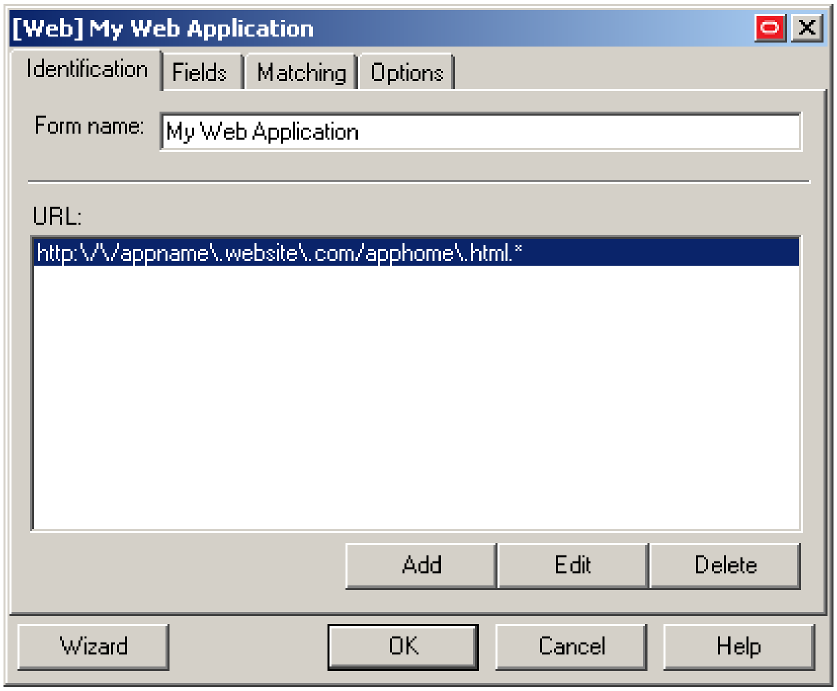
iv. When you have configured all of the visible input fields, click **OK**.

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.

> **Warning:** Oracle highly recommends removing the default wildcard regular expression prefix of **.*?** from your URL definitions to prevent potential phishing attacks.

- If you need to configure additional field detection criteria via the **Matching** tab, continue to Configuring Additional Field Detection Criteria.



Form name: My Web Application

URL:

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

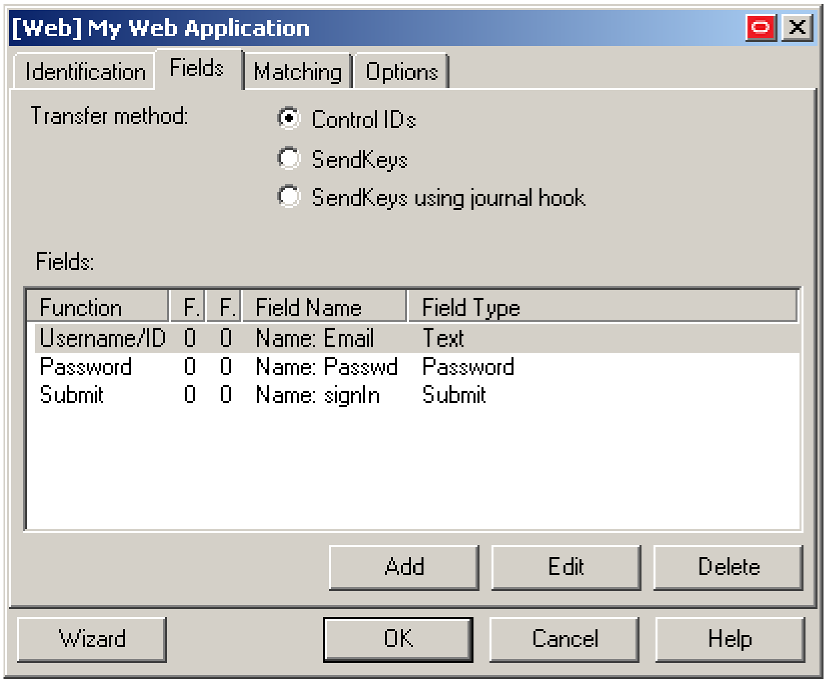
> **Note:** If the contents of the page are dynamic, i.e., change after the page has completed loading, select the **Dynamic Page** option in the **Options** tab to enable active page polling. This will allow the Agent to monitor the page for changes.

## Manually Configuring a Field or Control

If you were unable to configure an input field using the Web Form Wizard, or if you want to change a field's existing configuration, follow the procedures below.

### Configuring a Field or Control Using the "Control IDs" Method

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**.
3. In the form properties dialog appears, select the Fields tab.



[Web] My Web Application

Transfer method:

● Control IDs
○ SendKeys
○ SendKeys using journal hook

Fields:

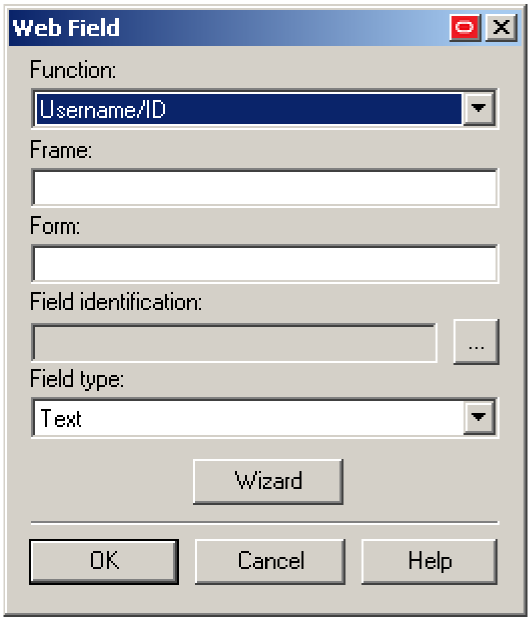
| Function | F. | F. | Field Name | Field Type |
|---|---|---|---|---|
| Username/ID | 0 | 0 | Name: Email | Text |
| Password | 0 | 0 | Name: Passwd | Password |
| Submit | 0 | 0 | Name: signIn | Submit |

4. In the **Fields** tab, 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 "Web Field" 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 Type** drop-down list, select the type of the target input field.



Web Field

Function: Username/ID

Frame:

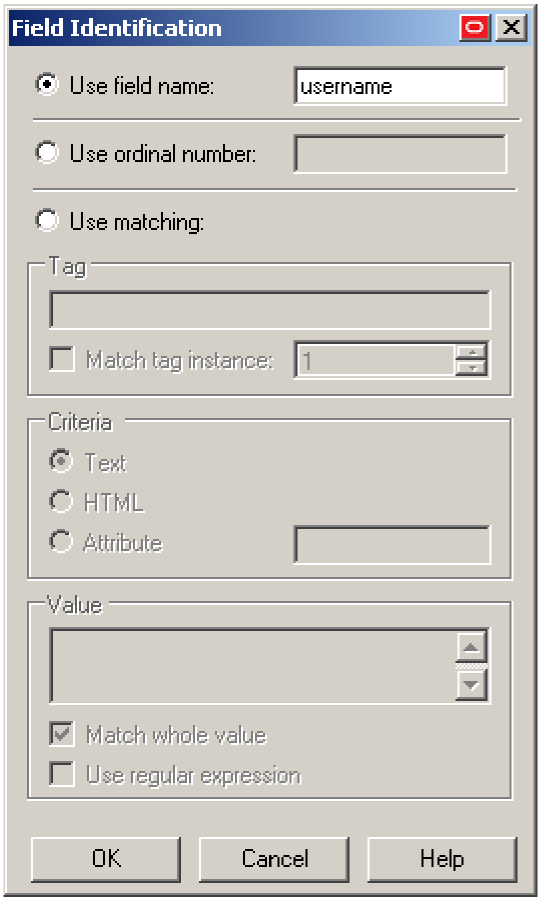
Form:

Field identification:

Field type: Text

v. In the **Field Identification** field, click the **…** (ellipsis) button and do the following:

a) Select the method that you want Logon Manager to identify the field. The available methods are field name, ordinal number, and matching.



Field Identification

● Use field name: username

○ Use ordinal number:

○ Use matching:

Tag

□ Match tag instance: 1

Criteria

● Text

○ HTML

○ Attribute

Value

☑ Match whole value

□ Use regular expression

b) Enter the appropriate value(s) for the selected identification method.

> **Note:** For more information on configuring matching, see Understanding Detection Matching.

c) Click **OK** to save your changes and dismiss the dialog.

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 a Field or Control Using the "SendKeys" Method

Logon Manager does not allow the direct addition of "SendKeys" actions after the form's response method has been changed to "SendKeys." To add a new action to a form definition that has been switched to the "SendKeys" response method, you must first switch the form back to the "Control IDs" method, add the desired action as described in Configuring a Field or Control Using the "Control IDs" Method, then switch the form definition back to the "SendKeys" method. This procedure explains how to convert existing "Control IDs" actions to "SendKeys" actions and how to configure the converted "SendKeys" actions.

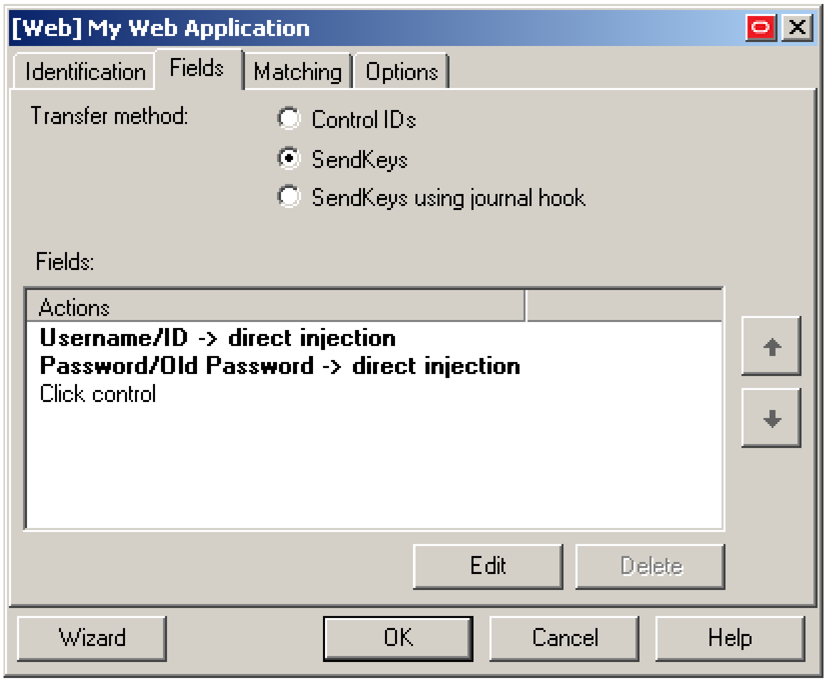
> **Note:** When you switch a "SendKeys" form definition back to "Control IDs," all "SendKeys" actions whose **Inject directly into control** option has been disabled will be deleted. Oracle highly recommends planning your form configuration thoroughly *before* you begin the configuration procedure.

To configure your "SendKeys" actions, 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**.
3. In the form properties dialog that appears, select the **Fields** tab and do the following:
   a. (Optional) If this is the first time you are switching to the "SendKeys" method, select **SendKeys** in the "Transfer Method" selector.

> **Note:** When you select the "SendKeys" method,  all actions in the "Fields" list are converted to "SendKeys" actions with the **Inject directly into control** option (found in the action's property dialog) enabled. In other words, those actions remain programmatic until you disable the **Inject directly into control** option for each action that you want to be executed using the "SendKeys" method.
>
> Oracle recommends using programmatic injection whenever possible and using "SendKeys' only when programmatic interaction with a field or control is not possible.
>
> **Note:** To learn about when to use the **SendKeys using journal hook** option, see Troubleshooting Form Response When Using "SendKeys."

The **Fields** tab display changes to reflect the “SendKeys” response method:



[Web] My Web Application

Transfer method:

○ Control IDs
● SendKeys
○ SendKeys using journal hook

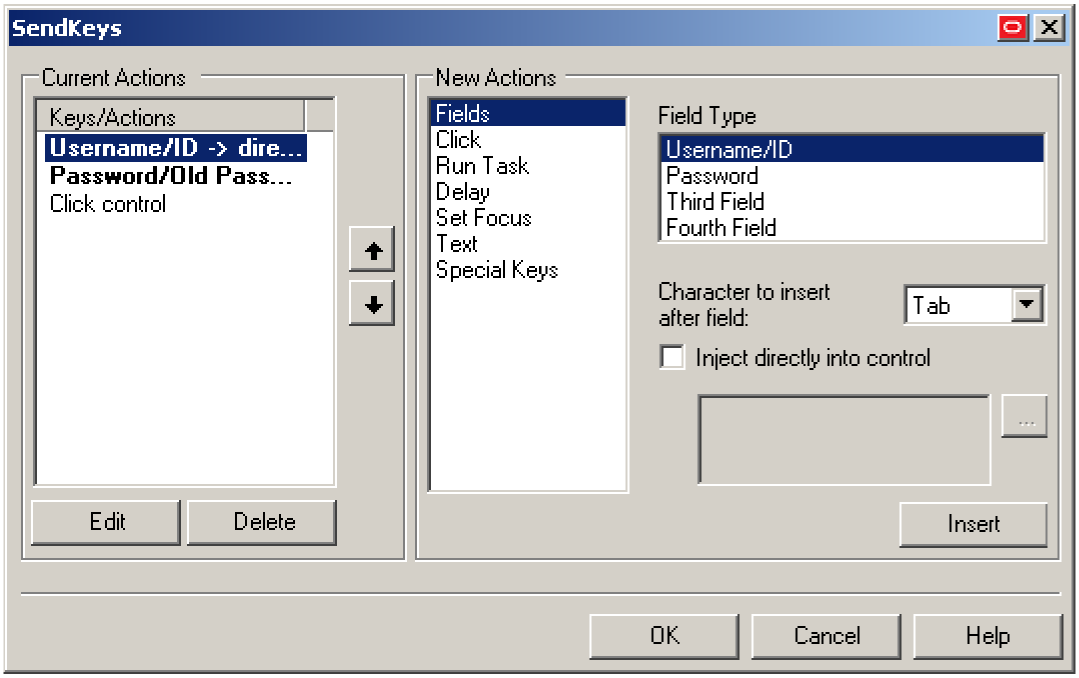
Fields:

Actions

**Username/ID -> direct injection**
**Password/Old Password -> direct injection**
Click control

b. To change the order in which the “SendKeys” actions are executed, select the action whose position you want to change, and use the up and down arrow buttons to move it.

c. To configure a “SendKeys” action, select it from the list, click **Edit**, and use the “SendKeys” dialog that appears to configure the action. Consult the Console help for an explanation of each option present in the dialog.



SendKeys

Current Actions

Keys/Actions

**Username/ID -> dire...**
**Password/Old Pass...**
Click control

New Actions

Fields
Click
Run Task
Delay
Set Focus
Text
Special Keys

Field Type

Username/ID
Password
Third Field
Fourth Field

Character to insert after field: Tab

□ Inject directly into control

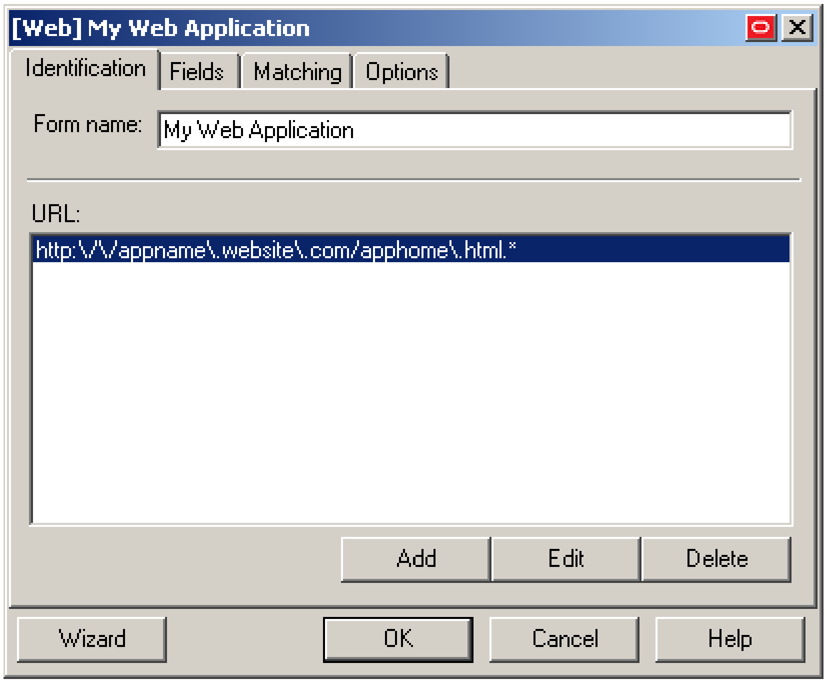
   d. When you have configured all of your actions, click **OK** to save your changes and dismiss the “SendKeys” dialog.
6. Click **OK** in the form properties dialog to dismiss it.
7. Publish your changes to the repository, as described in Publishing a Template to the Repository, if applicable.

## 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:

> **Warning:** Oracle highly recommends removing the default wildcard regular expression prefix of **.*?** from your URL definitions to prevent potential phishing attacks.

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**.
3. In the form properties dialog that appears, select the **Identification** tab.



Form name: My Web Application

URL:

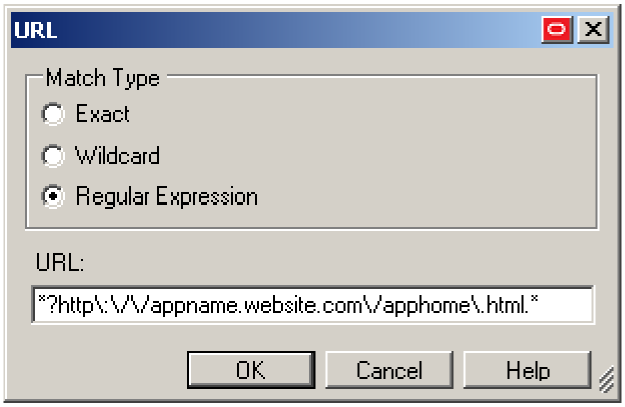
http:\/\/appname\.website\.com/apphome\.html.*

4. In the Identification tab, 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.

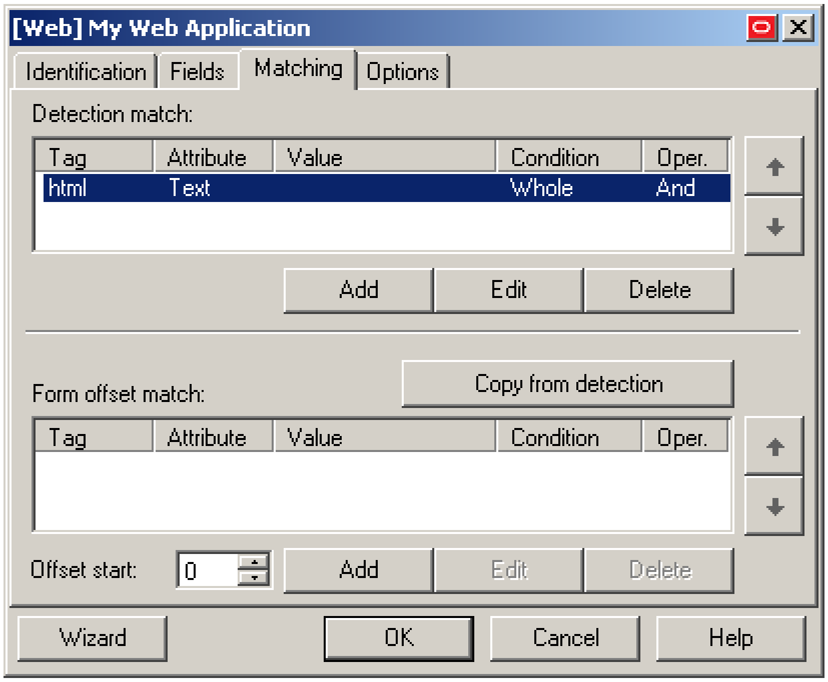
d. Click **OK** in the form properties dialog to dismiss it.

5. 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 | | Whole | And |

Form offset match:

| Tag | Attribute | Value | Condition | Oper. |
|---|---|---|---|---|

Offset start: 0

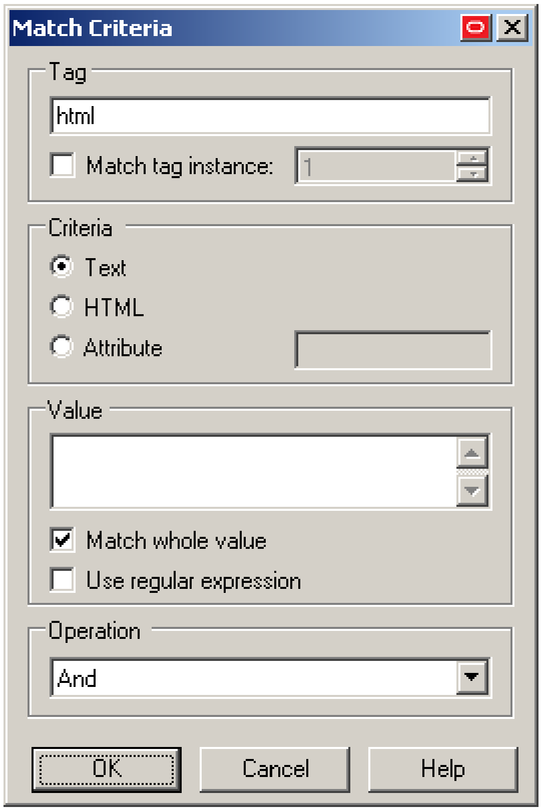
4. Complete the steps below in either the **Detection Match** or the **Form Offset Match** sections, depending on the desired matching type (see Detection Matching vs. Offset Matching for more information).

> **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 Logon Manager to match strictly against the criterion value. If you disable this option, Logon Manager 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 from the **Operation** drop-down list to indicate how this rule will chain with the preceding rule.



Match Criteria

Tag

html

☐ Match tag instance: 1

Criteria

● Text

○ HTML

○ Attribute

Value

☑ 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.

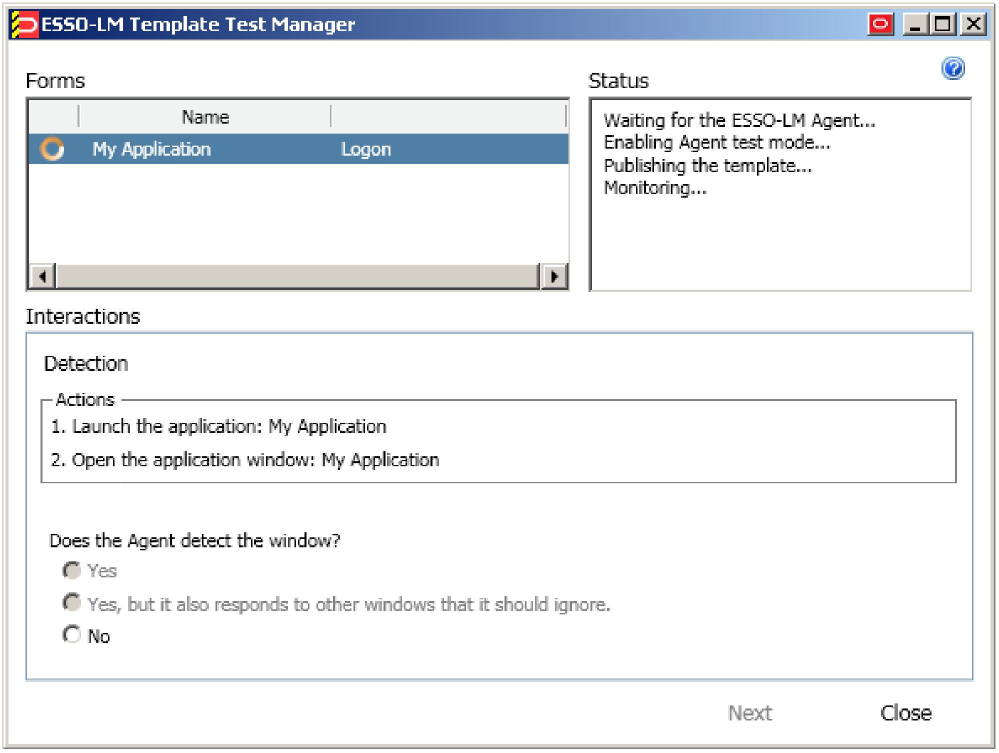
## Testing the Configuration of a Form

Once you have created and configured a form, Oracle highly recommends that you test your configuration before deploying it to end-user workstations. To do so, use the Template Test Manager feature found in the Logon Manager Administrative Console.

> **Caution:** The Template Test Manager provides interactive troubleshooting and remediation instructions for the most common configuration problems; it is not, however, designed to cover every possible problem scenario. For thorough troubleshooting, follow the instructions in this guide *in conjunction* with the Template Test Manager, as described below.

Complete the following sets of steps to test your form configuration before publishing:

1. In the left-hand tree, expand the **Applications** node and navigate to the target template.
2. Right-click the target template and select **Test** from the context menu.
3. In the "Logon Manager Template Test Manager" window that appears, do the following:
   a. In the **Forms** pane, select the target form.
   b. Follow the instructions displayed in the **Interactions** pane.



| Name | |
|---|---|
| My Application | Logon |

Waiting for the ESSO-LM Agent...
Enabling Agent test mode...
Publishing the template...
Monitoring...

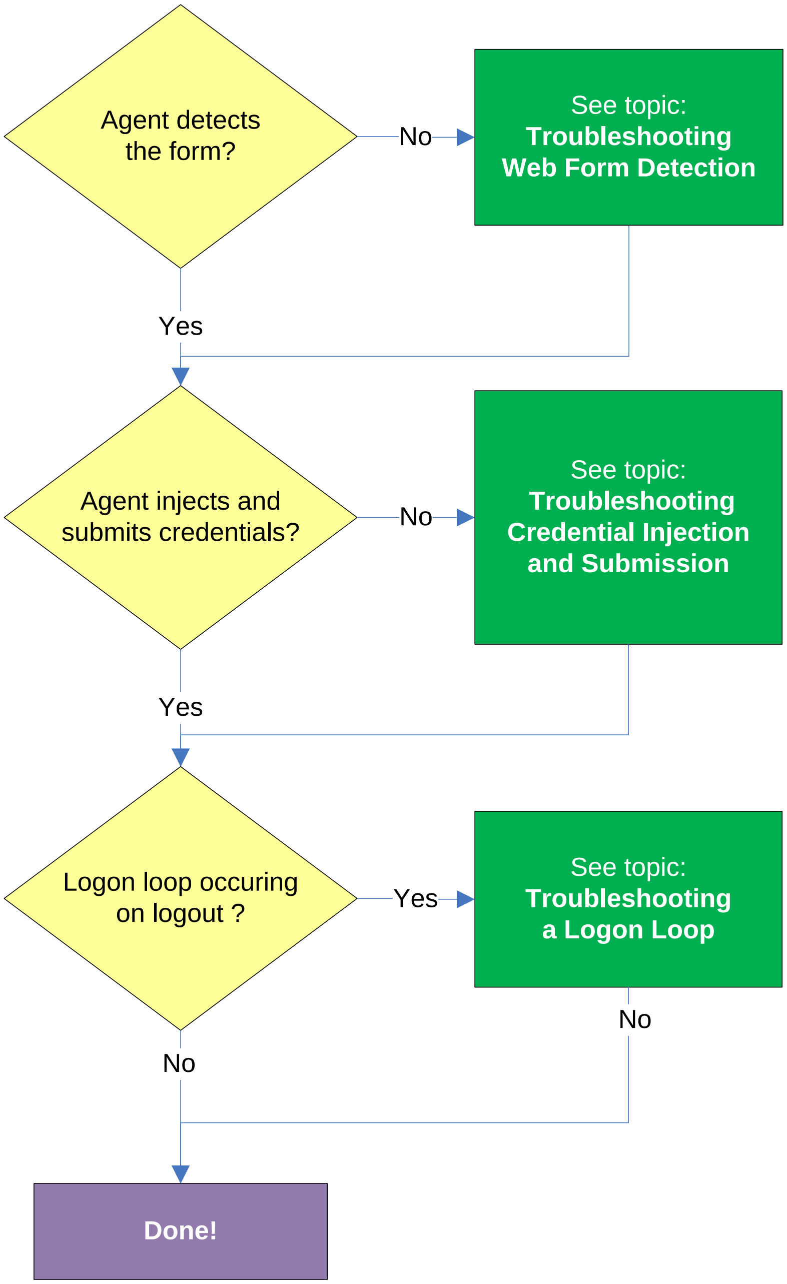
Detection

Actions
1. Launch the application: My Application
2. Open the application window: My Application

Does the Agent detect the window?
○ Yes
○ Yes, but it also responds to other windows that it should ignore.
○ No

   c. Do one of the following:
      - If the Agent responds to the application as desired and the test has completed successfully, click **Finish**.
      - If the Agent is not responding to the application as desired, and the instructions provided by the Template Test Manager do not resolve the issue, click **Close** and follow the troubleshooting flowcharts in the rest of this guide to determine and correct the problem. When you have finished, repeat steps 1-3 to test the corrected 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.

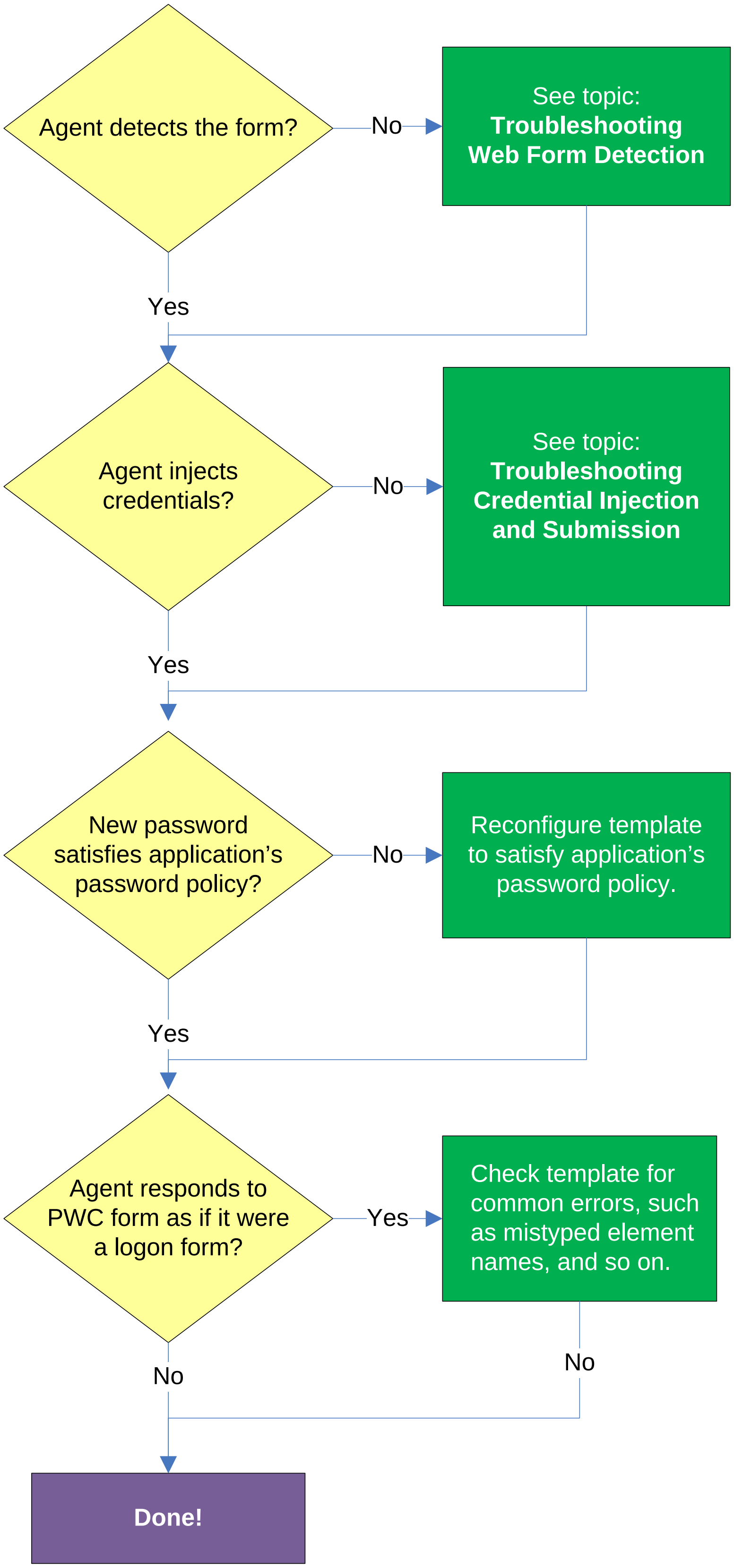
### Agent injects credentials?

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 Logon Manager 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

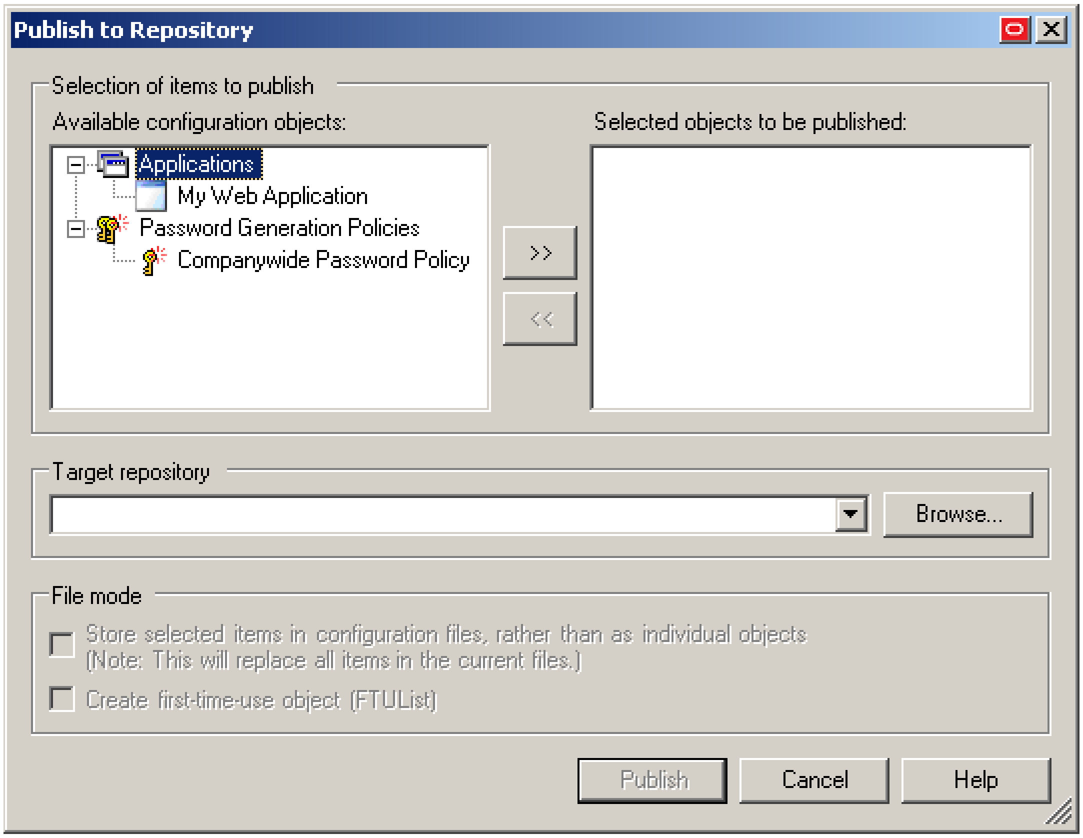
Once you have successfully tested your application template, you can distribute it to end-user machines by publishing it to the selected target container within your repository, either in a directory-style hierarchy (default), or as a flat configuration file.

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

**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 Logon Manager 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:
Selected objects to be published:
Applications
My Web Application
Password Generation Policies
Companywide Password Policy
>>
<<
Target repository
Browse...
File mode
Publish
Cancel
Help

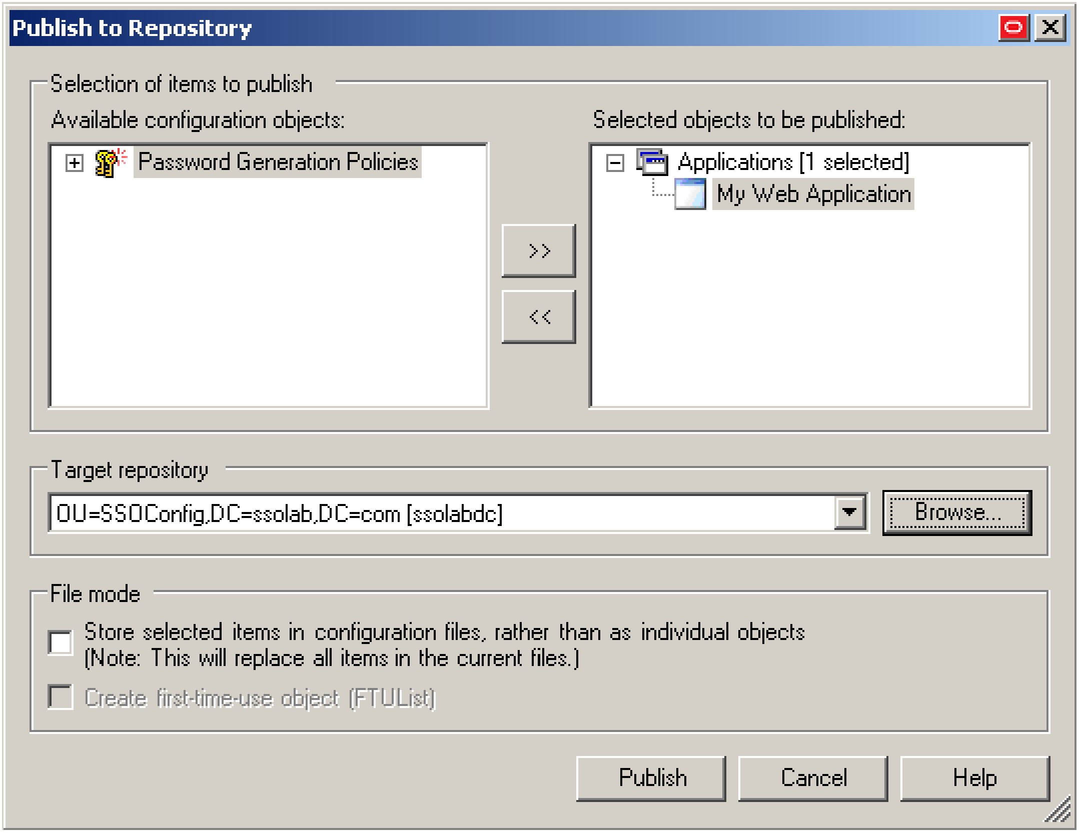
☐ Store selected items in configuration files, rather than as individual objects (Note: This will replace all items in the current files.)

☐ 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

OU=SSOConfig,DC=ssolab,DC=com [ssolabdc]

File mode

☐ 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:

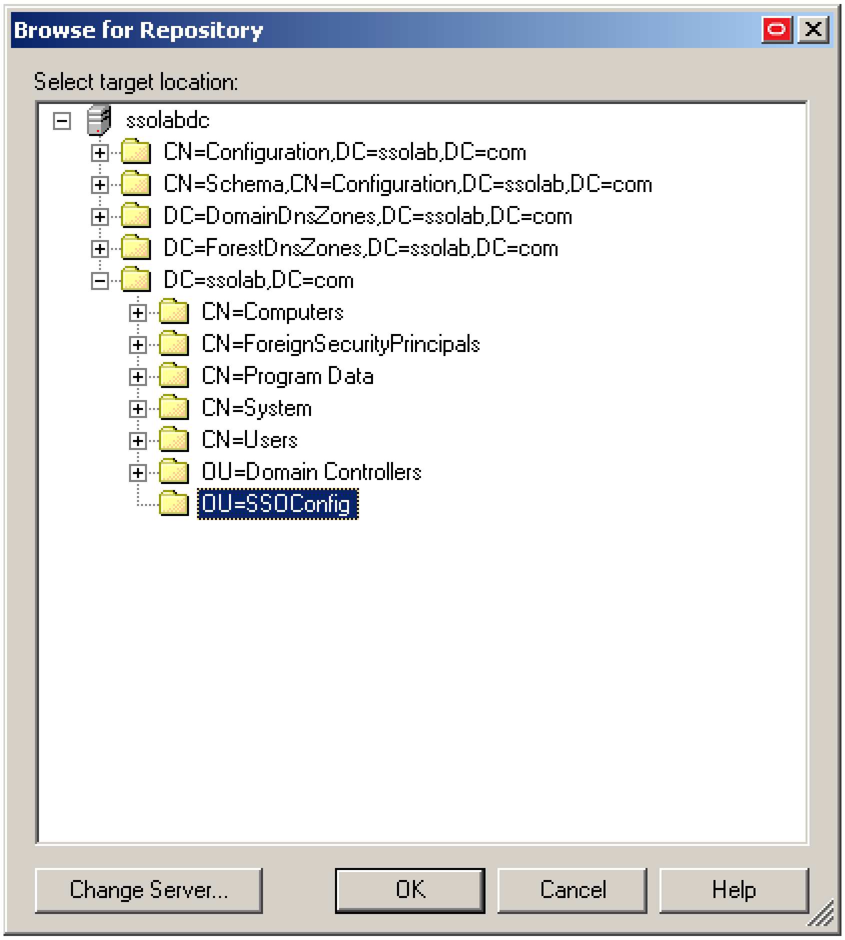
   - If you have previously published to the desired container, select it from the **Target Repository** drop-down list.

   - 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.



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.

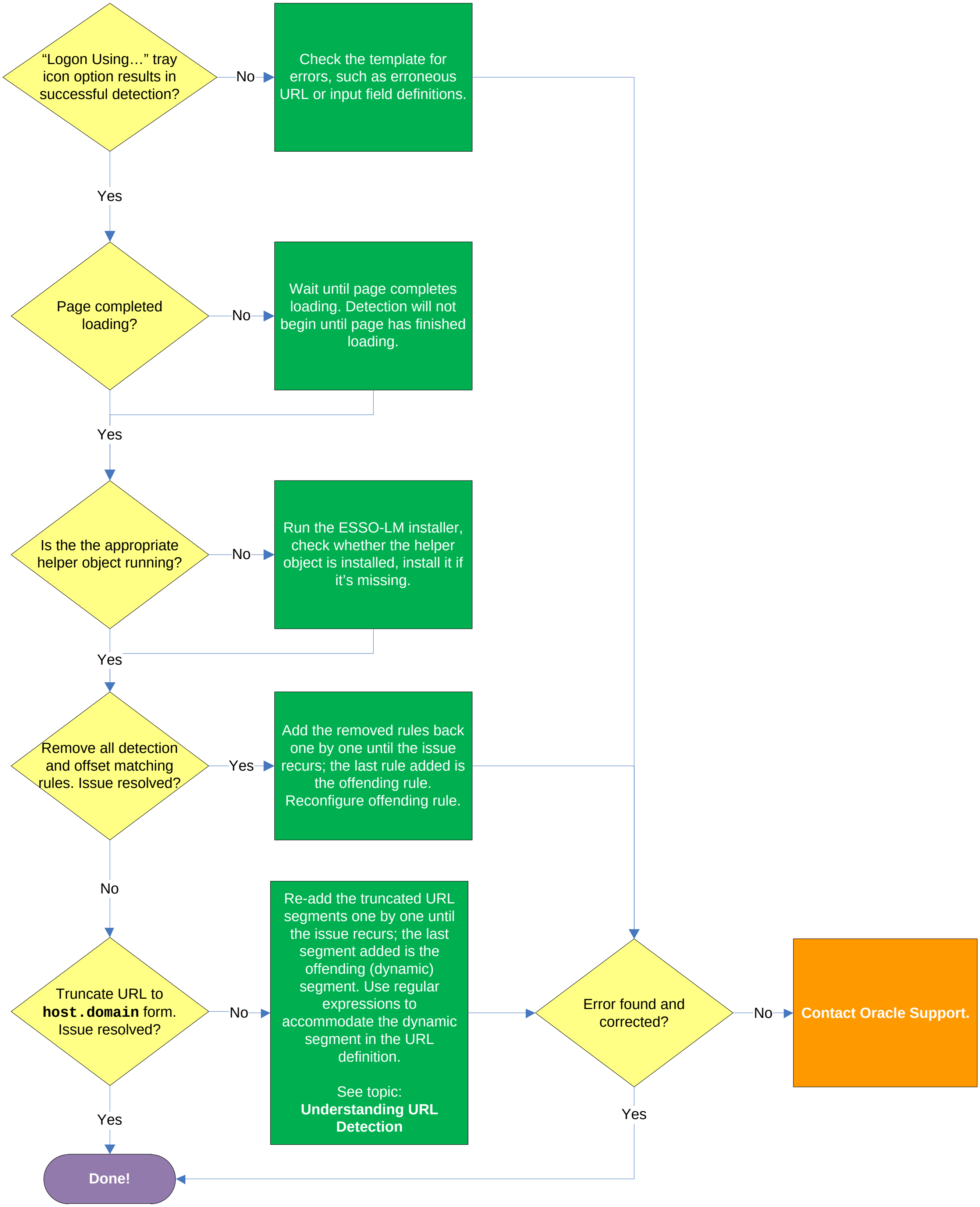
8. Click **Publish**. The Console publishes the selected objects to the target repository.

> **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 Logon Manager Administrative Console help.

## 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 Logon Manager” tray icon option results in successful detection?

If the Agent does not detect the form, but invoking the **Logon Using Logon Manager** option from the Agent’s system tray icon results in successful detection do the following:

- Make sure that the **Auto-recognize** option in the form definition’s **Options** tab is enabled.
- 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 Logon Manager with a means of sending and receiving data to/from the browser. If the helper objects are not running in the background, Logon Manager 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?

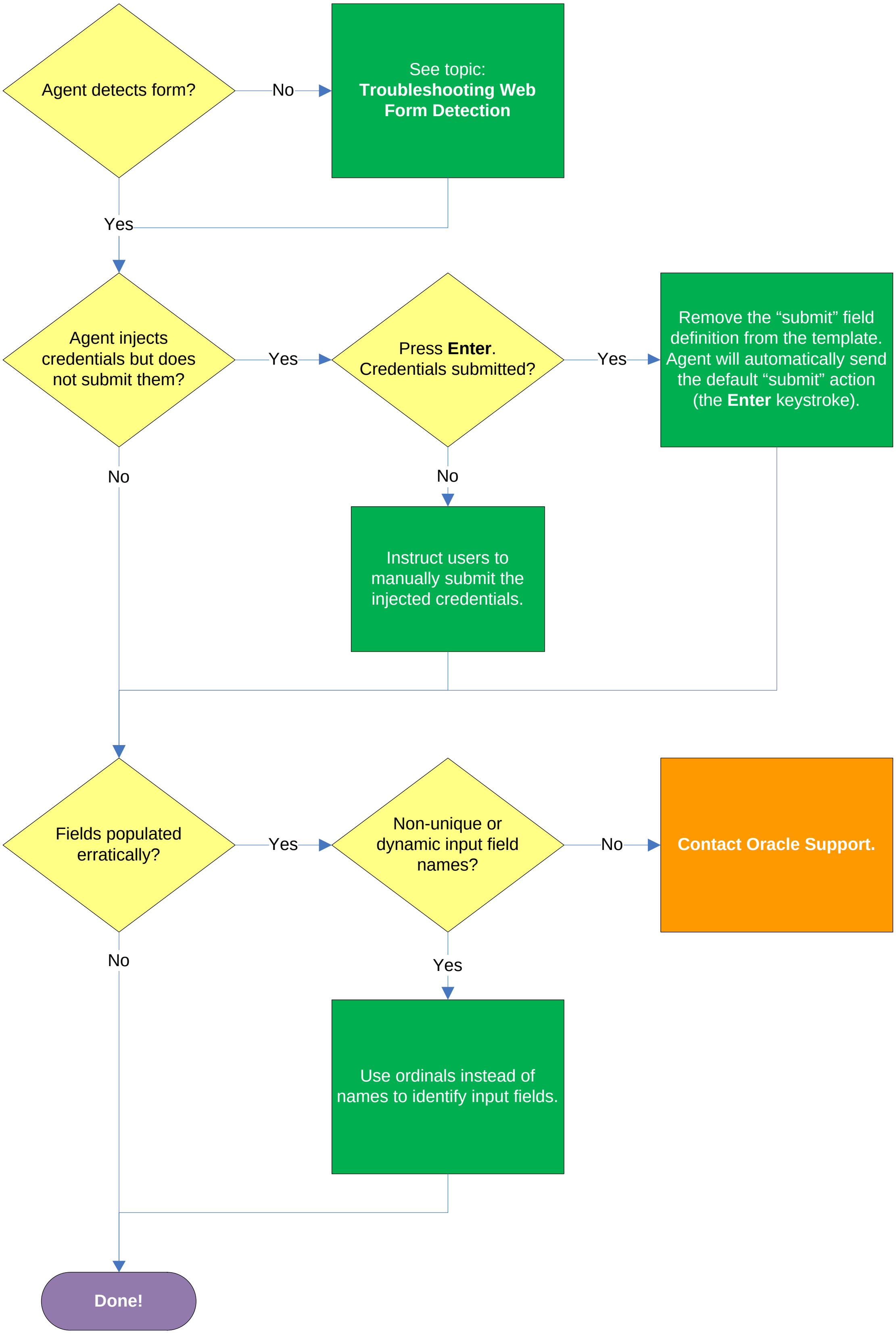
You might have configured one or more matching rules in a way which prevents successful detection of the target Web form. Remove all matching rules and re-add them one by one while testing detection until you identify the offending rule.

### 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 Form Response when Using Control IDs



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:

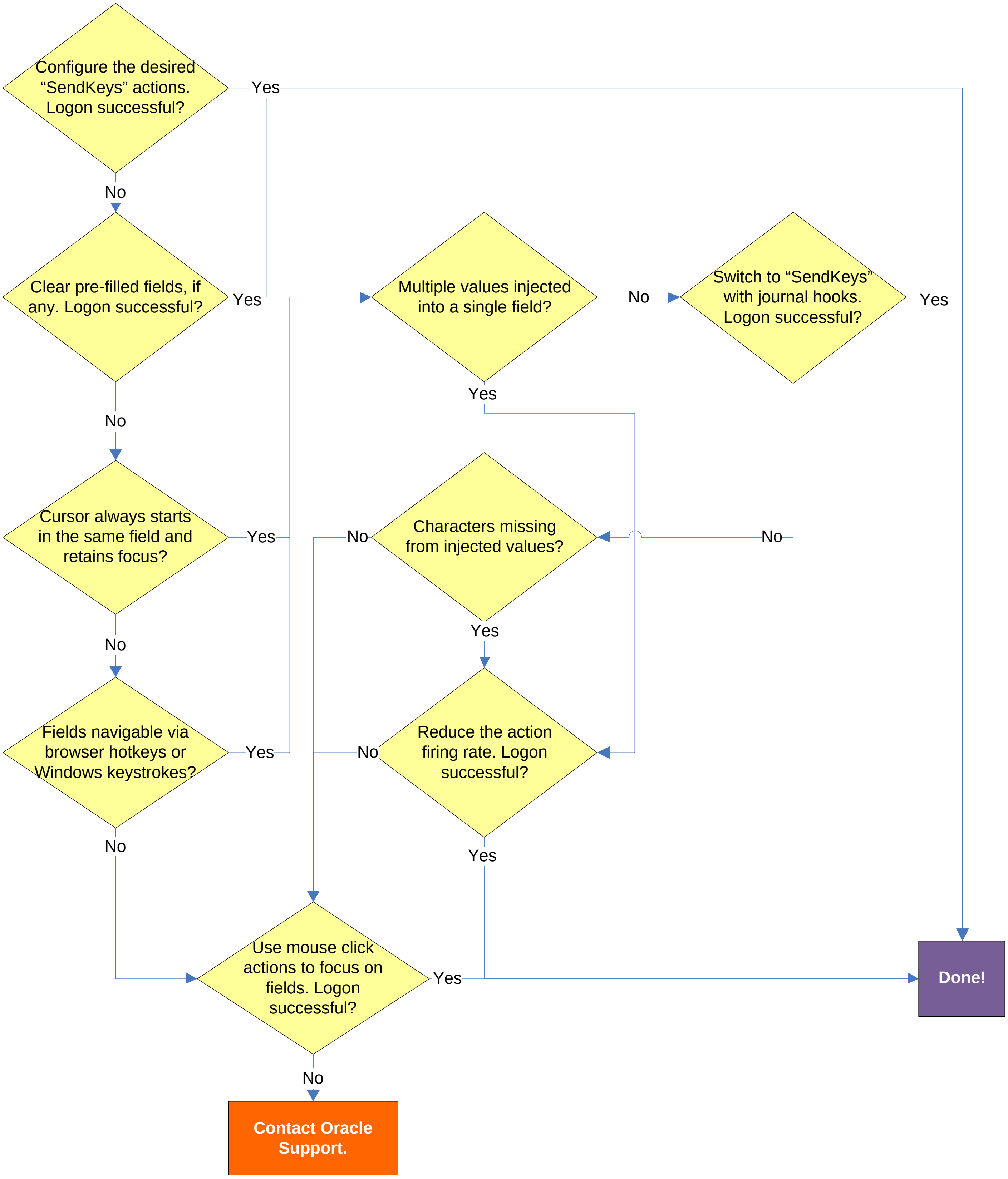
- If pressing **Enter** submits the credentials, remove the "submit" input field definition from the template. When the "submit" field definition is absent, the Agent will send its default submit action – the **Enter** keystroke.
- If pressing **Enter** does not submit the credentials, instruct the users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

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

## Troubleshooting Form Response When Using "SendKeys"



Configure the desired "SendKeys" actions. Logon successful?
Yes
No
Clear pre-filled fields, if any. Logon successful?
Yes
No
Cursor always starts in the same field and retains focus?
Yes
No
Fields navigable via browser hotkeys or Windows keystrokes?
Yes
No
Multiple values injected into a single field?
No
Yes
Switch to "SendKeys" with journal hooks. Logon successful?
Yes
No
Characters missing from injected values?
No
Yes
Reduce the action firing rate. Logon successful?
No
Yes
Use mouse click actions to focus on fields. Logon successful?
Yes
No
Done!
Contact Oracle Support.

### Pre-filled fields cause erroneous logon?

Some applications might pre-fill the logon fields when the logon form is displayed – for example, the user name field might be pre-filled with the name of the last successfully logged on user. You may have to send one or more **Backspace** or **Delete** key strokes to clear such a pre-filled field before injecting credentials into it.

### Cursor always starts in the same field and retains focus?

If the cursor does not always start in the same field and the field loses focus before the Agent populates it, see if the application permits you to navigate to the field through a specific hotkey combination (such as **Alt**+**U**) or by using standard Windows keys, such as **Tab**, arrows, and so on. If you cannot use keystrokes to navigate to the field, use a mouse click action targeting the desired field or control.

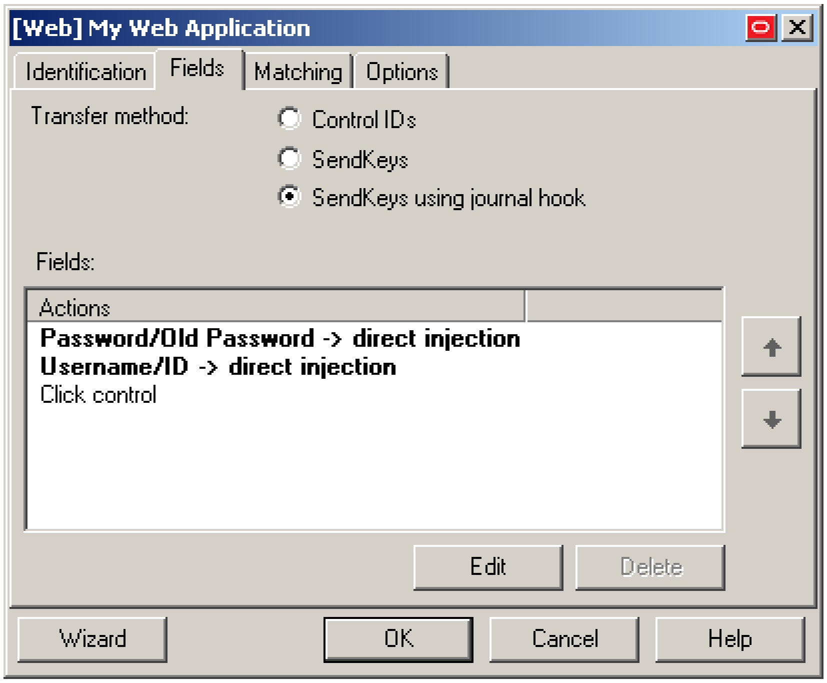
### Multiple values injected into a single field?

If the Agent is inserting multiple values (e.g., both the user name and the password) into a single field, it might be firing the "SendKeys" actions too quickly for the application to respond properly. In such cases, experiment with slowing down the action firing rate until credential injection is reliable. To do so, either insert a "Delay" action in between other actions, or set the **SendKeys event interval** global Agent setting under **End-User Experience → Response** to either **Use for slow system** or **Use for very slow system**.

### Switching to "SendKeys" with journal hooks restores reliable injection?

If the Agent continues injecting multiple values into a single field after you have tried the suggestions in the previous step, switch the form interaction method to "SendKeys with journal hook." To do so, select the **General** tab in the application template, select the desired form, click **Edit**, select the **Fields** tab, and set the **Transfer Method** option to **SendKeys using journal hook**.

> **Note:** The "SendKeys using journal hook" method causes the Agent to use an alternate API to send keystrokes and mouse clicks to the browser; it is typically the most effective in Citrix environments.



Transfer method:
○ Control IDs
○ SendKeys
● SendKeys using journal hook

Fields:

| Actions | |
|---|---|
| **Password/Old Password -> direct injection** | |
| **Username/ID -> direct injection** | |
| Click control | |

### Characters missing from injected values?

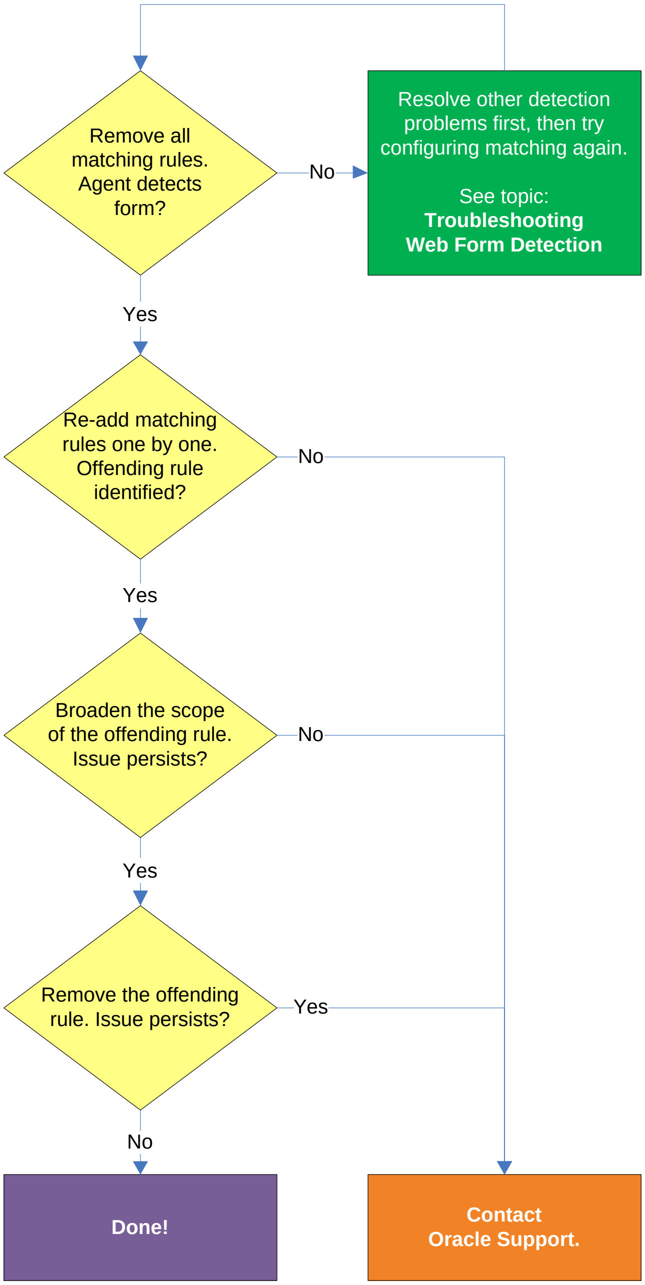
If you find that individual characters are omitted from the injected field values, the Agent might be firing the “SendKeys” actions too quickly for the application to accept them properly. In such cases, experiment with slowing down the action firing rate until credential injection is reliable. To do so, either insert a “Delay” action in between other actions or set the **SendKeys event interval** global Agent setting under **End-User Experience → Response** to either **Use for slow system** or **Use for very slow system**.

### Using mouse click actions to focus on fields results in successful logon?

If logon is still unsuccessful, consider using mouse click actions to focus on all fields and controls within the form.

If none of the above steps resolve your issue, contact Oracle Support for 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.

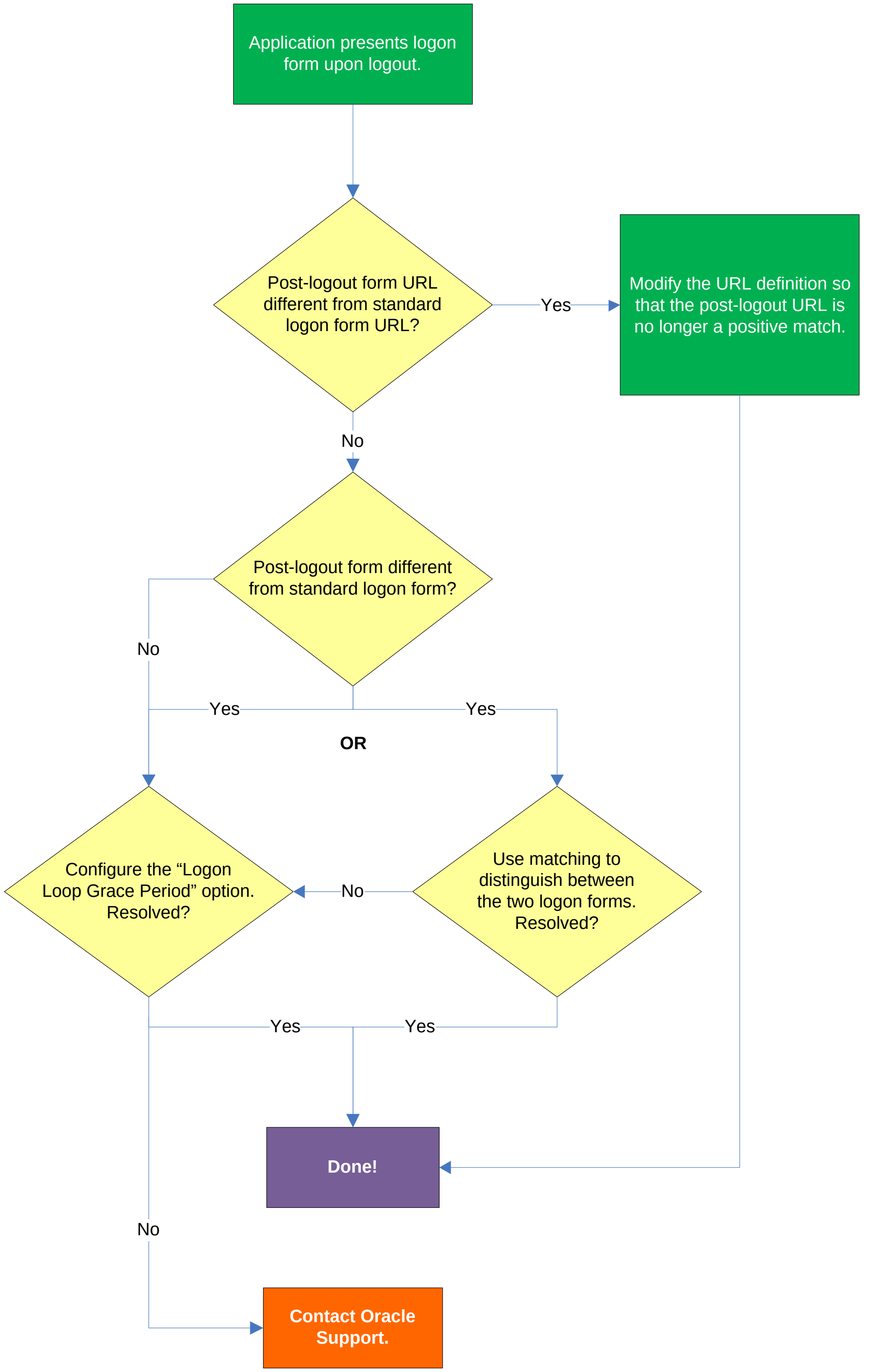
### Removing all matching rules restores form detection?

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 Logon Manager 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 Logon Manager 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:

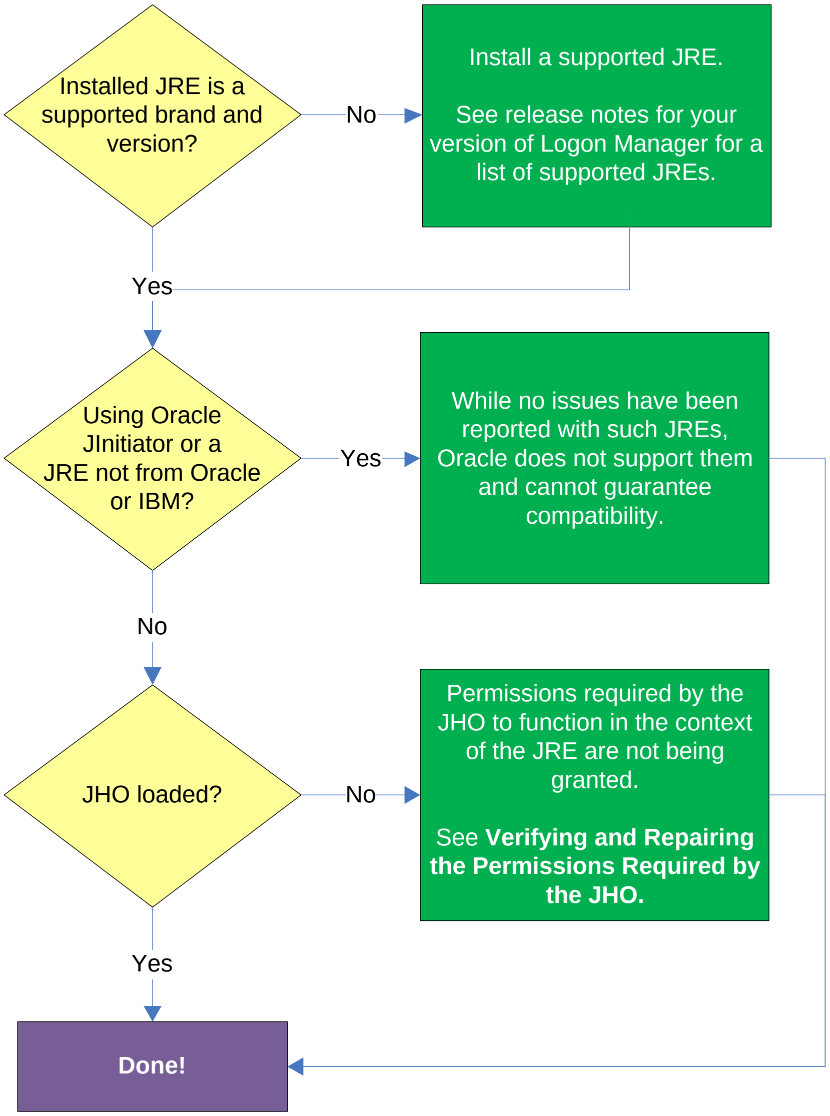
5. In the Logon Manager Administrative Console, open the desired template and select the **Miscellaneous** tab.
6. In the **Logon Loop Grace Period** field, select the desired mode of operation from the drop-down list:
   - **Prompt** – if the Agent detects the application's logon form while the grace 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.
7. 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.
8. Save your changes and commit them to your repository, if applicable.

**Note:** If you have configured the logon grace period timer, and logon loop is still occurring for a specific form definition, make sure that the **Adhere to logon loop grace period** option in the form definition's **Options** tab is enabled.

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 Logon Manager for a list of supported JREs.
Yes
Using Oracle JInitiator or a JRE not from Oracle or IBM?
Yes
While no issues have been reported with such JREs, Oracle does not support them and cannot guarantee compatibility.
No
JHO loaded?
No
Permissions required by the JHO to function in the context of the JRE are not being granted.
See Verifying and Repairing the Permissions Required by the JHO.
Yes
Done!

### Installed JRE is a supported brand and version?

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

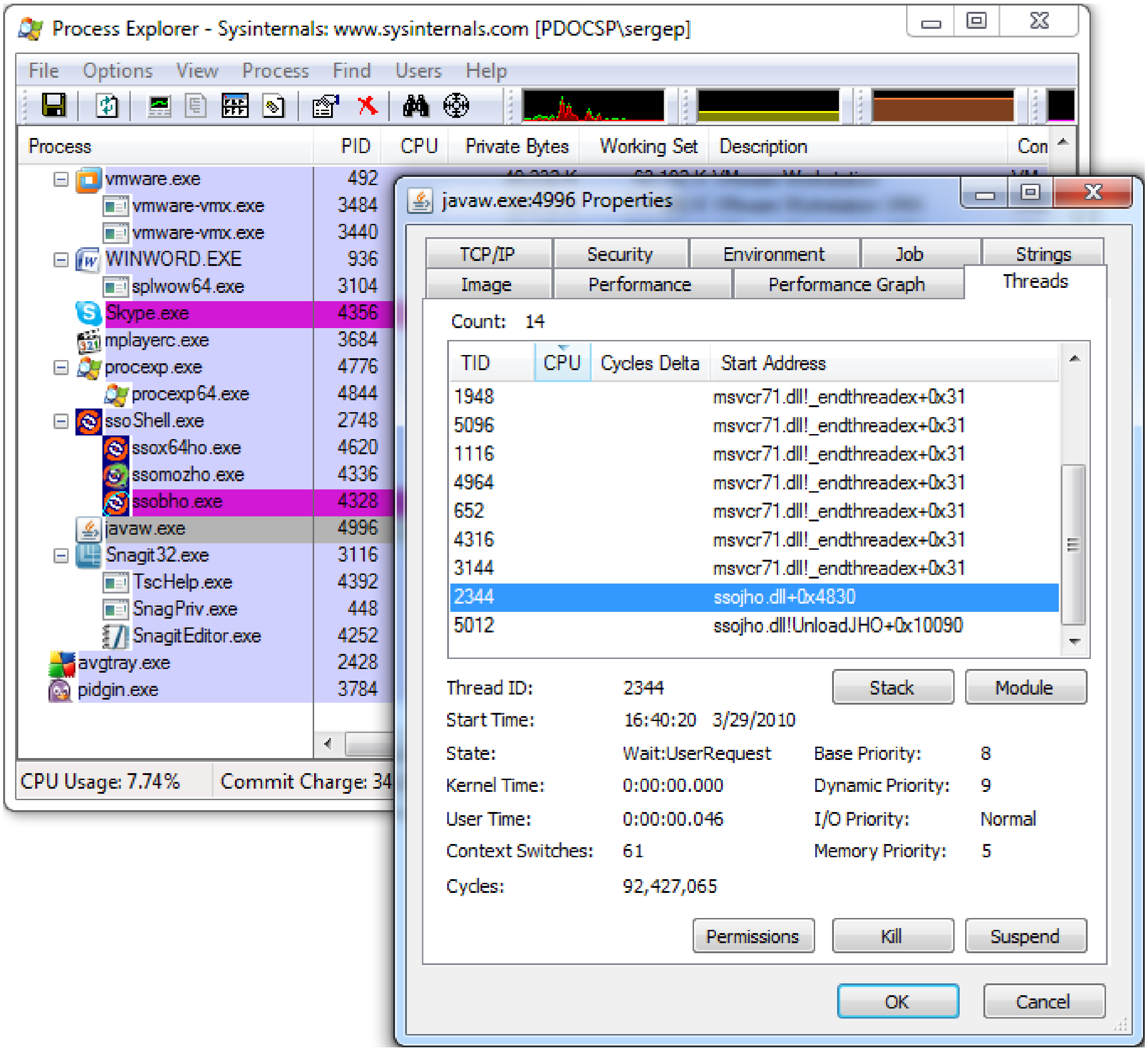
### Using Oracle JInitiator or another JRE not made by Oracle or IBM?

While no issues have been reported when deploying Logon Manager with Oracle JInitiator or non-Oracle/non-IBM JREs, Oracle does not support nor warrant the proper functioning of Logon Manager with such JREs.

### JHO loaded?

In certain situations, a configuration issue might prevent the JHO from loading when the Java application 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 |

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 the JHO is not loading for the target JRE, check that the permissions required by the JHO are being granted through the user's `.java.policy` file (located in their home directory) as described in the next section.

## Verifying and Repairing the Permissions Required by the JHO

The JHO requires that a set of specific permissions is granted to it via the user's `.java.policy` file so that it can properly operate in the context of the target JRE. If these permissions are not in effect, Logon Manager will be unable to detect or respond to Java applications.

### Verifying That the Required Permissions Are Being Granted

Every time Logon Manager starts, the permissions are copied from a template `.java.policy` file (located in `%INSTALLDIR%\v-GO SSO\Helper\Java`) to the user's `.java.policy` file located in the user's home directory. These permissions are then automatically granted to the JHO every time a Java application is launched.

If you suspect that the required permissions are not being granted to the JHO, check your JRE's log after starting Logon Manager for any missing permission exceptions referencing the JHO. If such exceptions are present, the required permissions are not being granted. This can be caused by one of the following:

- The user's `.java.policy` file does not contain the permissions required by the JHO,
- The target JRE's security settings (usually stored in the JRE's `java.security` file) do not reference the user's `.java.policy` file.

### Restoring the Missing JHO Permissions

If the user's `.java.policy` file is missing the required JHO permissions, the file's ACL might be preventing Logon Manager from writing the permissions to it; in such case, check the file's ACL to make sure that write privileges are not being denied.

If the ACL is correct and the required permissions are not present in the user's `.java.policy` file, do one of the following:

- If the user's `.java.policy` file contains other permissions or configuration information that you want to preserve, edit the file and manually add the missing JHO permissions.
- If the user's `.java.policy` file does not contain other permissions or configuration information that you wish to preserve, simply delete it and restart Logon Manager; the file will be recreated and the required permissions inserted automatically.

**Note:** As JREs are updated by their vendors, it is possible that a future release introduces one or more new permissions that will be required by the JHO but which could not be accounted for at the time of Logon Manager's release. In such case, examine your JRE's error log for any missing permission exceptions after launching Logon Manager and a target Java application and add those permissions to Logon Manager's `.java.policy` template file so that they can be granted via the user's `.java.policy` file the next time Logon Manager starts.

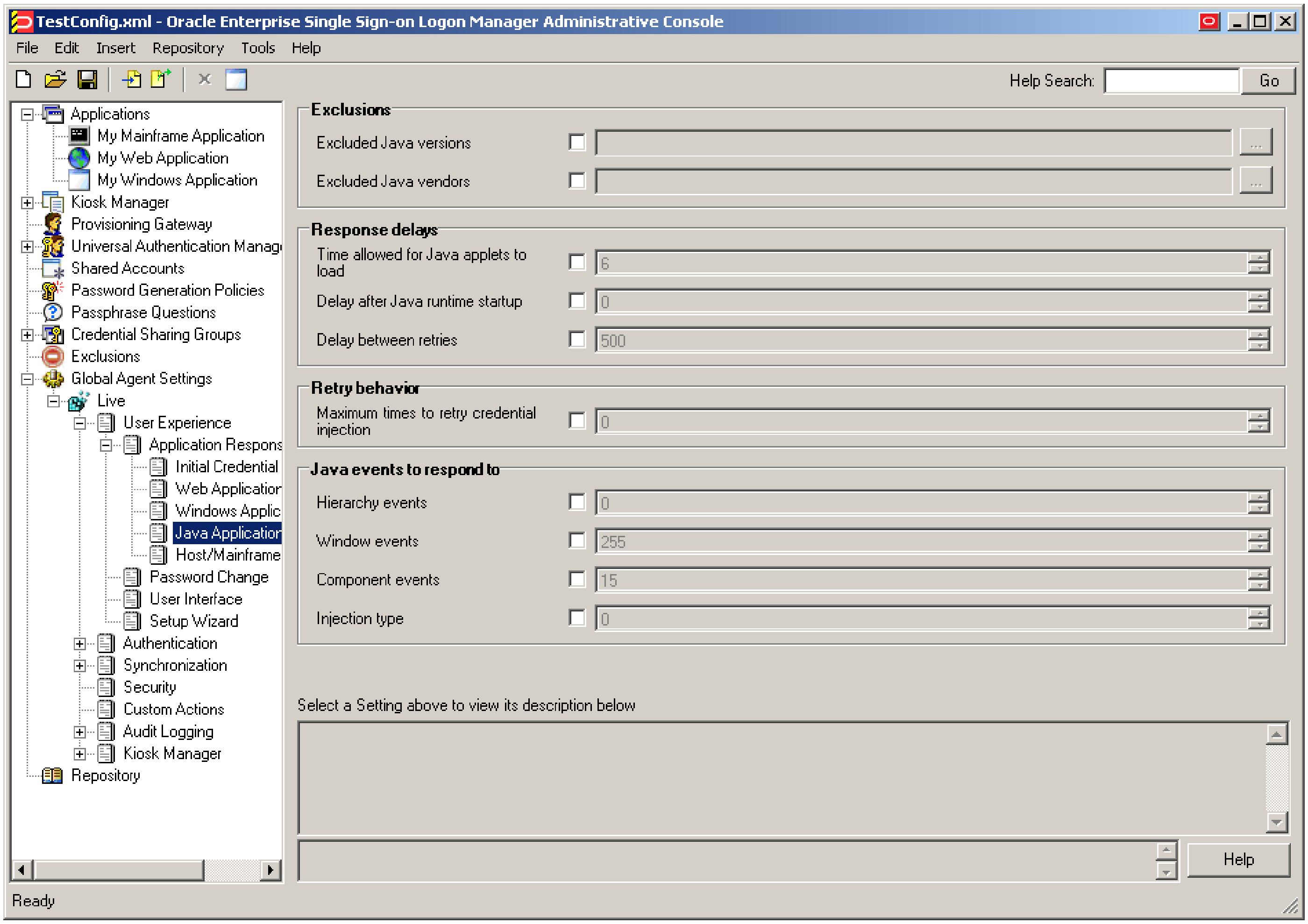
## Configuring the Behavior of the Java Helper Object (JHO)

Logon Manager provides global Agent settings that allow you to control the following aspects of the JHO's behavior:

- Exclude specific JRE versions
- Exclude specific JRE vendors
- Define a response delay to account  for the loading of Java applets
- Define  a response delay to account for the initialization of the JRE
- Define a delay between response retries
- Define a maximum number of response retries
- Define specific hierarchy, window, component and injection type events that the JHO will recognize or ignore

These settings are located in the tree in the Logon Manager Administrative Console at the following location:

**Global Agent Settings → Live → User Experience → Application Response → Java Applications**



**Exclusions**

Excluded Java versions □

Excluded Java vendors □

**Response delays**

Time allowed for Java applets to load □ 6

Delay after Java runtime startup □ 0

Delay between retries □ 500

**Retry behavior**

Maximum times to retry credential injection □ 0

**Java events to respond to**

Hierarchy events □ 0

Window events □ 255

Component events □ 15

Injection type □ 0

Select a Setting above to view its description below

To learn more about these settings and how to configure them, see the Console help. For even more granular control, Logon Manager also provides registry settings described in the next sections. Oracle recommends these settings be used only by experienced administrators.

### Manually Restricting the JHO to Specific JREs

Aside from the Console settings described earlier, Logon Manager provides registry settings that allow you to create extremely granular inclusion and exclusion rules that Logon Manager 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).<br>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:<br>`JhoIncludeHostName1=.*javaw?\.exe`<br>No-value default:<br>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.<br>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:<br>`JhoIncludeHostCommandLine1=.*Login\.jar.*`<br>No-value default:<br>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).<br>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:<br>`JhoExcludeHostName1=.*javaw?\.exe`<br>No-value default:<br>Blank (i.e., nothing will cause the JHO to ignore the application) |
| `JhoExcludeHostCommandLineN` | Specifies the command used to launch the application.<br>The command string usually includes the full path and name of the JVM executable, the JAR file, and any required command-line parameters.<br>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:<br>`JhoExcludeHostCommandLine1=.*Login\.jar.*`<br>No-value default:<br>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.<br>Default value:<br>`1.2` (the earliest JRE version supported by the JHO) |
| `JhoMaximumJavaVersion` | Specifies the highest allowed JRE/JDK version.<br>Default value:<br>Blank (i.e., no upper JRE version limit is imposed) |

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

Aside from the settings provided in the Console, Logon Manager provides registry settings that control the event response behavior of the JHO at an even more granular level. You can use these settings to troubleshoot and optimize Logon Manager performance when responding to Java applications by modifying the JHO'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:<br>`HIERARCHY_EVENT_CHANGED = 0x1`<br>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.<br>This flag is a combination of the following values:<br>• `WINDOW_EVENT_OPENED = 0x1`<br>• `WINDOW_EVENT_CLOSED = 0x2`<br>• `WINDOW_EVENT_ACTIVATED = 0x4`<br>• `WINDOW_EVENT_DEACTIVATED = 0x8`<br>• `WINDOW_EVENT_CLOSING = 0x10`<br>• `WINDOW_EVENT_ICONIFIED = 0x20`<br>• `WINDOW_EVENT_DEICONIFIED = 0x40`<br>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:<br>• COMPONENT_EVENT_SHOWN = 0x1<br>• COMPONENT_EVENT_HIDDEN = 0x2<br>• COMPONENT_EVENT_ADDED = 0x4<br>• COMPONENT_EVENT_REMOVED = 0x8<br>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:<br>• INJECT_TYPE_DEFAULT = 0<br>• INJECT_TYPE_METHOD = 1<br>• INJECT_TYPE_ACCESSIBLE = 2<br>• INJECT_TYPE_NONACCESSIBLE = 3<br>• INJECT_TYPE_ROBOT = 4<br>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:<br>• INJECT_TYPE_METHOD (if an appropriate set method had been found for the control)<br>• INJECT_TYPE_ACCESSIBLE (if the control supports accessibility)<br>• INJECT_TYPE_NONACCESSIBLE<br>• INJECT_TYPE_ROBOT<br>**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 Logon Manager:

- 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)

# **Part 3:** Mainframe Applications

This part explains the concepts necessary to understand how and why you should configure application templates to solve specific sign-on scenarios, as well as the recommended best-practice procedures for configuring and testing mainframe application forms, and diagnosis and resolution steps for the most common issues that may cause the Agent to erratically detect and/or respond to application forms. It covers the following topics:

- Overview of a Sign-On Event
- Understanding Form Detection and Response
- Determining the Optimal Configuration for a Form
- Configuring a Form
- Testing the Configuration of a Form
- Publishing a Template to the Repository
- Viewing and Modifying Emulator Configuration
- Troubleshooting Form Detection
- Troubleshooting Form Response
- Troubleshooting a Logon Loop
- Troubleshooting with the SSO MHO Status Tool

**Tip:** If the steps presented in this chapter do not resolve your issue, you can troubleshoot further by tracing and logging the activity of Logon Manager and submitting the logged information to Oracle Support for analysis. For this purpose, Oracle Support provides the Trace Controller utility. For information on how to use the utility, see the “Using the Trace Controller Utility” topic in the *Enterprise Single Sign-On Suite Plus Administrator’s Guide*.

## Overview of a Sign-On Event

Logon Manager can provide single sign-on capability for mainframe/host applications accessed via a terminal emulator. A terminal emulator is an application that establishes a terminal session from the user's Windows workstation to a multi-user mainframe system over a variety of protocols, including IBM 5270/5050, Telnet, SSH, and so on. Logon Manager 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.

> **Note:** The list of currently supported emulators can be found in the release notes for your version of Logon Manager. You can also find out whether an emulator is supported by viewing the contents of the `mrfmlist.ini` file using the Logon Manager Administrative Console.
> See Viewing and Modifying Emulator Configuration for instructions.

In order to recognize text-based mainframe application forms displayed within supported Windows-based terminal emulators, Logon Manager does the following:

1. **Detects the emulator's HLLAPI interface, window, terminal session, and target form**:
   a. At startup, loads the emulator detection data specified in the `mfrmlist.ini` file and begins monitoring all HLLAPI interfaces detected on the system for new terminal sessions. (If the emulator does not provide HLLAPI notifications and polling is enabled, Logon Manager queries those emulators via HLLAPI every 700ms, by default).

   > **Note:** The `mfrmlist.ini` file contains detection data (such as HLLAPI interface library name and path, window title, class, and any required custom configuration parameters) for supported terminal emulators. This data allows Logon Manager to identify and interact with emulators and the terminal sessions they display.

   b. When a running emulator notifies Logon Manager via HLLAPI that a new terminal session has been opened (or, if polling is enabled, Logon Manager detects a new session in the emulator), Logon Manager searches through all available mainframe application templates for a template containing a form definition that matches the text displayed in the terminal session.
   c. Once per each detected HLLAPI event (keystroke, screen update, and so on), rescans the available mainframe application templates and re-detects the specified match text.
2. **Responds to the detected form and completes the logon**:
   a. Retrieves the associated credentials from the user's store (if they exist) and injects them at field coordinates defined in the template. (If the credentials don't exist, the Agent prompts the user to store them.)
   b. Submit the credentials to the application for processing.

c. (Optional) Detects any follow-up forms or dialogs, such as new password change success or failure, and performs the required action.

## Understanding Form Detection and Response

Logon Manager uses the Mainframe Helper Object (MHO), instantiated as a background process named `ssomho.exe`, to interact with terminal emulators via the High-Level Language API (HLLAPI). This allows Logon Manager to programmatically detect an emulator's HLLAPI interface, monitor (or query) the emulator for session instantiation, screen updates, and other events, and interact with the emulator to detect the match text, inject credentials, and submit them to the application for processing.

The MHO directly interfaces with most modern 32-bit emulators via HLLAPI. Additionally, Oracle provides support for some common non-HLLAPI emulators and 16-bit emulators, for which you need to install additional helper objects listed below (available under the **Extensions → Mainframe Support** node in the Logon Manager installer):

- **Console windows** – supports Windows command-line console windows instantiated by the `cmd.exe` interpreter and running 32-bit applications, such as the Microsoft Telnet client.
- **PuTTY** – supports the PuTTY terminal emulator.

**Note:** If you need a Telnet, SSH, or RS-232 connection to an application and your current terminal emulator only supports HLLAPI for IBM 5270/5050 connections, Oracle supports and highly recommends using the free PuTTY terminal emulator for connecting via those protocols.

When Logon Manager starts, the MHO reads the contents of the `mfrmlist.ini` file (located in `%PROGRAMFILES%\Passlogix\v-GO SSO\Helper\Emulator`), which contains detection data (such as HLLAPI interface library name and location, window title and class, executable name, and any custom configuration options required to support the emulator) for the supported emulators.

**Note:** While Logon Manager ships with a "factory" `mfrmlist.ini` file, you also have the ability to add an unlisted emulator to the file in order to permit Logon Manager to detect and interact with that emulator. For more information, see Viewing and Modifying Emulator Configuration.

The MHO then parses the configuration data for each emulator present in the `mfrmlist.ini` file and begins monitoring (or polling, if configured as such) each registered HLLAPI interface it detects. Once the configuration data has been loaded, the MHO monitors the detected HLLAPI interfaces for new session short-names.

For each new detected terminal session short-name, the MHO queries the emulator via HLLAPI for more information, such as the title and class of the session window. This information is matched against the window title and class present in the emulator's entry in the `mfrmlist.ini` file.

While HLLAPI in itself is an open standard, emulator vendors often implement terminal emulators to varying degrees of adherence and/or completeness. Because of this, some HLLAPI-enabled emulators do not properly notify Logon Manager of session events, such as session instantiation, screen updates, and so on. For those emulators, the `ChangeNotificationBroken=1` configuration parameter, which enables polling, must be present within their entries in the `mfrmlist.ini` file.

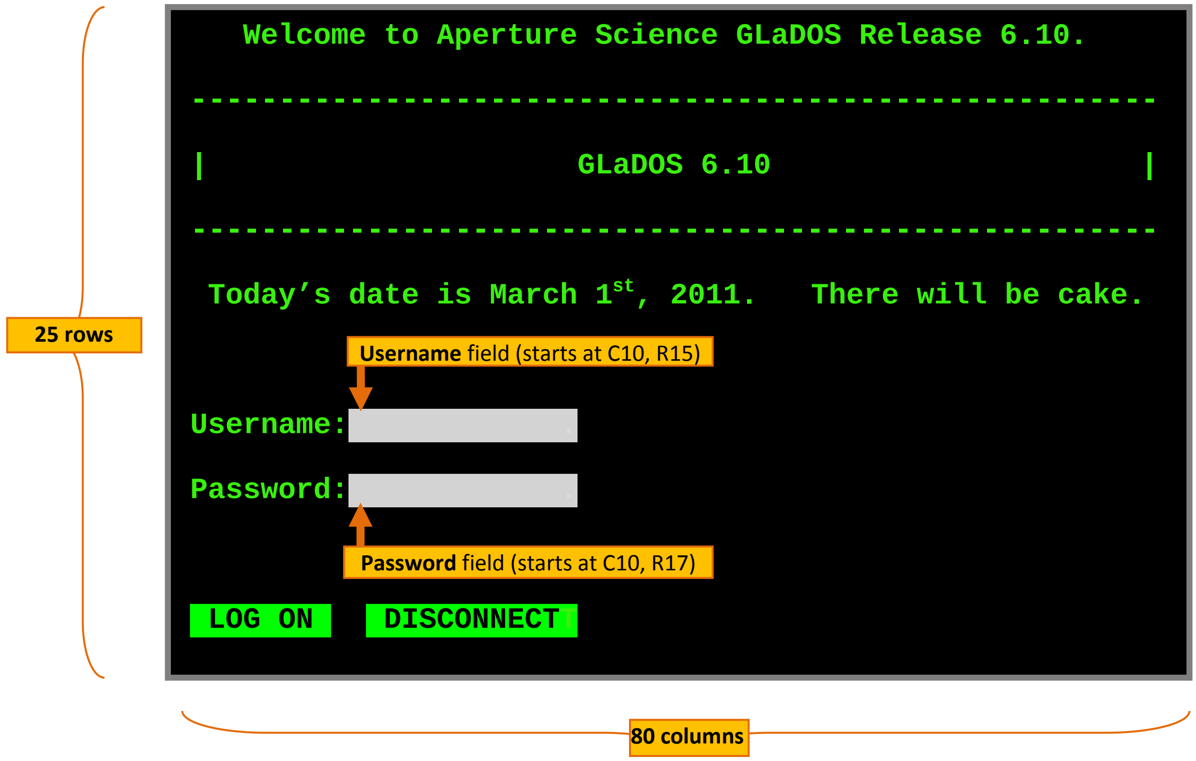
**Note:** See Viewing and Modifying Emulator Configuration for instructions on how to view an emulator entry in the `mfrmlist.ini` file and add this parameter, if necessary. (Oracle-tested emulators that require this parameter are preconfigured to include it in the `mfrmlist.ini` file "out of the box.")

Once the emulator has been positively identified, the MHO examines all available mainframe application templates for matches against the text displayed in the terminal session and loads the first template that is a positive match against the detected form text. The MHO also rescans all mainframe templates and re-detects the match text every time new HLLAPI event is detected in the emulator.

**Note:** An *application template* is a set of configuration options that instruct the Logon Manager Agent how to detect and respond to application windows and the forms they contain.

Mainframe applications operate in *text-mode*, i.e., they use a set of monospaced characters displayed on a known-size grid - typically 80 rows by 25 columns - that comprises the terminal screen. Because of this, each character on the screen can be uniquely identified by a set of vertical and horizontal coordinates. When defining a mainframe application form, you provide v Logon Manager with the starting coordinates of each target field (i.e., the coordinates where Logon Manager should begin injection).

For example, the **Username** field begins at column 10, row 15 (i.e., Logon Manager will inject the user name starting after the semicolon in row 15), while the password field begins two rows lower, at column 10, row 17, after the `Password:` field text:



```
   Welcome to Aperture Science GLaDOS Release 6.10.

-------------------------------------------------------

|                    GLaDOS 6.10                      |

-------------------------------------------------------

 Today's date is March 1st, 2011.   There will be cake.


Username:

Password:


 LOG ON    DISCONNECT
```

25 rows
Username field (starts at C10, R15)
Password field (starts at C10, R17)
80 columns

**Note:** Logon Manager supports terminal displays of any size, and a single template will provide response to an application running in different sizes, as long as the absolute coordinates of the configured match text and fields remain the same when the screen size changes.

**Tip:** Most emulators display the coordinates of the terminal cursor in their status bar. To quickly determine the coordinates of a field, position the terminal cursor at the beginning of the field and check the position displayed in the emulator's status bar.

During detection, Logon Manager retrieves the coordinates of the specified match text and checks whether that text exists at the specified coordinates, while during response, Logon Manager locates the specified field coordinates and injects credentials into those screen locations.

## Fixed Screen Forms vs. Scrolling Screen Forms

The example shown and described earlier in this section portrays a fixed-screen application, which is the most commonly encountered type of application. However, some applications, such as Telnet clients, may utilize a scrolling-screen display. The differences are as follows:

- **Fixed-screen.** The application screen has a fixed horizontal (number of columns) and vertical (number of rows) size, i.e., it does not scroll vertically. Most mainframe applications, such as inventory or machine control systems, fall into this category. In a fixed-screen session, every character displayed in the session has a static position, i.e. its coordinates do not change.
- **Scrolling-screen.** The application screen has a fixed horizontal size (number columns) but an unlimited, or continuous, vertical size (number of rows). In a scrolling-screen session, fields do not have static coordinates; in order to locate them, Logon Manager requires you to specify the column position of the cursor (i.e., where the cursor is expected to be found horizontally after completing each step in the logon sequence), as well as the sequence of fields and actions required to complete the logon. Logon Manager can detect via HLLAPI whenever the screen has scrolled and by how much, and advance to the next field or action in the sequence as necessary.

**Note:** For scrolling-screen forms Logon Manager also supports fields that have dynamic horizontal coordinates, allowing you to input a regular expression as the horizontal coordinate. For more information, see Configuring a Form for Dynamic Column Positioning (Scrolling-Screen Only).

## Supported Form Types

As of the release date of this document, Logon Manager supports the following types of forms in mainframe applications:

- Logon
- Logon success (a message confirming successful logon)
- Logon failure (a message indicating the injected credentials have been rejected)
- Password change
- Password change success (a dialog confirming successful password change)
- Password change failure (a dialog indicating the new password was rejected)

A mainframe logon form, for example, typically contains at least a user name field, a password field, and, sometimes, additional controls that permit Logon Manager to navigate the screen. Usually credentials are submitted via the default Logon Manager “submit” action, the **Enter** keystroke (which should not be explicitly specified in the form definition).

**Note:** Some non-HLLAPI emulators may provide a scripting language that allows you to display a Windows-style logon dialog. In such cases, you can create a Windows application template for the scripted logon dialog and use SendKeys to inject credentials and complete the logon. For instructions on creating the required script, consult the emulator documentation provided by its vendor; for instructions on creating a Windows application template, see the *Best Practices* guide *Template Configuration and Diagnostics for Windows Applications*.

When defining a form in a mainframe application template, you must identify the coordinates of the target fields in the application's display. Since Logon Manager detects the emulator window before it detects the form displayed in the session screen, a single application template will work for all supported emulators that behave identically to the original emulator used to create the template. Thanks to HLLAPI, Logon Manager does not need to perform screen scraping nor does it need to analyze entire screens of text for a match – instead, Logon Manager re-detects the defined match text every time a new HLLAPI event is detected within the emulator.

The same terminal session may display different forms depending on the invoked function (i.e., logon or password change); thus, a single template can contain definitions for the multiple forms that the window can display. For most applications, you need to only define the forms to which you want Logon Manager to respond. Logon Manager can automatically populate the appropriate fields in a form with credentials retrieved from the user's credential store, as well as submit the credentials to the application for processing.  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.

**Note:** Logon Manager is unable to differentiate between multiple sessions of an application if the application display does not contain unique session identification information (such as a host name). For example, if you open two sessions to two mainframe hosts running the same application but each requiring different credentials, Logon Manager will attempt to complete the logon (or password change) with the credentials stored in the template for both sessions. In such cases Oracle recommends storing credentials for all possible sessions for an application within a single template and using the "Logon Chooser" feature to prompt the user to manually select the credentials for each session.

## Supported Credential Injection Methods

Depending on the design of the target application, Logon Manager can use one of the following methods to interact with the fields and controls in the target form:

- **HLLAPI** - This is the default and preferred form interaction method for most mainframe applications, particularly those using the IBM 5270/5050 protocols. This method uses the HLLAPI interface to programmatically inject credentials at the specified coordinates.
- **SendKeys** - This method injects credentials by emulating keystrokes. Use this method if the emulator does not support HLLAPI or when special actions are required to complete the logon sequence – for example, the application requires the user to manually advance to the next field via a keystroke, and the field remains disabled until the user navigates to it. Oracle recommends using HLLAPI whenever possible for a more robust configuration.

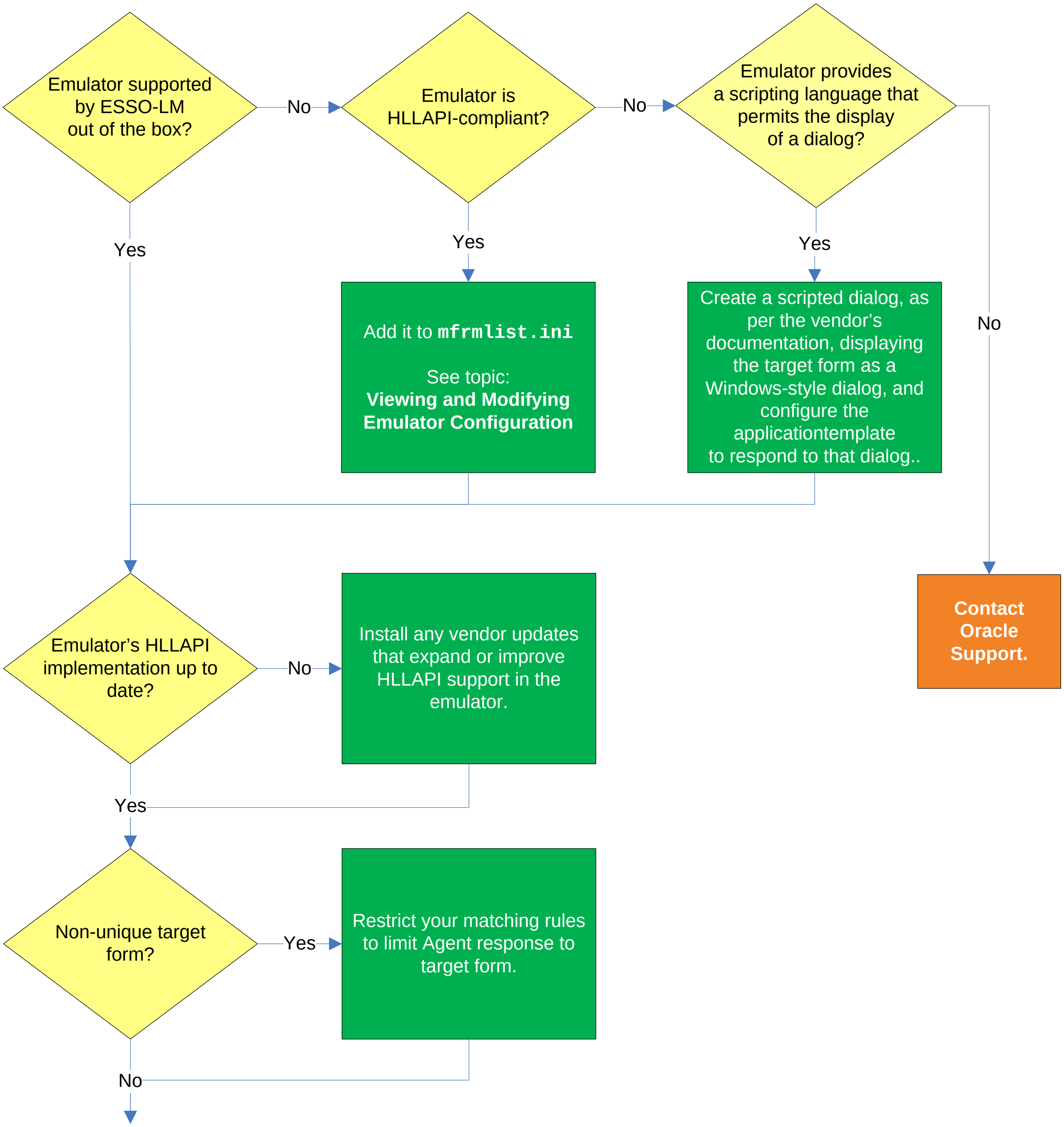
> **Note**: When configuring a mainframe application template, as soon as you begin adding additional fields or actions to the form definition after completing the wizard, Logon Manager switches from HLLAPI to SendKeys-based injection.

- **SendKeys as a Windows Application** – recommended only when no other injection method works, you may attempt to configure your mainframe as a Windows application and use SendKeys to both navigate the application's logon screen and inject credentials. This method is very limited in scope, as Logon Manager is unable to detect any screen updates – actions are sent to the application "blindly." For this reason, this method will only work for the application's initial screen (usually, the logon screen) and no subsequent screens that appear once logged in.

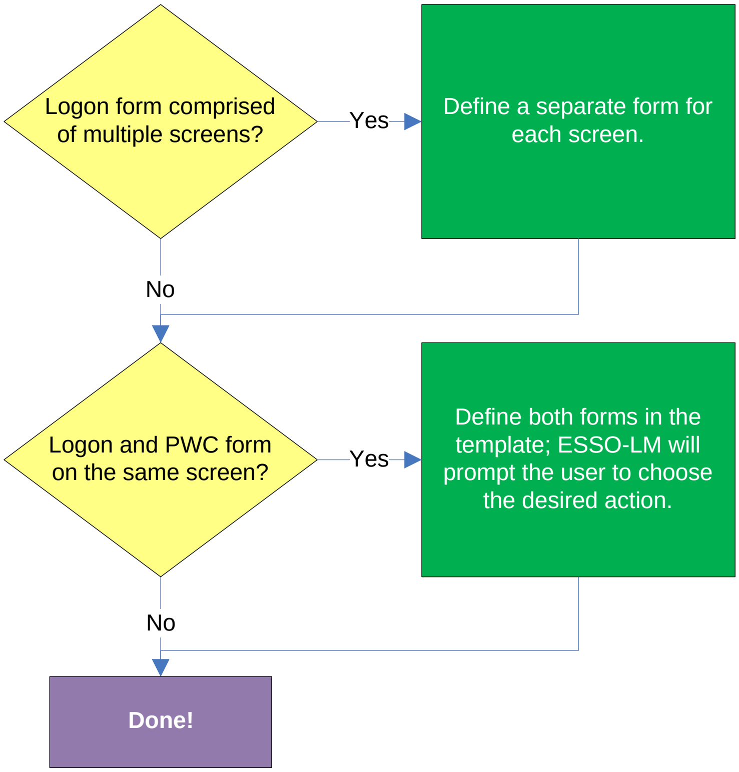
## Determining the Optimal Configuration for a Form

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



Emulator supported by ESSO-LM out of the box?
No
Emulator is HLLAPI-compliant?
No
Emulator provides a scripting language that permits the display of a dialog?
Yes
Yes
Yes
Add it to mfrmlist.ini
See topic: Viewing and Modifying Emulator Configuration
Create a scripted dialog, as per the vendor's documentation, displaying the target form as a Windows-style dialog, and configure the applicationtemplate to respond to that dialog..
No
Emulator's HLLAPI implementation up to date?
No
Install any vendor updates that expand or improve HLLAPI support in the emulator.
Contact Oracle Support.
Yes
Non-unique target form?
Yes
Restrict your matching rules to limit Agent response to target form.
No


Logon form comprised of multiple screens?
Yes
Define a separate form for each screen.
No
Logon and PWC form on the same screen?
Yes
Define both forms in the template; ESSO-LM will prompt the user to choose the desired action.
No
Done!

### Emulator supported by Logon Manager out of the box?

If the emulator has been tested by Oracle and found to be compatible with Logon Manager, it is supported by Logon Manager out of the box and a corresponding entry exists for the emulator in the `mfrmlist.ini` file. The list of currently supported emulators can be found in the latest application release notes. It can also be determined by looking for an entry for the emulator in question in the `mfrmlist.ini` file, as described in Viewing and Modifying Emulator Configuration.

If the emulator is not supported by Logon Manager out of the box and it is HLLAPI-compliant, you may add it to the `mfrmlist.ini` file (as described in Viewing and Modifying Emulator Configuration) in order to determine whether it is fully compatible with Logon Manager. While Oracle Support will aid you in this effort should you run into configuration issues, Oracle is unable to guarantee that an untested emulator will properly function with Logon Manager in all expected capacities.

**Note:** If you add an unlisted emulator to the mfrmlist.ini file and find it to be fully functional with Logon Manager, Oracle requests that you submit the emulator name and version to Oracle Support so that we can consider including official support for that emulator in the next release of Logon Manager.

### Is the emulator HLLAPI-compliant?

Depending on whether the emulator is HLLAPI-compliant, do one of the following:

- If the emulator is HLLAPI-compliant but is not officially supported by Logon Manager, you can manually add the emulator to the `mfrmlist.ini` file to allow Logon Manager to interact with it and determine whether Logon Manager's response to the target application via that emulator is reliable. For instructions, see Viewing and Modifying Emulator Configuration.
- If the emulator does not support HLLAPI, check whether it provides a scripting language that allows you to code a script that will display the application's logon and password-change forms as Windows-style dialogs. If the emulator has this scripting capability, write the script according to the vendor's documentation, then configure a Windows application template to respond to the Windows-style dialogs. For more information on creating Windows application templates, see the *Best Practices* guide *Template Configuration and Diagnostics for Windows Applications.*

**Note:** For assistance in writing the script, consult the emulator's vendor. Oracle is unable to provide help in the writing and/or troubleshooting of third-party scripts.

If neither of the above options is available, contact Oracle Support for assistance.

### Is the emulator's HLLAPI implementation up to date?

In some cases, the emulator's initial HLLAPI implementation might be incomplete or defective.
If possible, Oracle recommends researching and applying any vendor updates that improve or expand the emulator's HLLAPI implementation in a way that improves the emulator's compatibility with Logon Manager. For more information on the level of your emulator's HLLAPI implementation, consult the emulator vendor.

### Non-unique target form?

When configuring the form, you must select a match text string within the form that does not appear anywhere else within the application; otherwise, Logon Manager may respond to screens that contain the non-unique match text even though they are not the target form. If a unique match text string is not present in the target form, investigate the possibility of modifying the application in order to add a piece of unique text to the form.

If modifying the application is not an option, configure the "logon chooser" feature in the template and instruct users to select the desired credentials when logging on to the affected application. For information on the "logon chooser" feature, see the Console help.

### Logon form comprised of multiple screens?

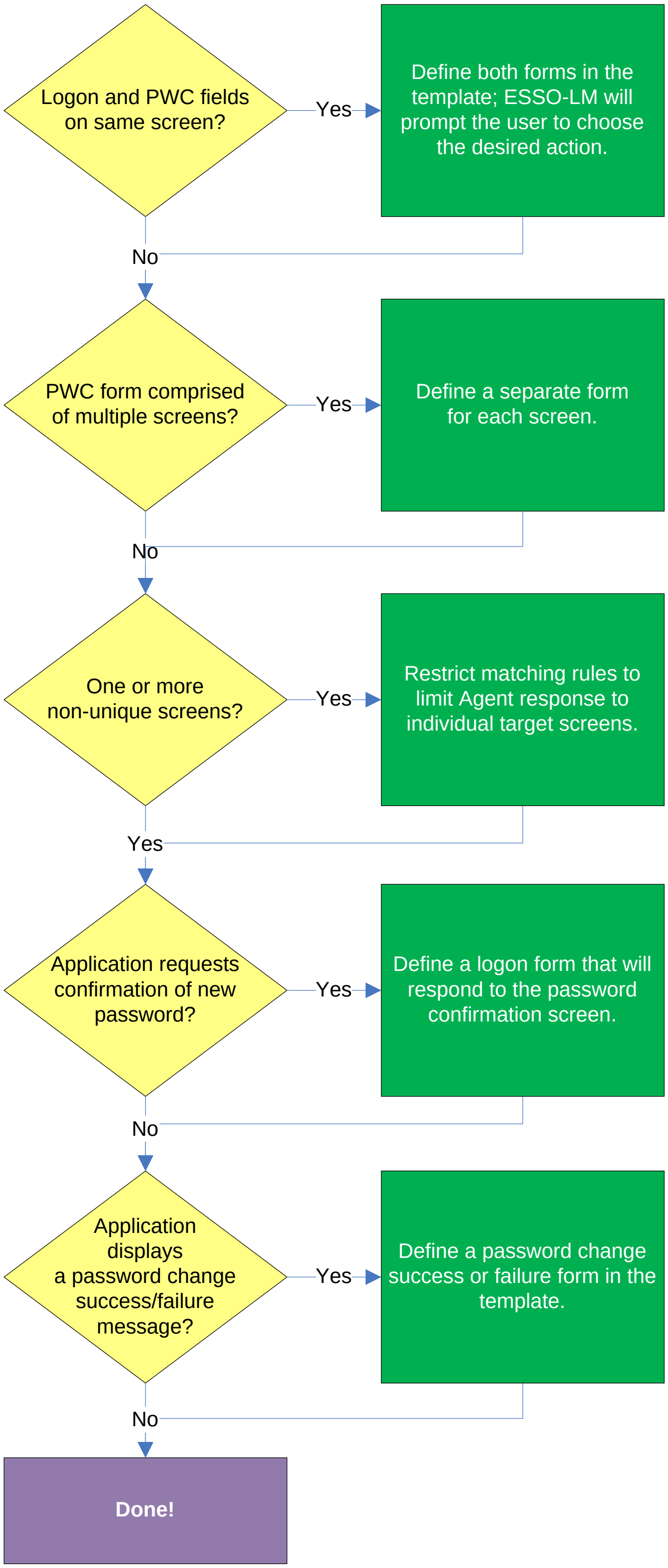
Since Logon Manager treats each new screen as a separate form, if the target form is comprised of multiple screens,( i.e., the user name field is displayed, then the screen is cleared and the password field is displayed on a brand new screen), define a separate form for each of the screens.

### Logon and password change (PWC) forms on the same screen?

Some applications display logon and password change fields and/or controls on the same screen. In such cases, 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. To allow the users to choose the desired 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.

## Determining the Optimal Configuration for a Password Change Form


Logon and PWC fields on same screen?
Yes
Define both forms in the template; ESSO-LM will prompt the user to choose the desired action.
No
PWC form comprised of multiple screens?
Yes
Define a separate form for each screen.
No
One or more non-unique screens?
Yes
Restrict matching rules to limit Agent response to individual target screens.
Yes
Application requests confirmation of new password?
Yes
Define a logon form that will respond to the password confirmation screen.
No
Application displays a password change success/failure message?
Yes
Define a password change success or failure form in the template.
No
Done!

### Logon and password change (PWC) forms on the same screen?

Some applications display logon and password change fields and/or controls on the same screen. In such cases, 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. To allow the users to choose the desired 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.

### PWC form comprised of multiple screens?

Since Logon Manager treats each new screen as a separate form, if the target form is comprised of multiple screens,( i.e., the application prompts the user to enter the old password, clears the screen, then prompts for the new password, clears the screen again, and then prompts for confirmation of the new password), define a separate form for each of the screens.

### One or more non-unique screens?

If one or more screens in a multiple-screen form are non-unique (i.e., Logon Manager cannot distinguish between them), response to those screens will be erroneous. When configuring the form, you must select a match text string within the form that does not appear anywhere else within the application; otherwise, Logon Manager may respond to screens that contain the non-unique match text even though they are not the target form. If a unique match text string is not present in the target form, investigate the possibility of modifying the application in order to add a piece of unique text to the form. If modifying the application is not an option, configure the "logon chooser" feature in the template and instruct users to select the desired credentials when logging on to the affected application.

### Application requests confirmation of new password?

If the application requests the users to confirm (re-enter) the new password during password change, define a logon form that will respond to the password confirmation screen (define only the password field in this secondary logon form).

### Application displays a password change success or failure message?

If the application displays a success or failure message after password change, define a password change success or a password change failure form in the template. For instructions, see Configuring a Form.

## 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.

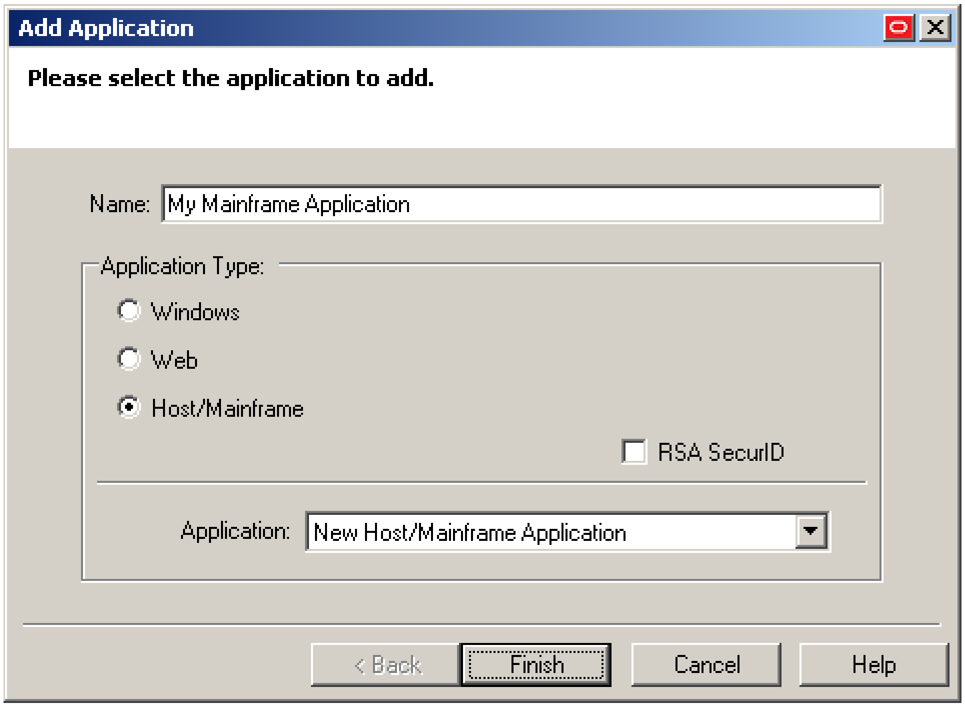
> **Note:** Before configuring a form, launch the desired terminal emulator, connect to the target application, and make sure its target form is displayed.

### Basic Configuration of a Fixed-Screen Form

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

1. Open the Logon Manager Administrative Console. By default, the shortcut is located in **Start → Programs → Oracle → Logon Manager 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 the logon 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.



Add Application

Please select the application to add.

Name: My Mainframe Application

Application Type:

○ Windows

○ Web

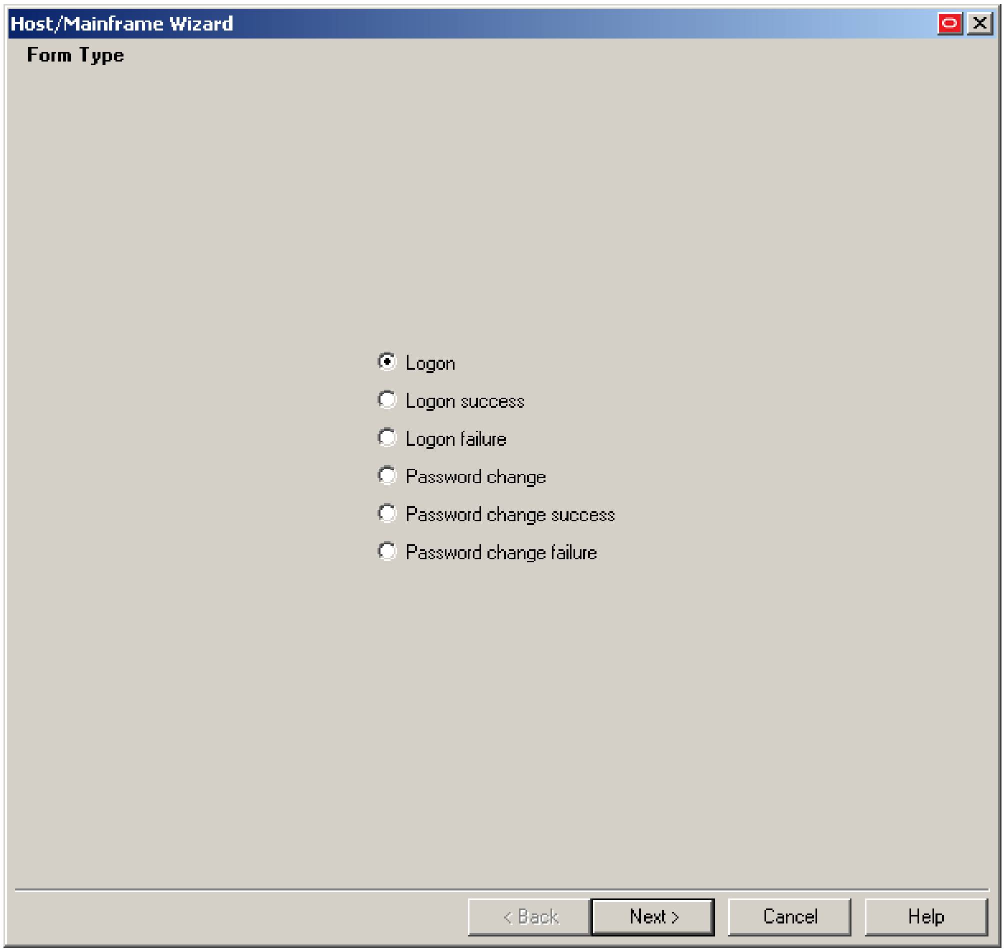
● Host/Mainframe

□ RSA SecurID

Application: New Host/Mainframe Application

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

3. In the wizard that appears, define the match text and fields that you want Logon Manager to interact with when logging on to the application:
    a. In the “Form Type” screen, select the desired form type and click **Next**.

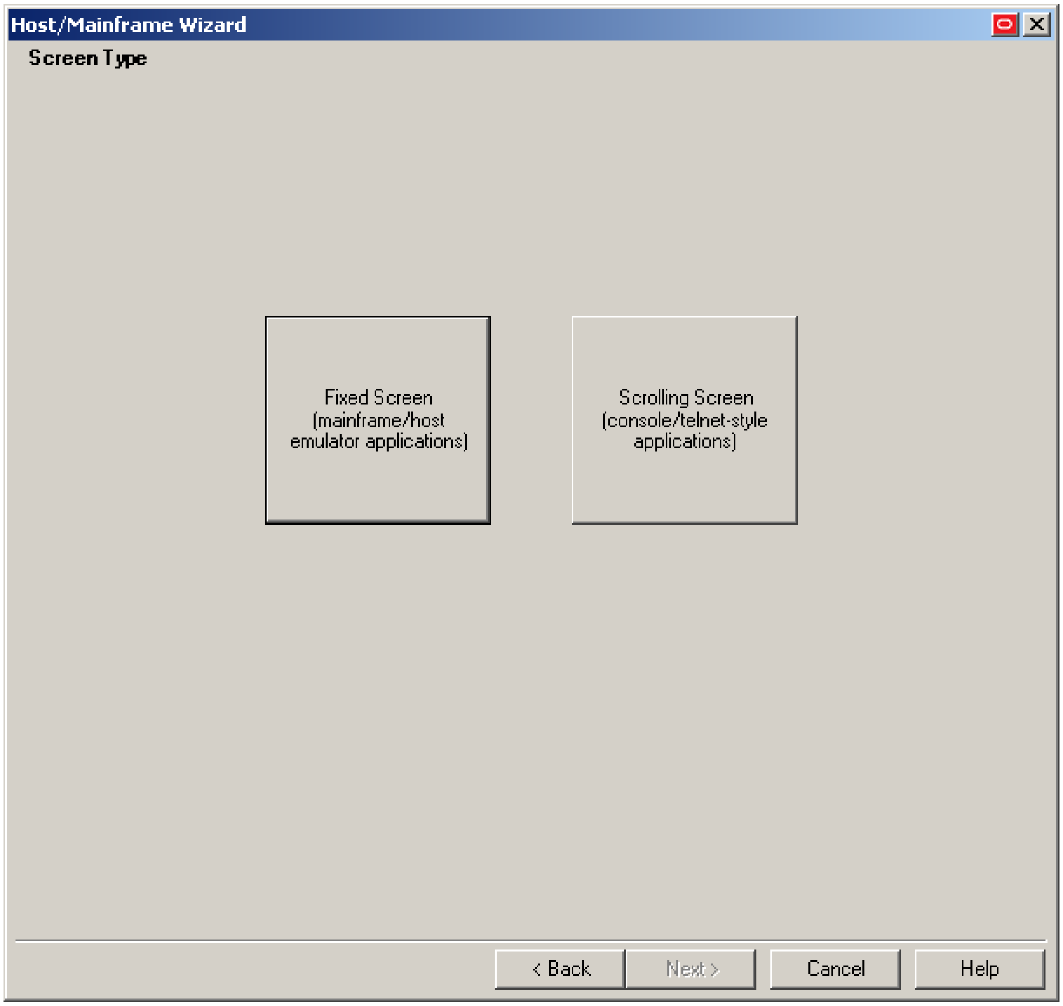


Host/Mainframe Wizard

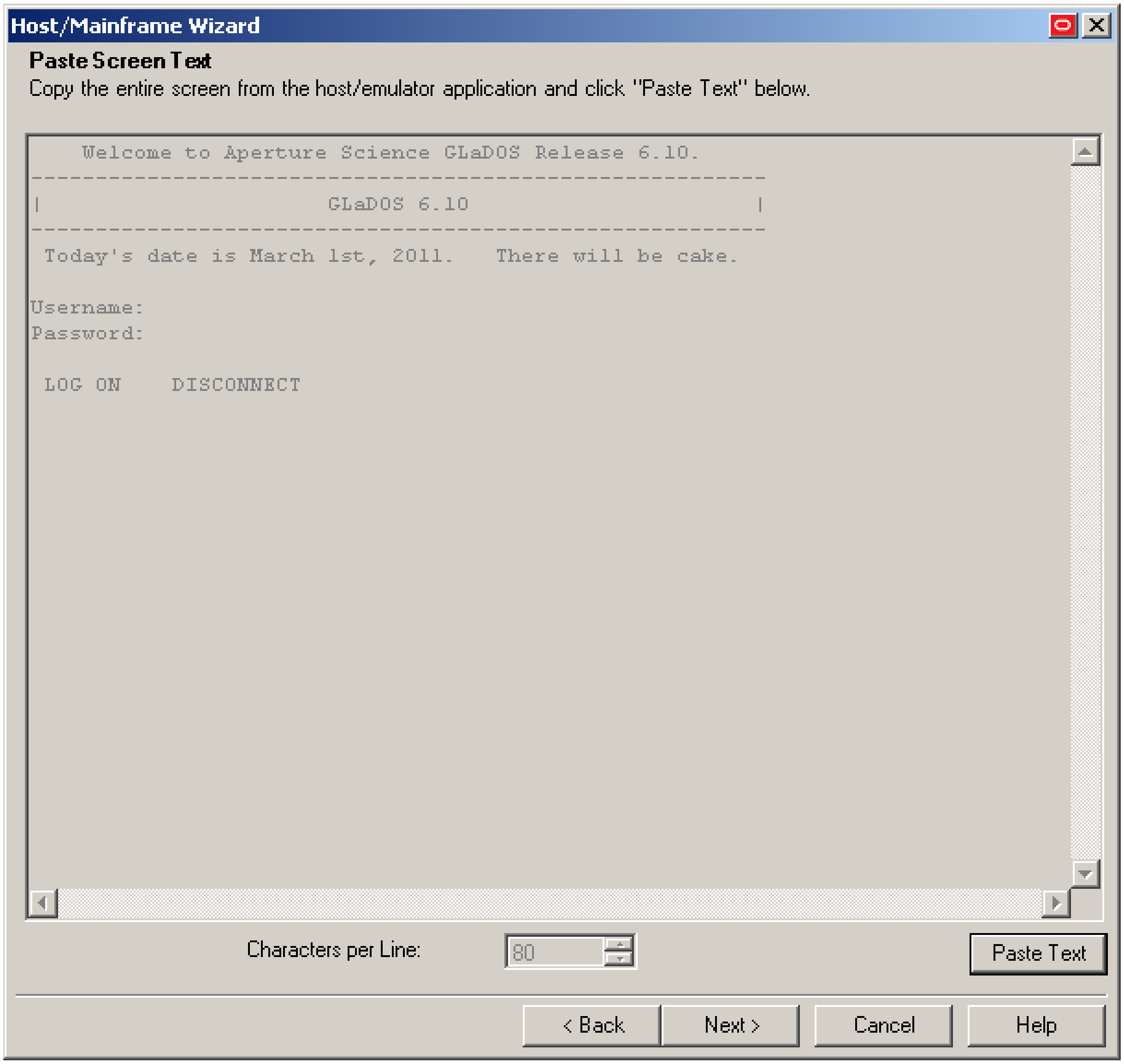
Form Type

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

b. In the “Screen Type” screen, click **Fixed Screen**.


Host/Mainframe Wizard
Screen Type
Fixed Screen (mainframe/host emulator applications)
Scrolling Screen (console/telnet-style applications)
< Back
Next >
Cancel
Help

c. When the “Paste Screen Text” screen appears, copy the entire text that comprises the target form in the emulator window to the clipboard and click **Paste Text** to paste it into the wizard. When you have pasted the text, click **Next**.



**Paste Screen Text**

Copy the entire screen from the host/emulator application and click "Paste Text" below.

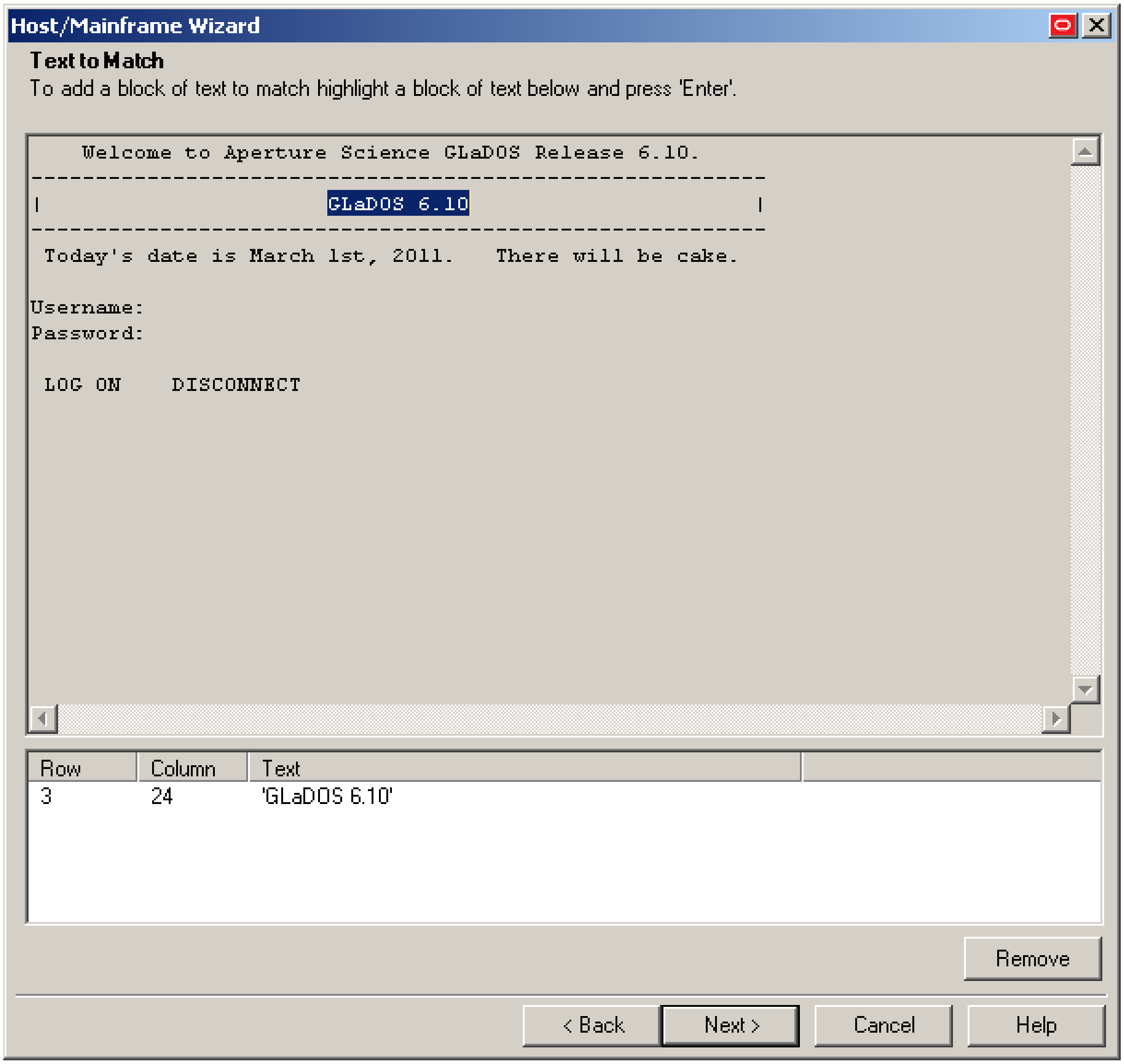
```
    Welcome to Aperture Science GLaDOS Release 6.10.
---------------------------------------------------------
|                     GLaDOS 6.10                       |
---------------------------------------------------------
 Today's date is March 1st, 2011.   There will be cake.

Username:
Password:

 LOG ON    DISCONNECT
```

Characters per Line: 80

d. In the “Text to Match” screen, define the text that Logon Manager will look for in the form to uniquely identify it and match it to the template:

> **Note:** Oracle recommends keeping your matching rules to a minimum while retaining the desired matching behavior. Matching on large portions of text can slow down the logon process.

i. Highlight the desired match text block.
ii. Press **Enter** to add the text matching rule.
iii. Repeat steps i-ii to define additional text matching rules.
iv. When you have defined the desired rules, click **Next**.



**Text to Match**

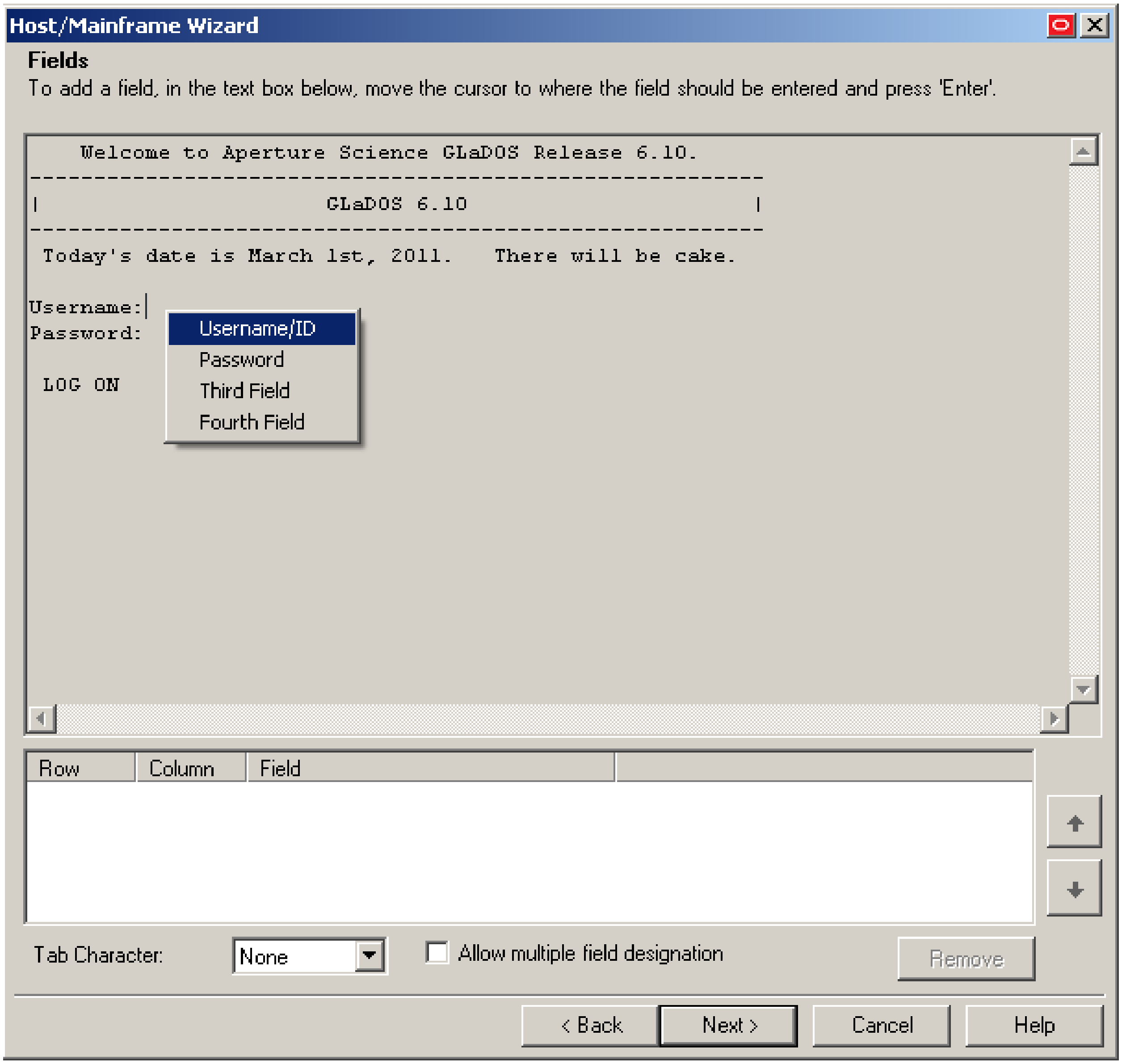
To add a block of text to match highlight a block of text below and press 'Enter'.

```
    Welcome to Aperture Science GLaDOS Release 6.10.
---------------------------------------------------------
|                       GLaDOS 6.10                      |
---------------------------------------------------------
 Today's date is March 1st, 2011.   There will be cake.

Username:
Password:

 LOG ON    DISCONNECT
```

| Row | Column | Text |
|---|---|---|
| 3 | 24 | 'GLaDOS 6.10' |

e. In the “Fields” screen, define the target fields into which Logon Manager will inject credentials:

   i. (Optional) If your form requires the injection of the same data into multiple fields (e.g., injecting the password into a **Password** and a **Confirm Password** field), enable the **Allow multiple field designation** option.
   ii. Place the cursor at the beginning of the field into which you want Logon Manager to inject a credential.
   iii. Press **Enter**.
   iv. In the context menu that appears, select the desired field type.
   v. Repeat steps i-iii to define additional fields.
   vi. When you have defined all desired fields, click **Next**.



Host/Mainframe Wizard

**Fields**

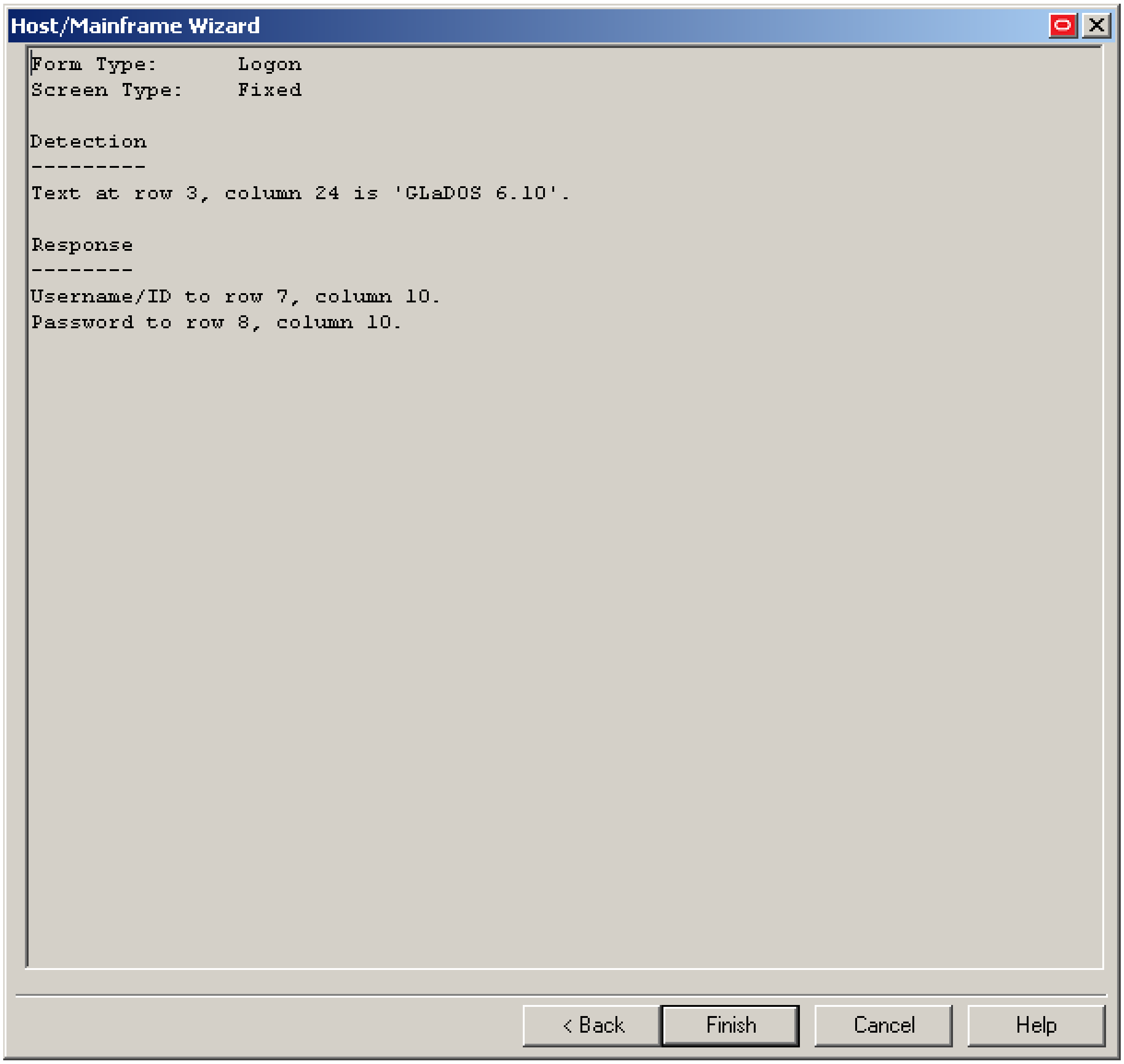
To add a field, in the text box below, move the cursor to where the field should be entered and press 'Enter'.

```
   Welcome to Aperture Science GLaDOS Release 6.10.
---------------------------------------------------------
|                      GLaDOS 6.10                      |
---------------------------------------------------------
 Today's date is March 1st, 2011.   There will be cake.

Username:
Password:

 LOG ON
```

Tab Character: None

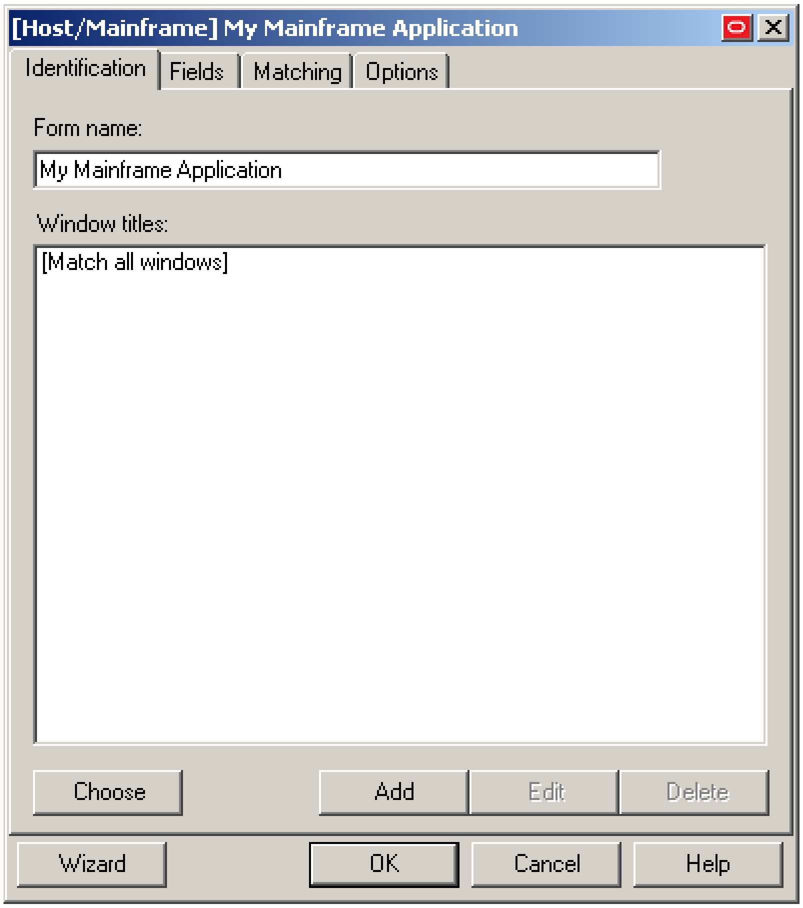
☐ Allow multiple field designation

f. In the summary screen that appears, review your configuration choices. If any of the choices seem erroneous, click **Back** and correct them; otherwise, click **Finish**.



```
Form Type:       Logon
Screen Type:     Fixed

Detection
---------
Text at row 3, column 24 is 'GLaDOS 6.10'.

Response
--------
Username/ID to row 7, column 10.
Password to row 8, column 10.
```

4. In the form properties dialog that appears, do one of the following:
    - If you need to perform additional configuration, do so now.
    - If you are satisfied with the configuration, click OK to dismiss the dialog.



[Host/Mainframe] My Mainframe Application

Form name: My Mainframe Application

Window titles: [Match all windows]

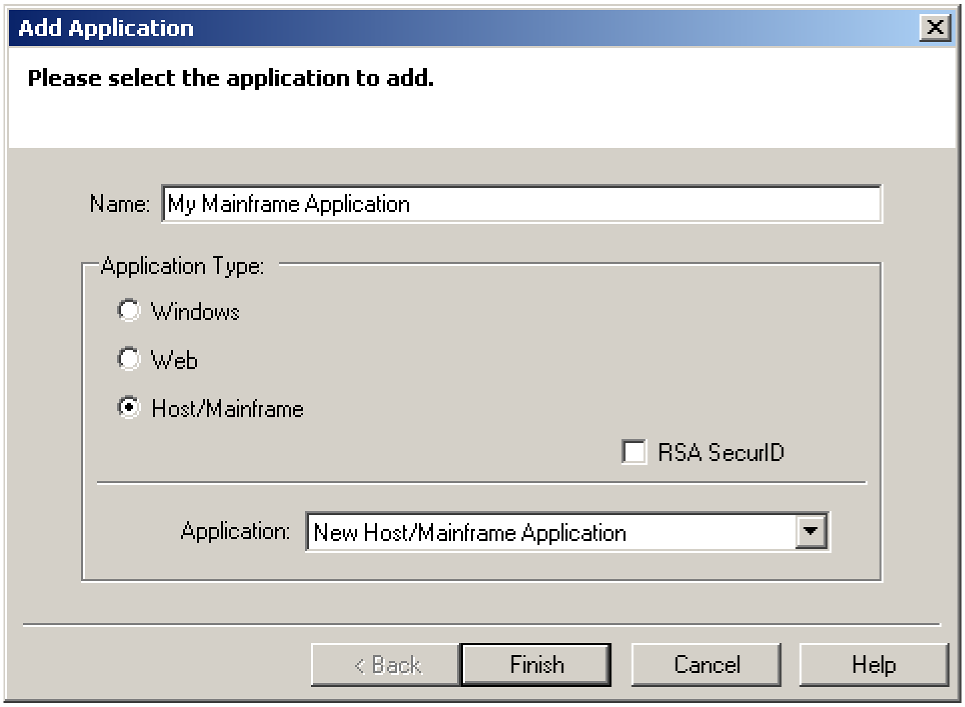
5. Publish your changes to the repository as described in Publishing a Template to the Repository, if applicable.

## Basic Configuration of a Scrolling-Screen Form

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

1. Open the Logon Manager Administrative Console. By default, the shortcut is located in **Start → Programs → Oracle → Logon Manager 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 the logon 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.



Name: My Mainframe Application

Application Type:

○ Windows

○ Web

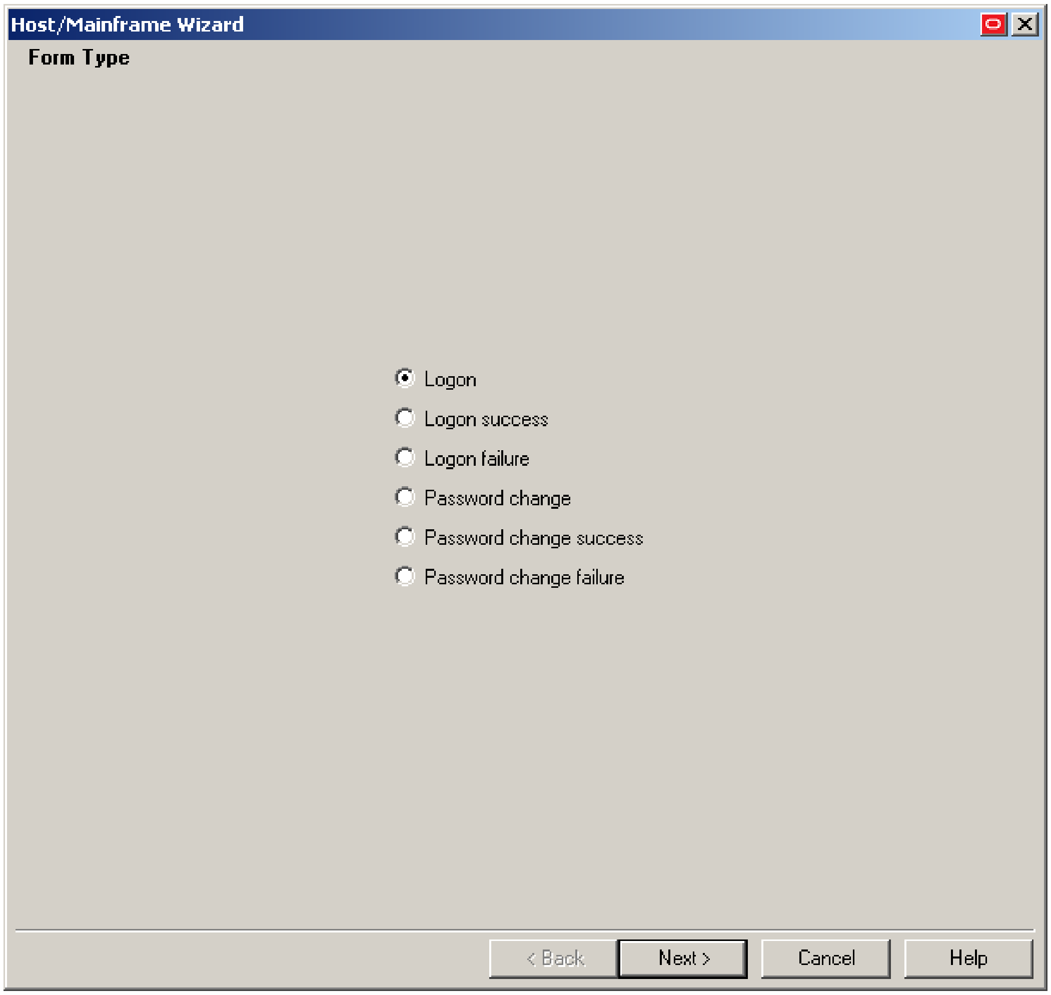
● Host/Mainframe

□ RSA SecurID

Application: New Host/Mainframe Application

   - If you want to add a logon 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 wizard that appears, define the match text and fields that you want Logon Manager to interact with when logging on to the application:
    a. In the “Form Type” screen, select the desired form type and click **Next**.

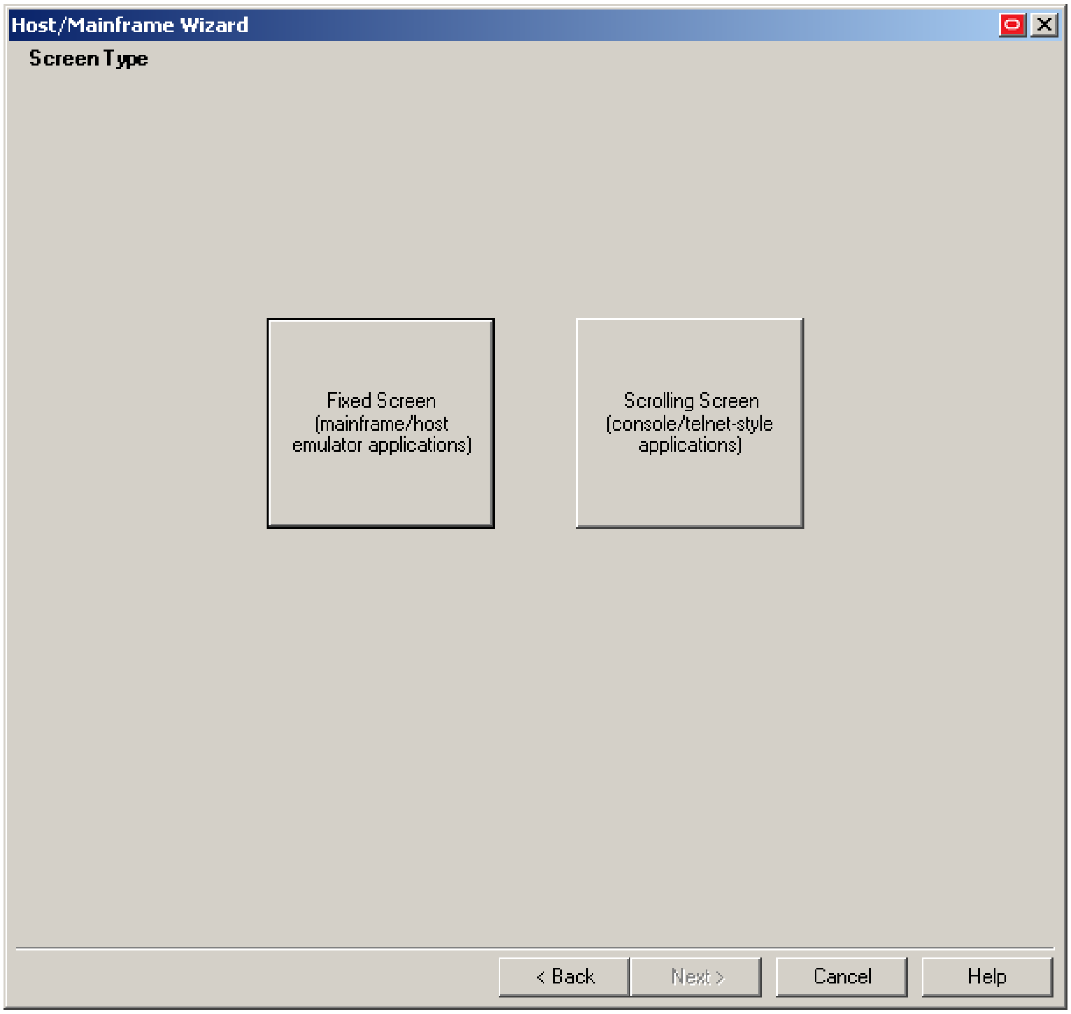


Host/Mainframe Wizard

Form Type

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

b. In the “Screen Type” screen, click **Scrolling Screen**.

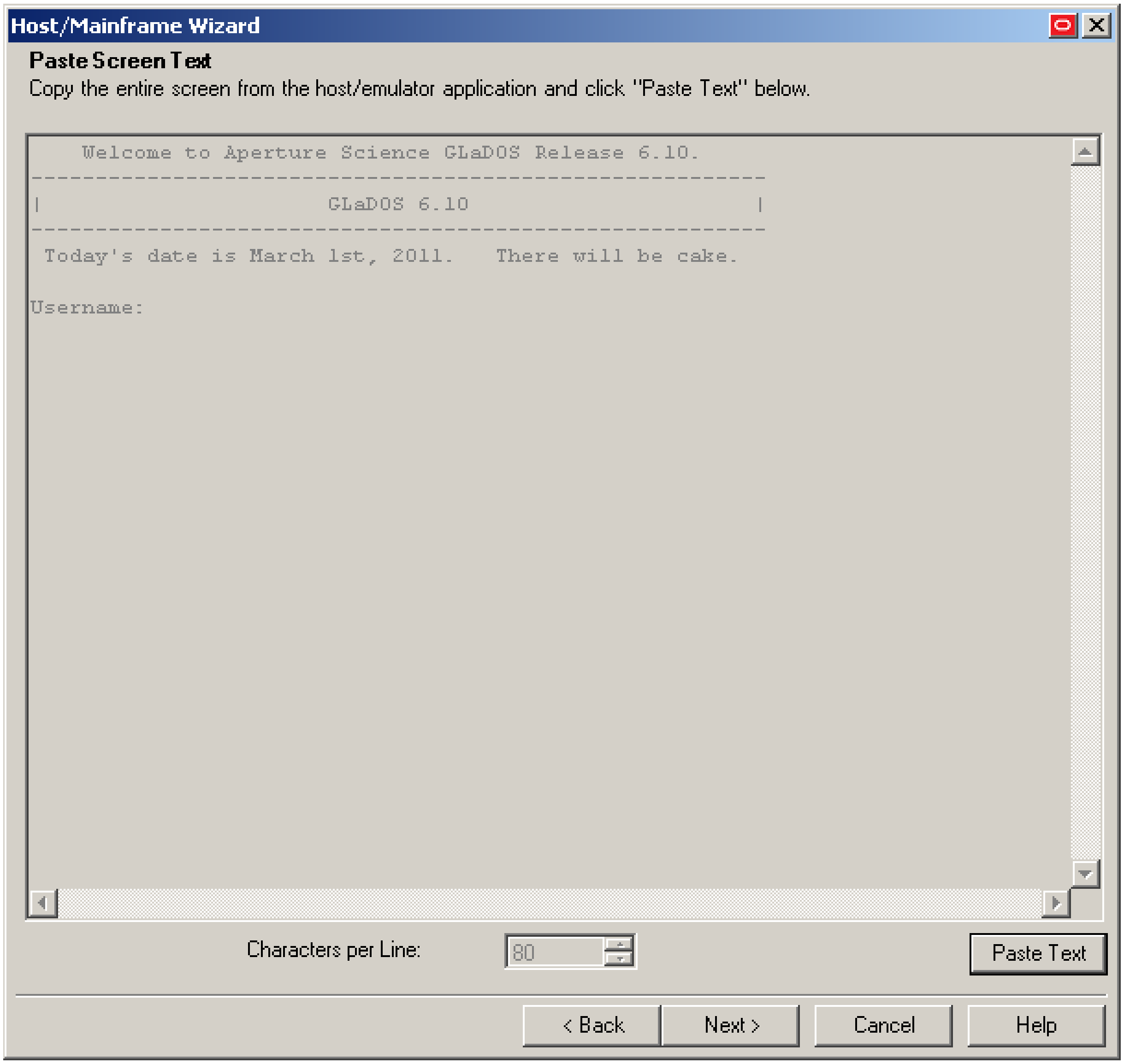


Host/Mainframe Wizard

Screen Type

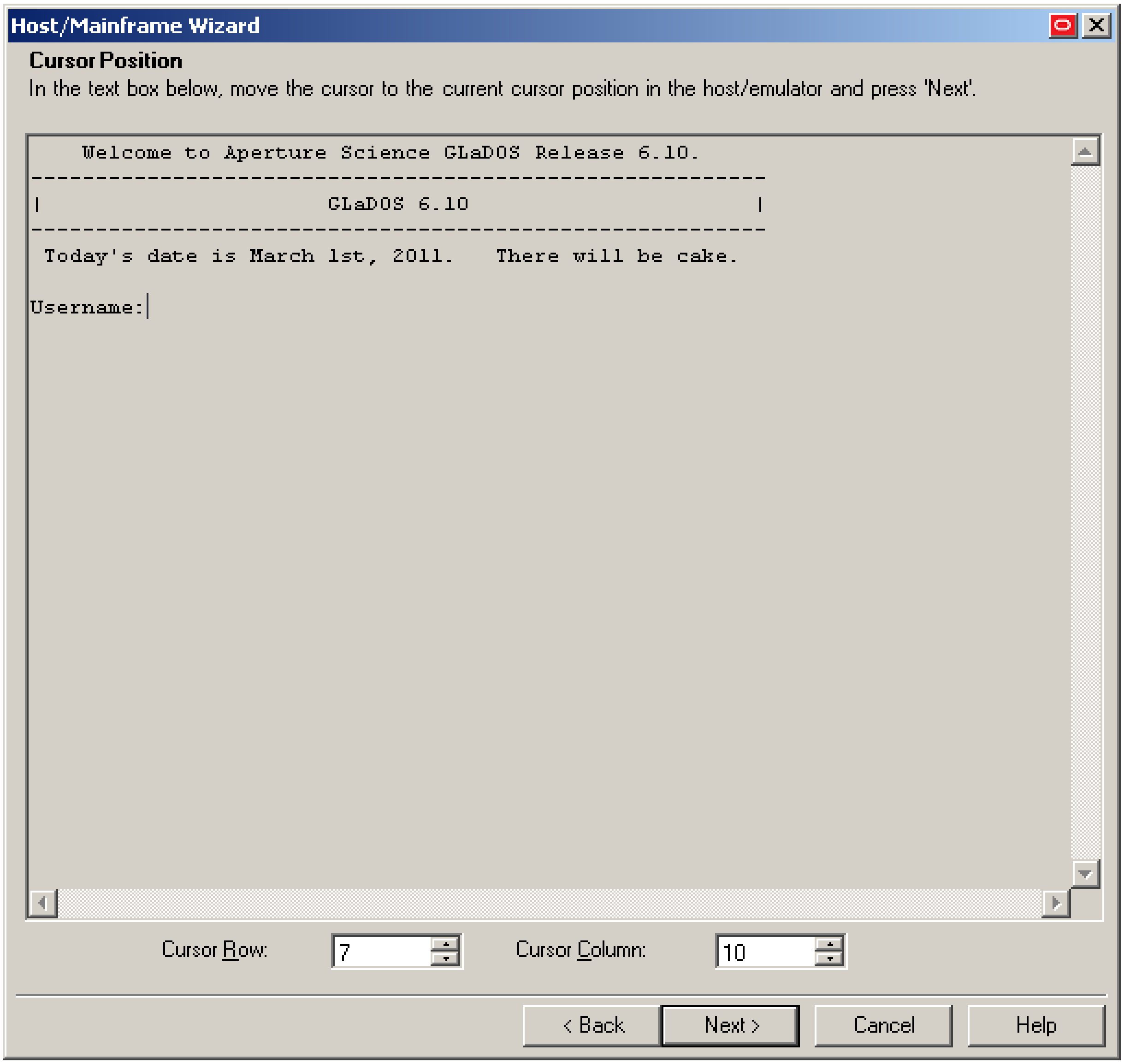
Fixed Screen (mainframe/host emulator applications)
Scrolling Screen (console/telnet-style applications)
< Back
Next >
Cancel
Help

c. When the “Paste Screen Text” screen appears, copy the entire text that comprises the target form in the emulator window to the clipboard and click **Paste Text** to paste it into the wizard. When you have pasted the text, click **Next**.



```
    Welcome to Aperture Science GLaDOS Release 6.10.
---------------------------------------------------------
|                      GLaDOS 6.10                      |
---------------------------------------------------------
 Today's date is March 1st, 2011.   There will be cake.

Username:
```

d. In the “Cursor Position” screen, specify the current position of the cursor in the emulator display as follows:
   i. Click the desired location in the screen text. (You can also enter the vertical and horizontal coordinates in the **Cursor Row** and **Cursor Column** fields.)
   ii. Click **Next**.



Host/Mainframe Wizard

Cursor Position

In the text box below, move the cursor to the current cursor position in the host/emulator and press 'Next'.

```
    Welcome to Aperture Science GLaDOS Release 6.10.
-----------------------------------------------------------
|                       GLaDOS 6.10                       |
-----------------------------------------------------------
 Today's date is March 1st, 2011.   There will be cake.

Username:|
```

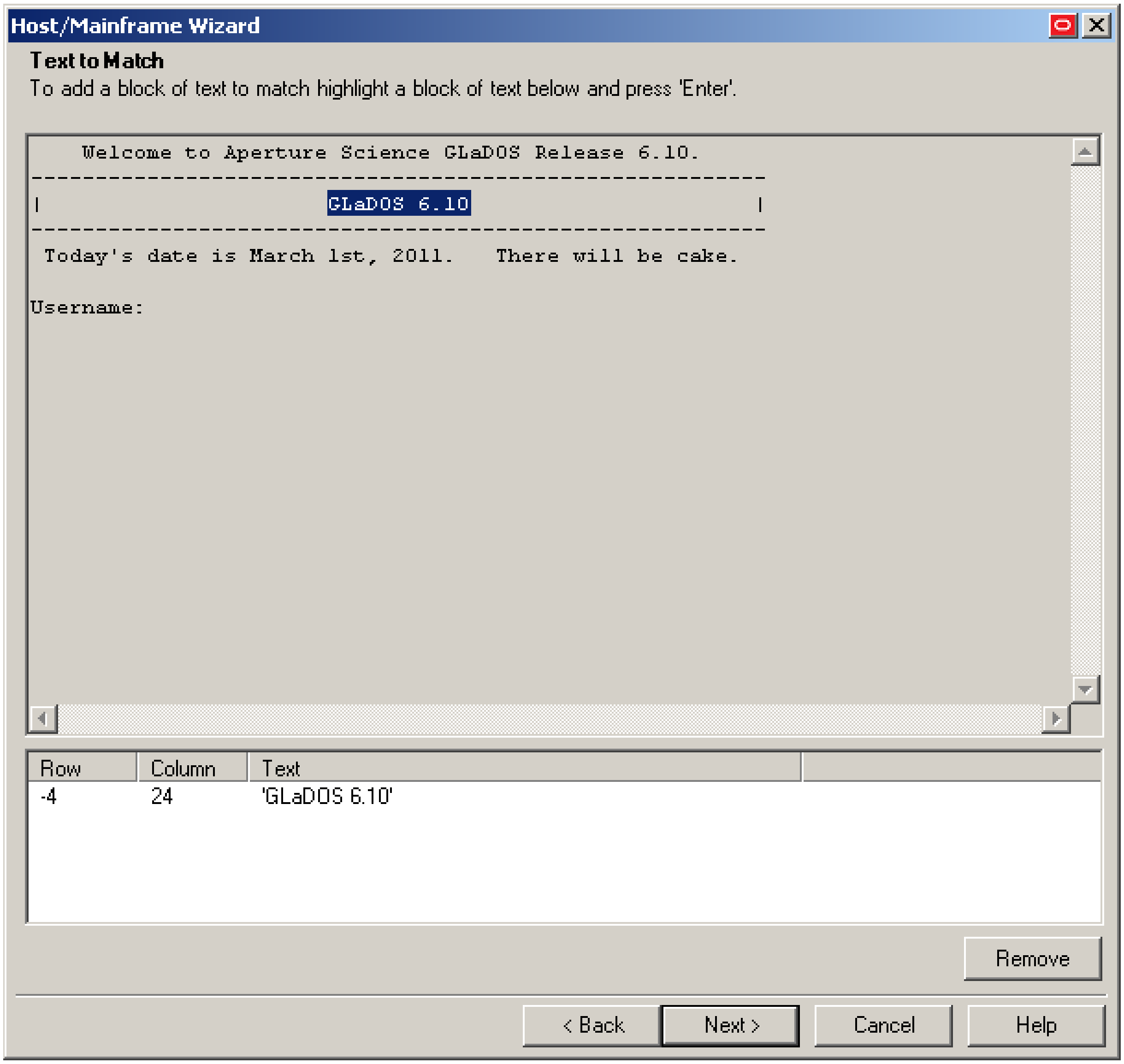
Cursor Row: 7

Cursor Column: 10

e. In the “Text to Match” screen, define the text that Logon Manager will look for in the form to uniquely identify it and match it to the template:

> **Note:** Oracle recommends keeping your matching rules to a minimum while retaining the desired matching behavior. Matching on large portions of text can slow down the logon process.

i. Highlight the desired match text block.
ii. Press **Enter** to add the text matching rule.
iii. Repeat steps i-ii to define additional text matching rules.
iv. When you have defined the desired rules, click **Next**.



Host/Mainframe Wizard

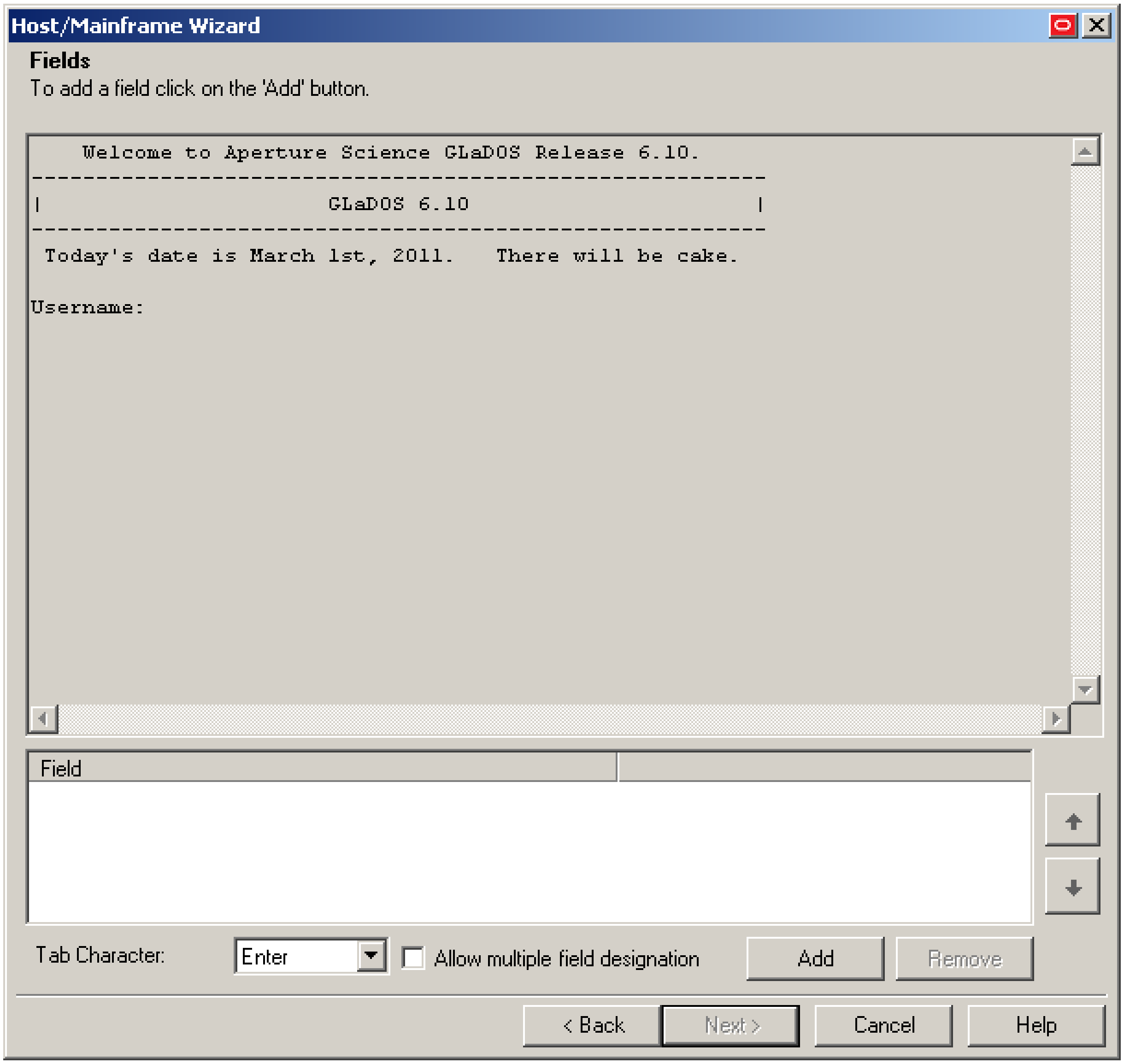
**Text to Match**

To add a block of text to match highlight a block of text below and press 'Enter'.

```
    Welcome to Aperture Science GLaDOS Release 6.10.
----------------------------------------------------------
|                       GLaDOS 6.10                      |
----------------------------------------------------------
 Today's date is March 1st, 2011.   There will be cake.

Username:
```

| Row | Column | Text |
|---|---|---|
| -4 | 24 | 'GLaDOS 6.10' |

f. In the “Fields” screen, define the target fields into which Logon Manager will inject credentials, in the order they appear in the application’s logon sequence:
   i. (Optional) If your form requires the injection of the same data into multiple fields (e.g., injecting the password into a **Password** and a **Confirm Password** field), enable the **Allow multiple field designation** option.
   ii. In the **Tab Character** field, select the keystroke Logon Manager will send to submit the data injected into the field to the application and advance to the next field.
   iii. Click **Add**.



Host/Mainframe Wizard

**Fields**

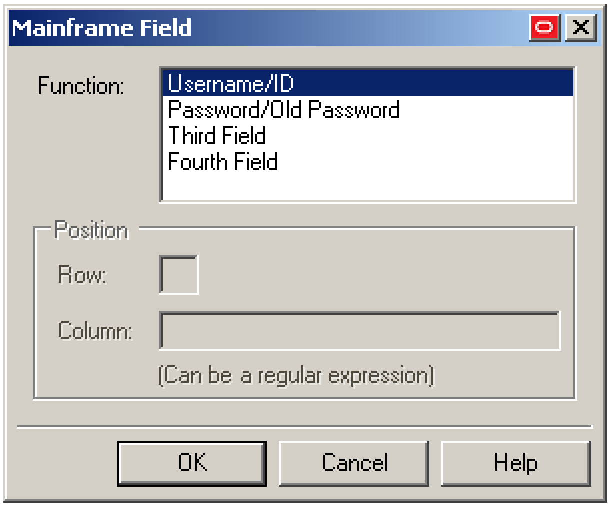
To add a field click on the 'Add' button.

```
    Welcome to Aperture Science GLaDOS Release 6.10.
---------------------------------------------------------
|                      GLaDOS 6.10                      |
---------------------------------------------------------
 Today's date is March 1st, 2011.   There will be cake.

Username:
```

Tab Character: Enter

☐ Allow multiple field designation

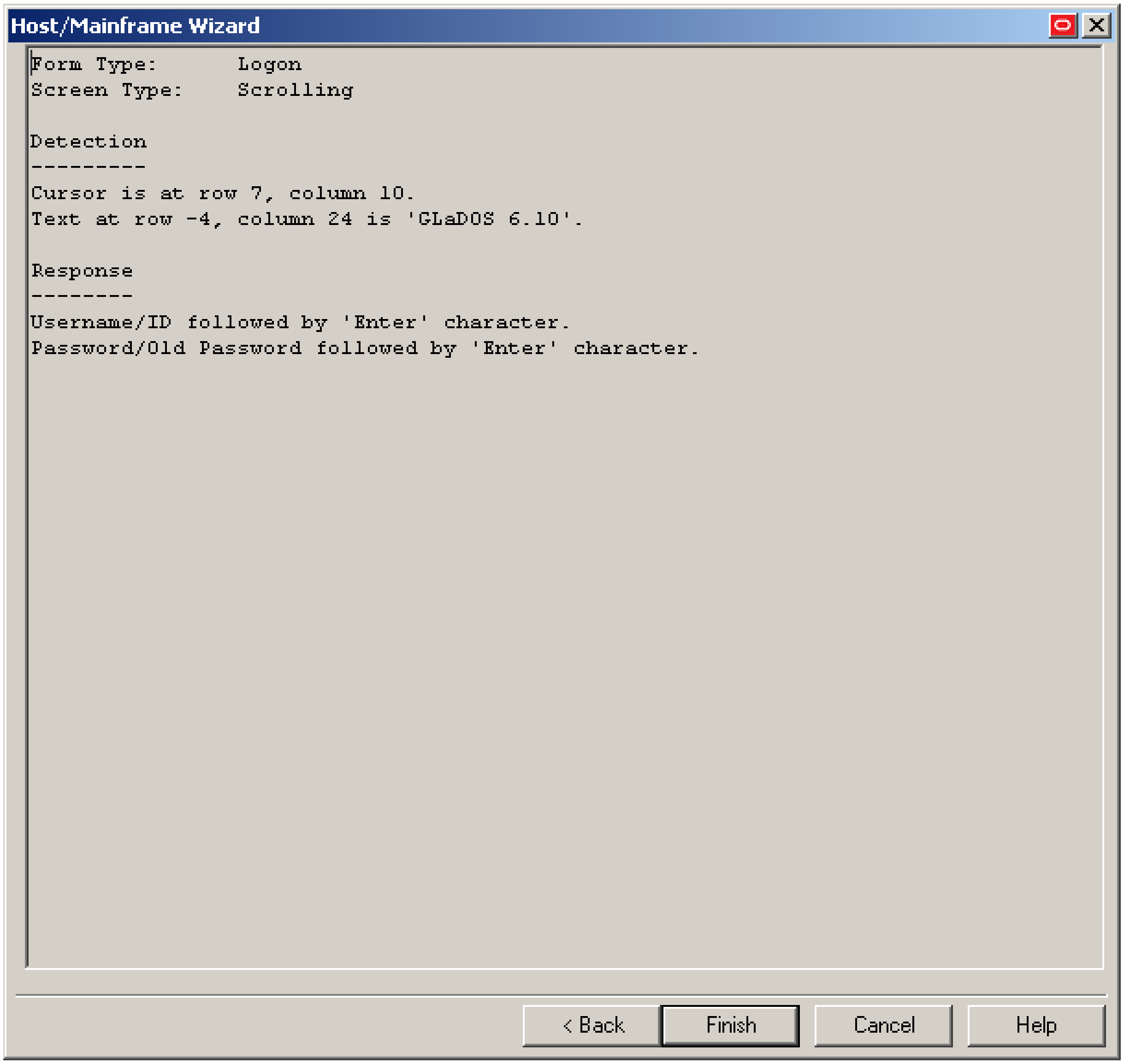
iv. In the dialog that appears, select the field type and click **OK**.



v. Repeat steps i-iii to define additional fields.

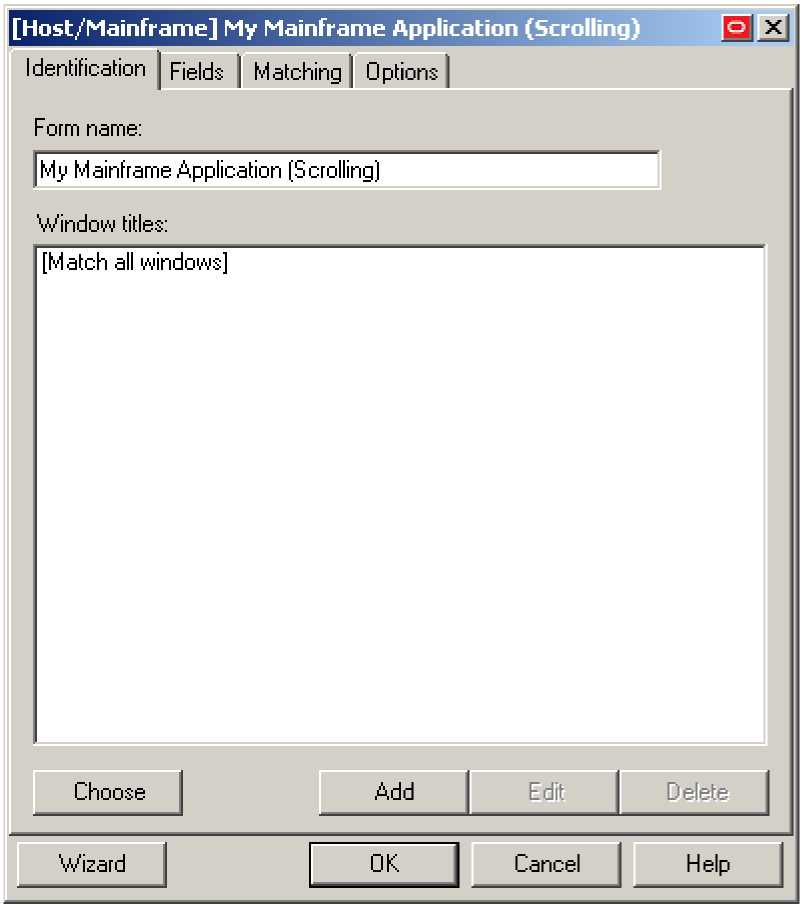
vi. When you have defined all desired fields, click **Next**.

4. In the summary screen that appears, review your configuration choices. If any of the choices seem erroneous, click **Back** and correct them; otherwise, click **Finish**.



```
Form Type:       Logon
Screen Type:     Scrolling

Detection
---------
Cursor is at row 7, column 10.
Text at row -4, column 24 is 'GLaDOS 6.10'.

Response
--------
Username/ID followed by 'Enter' character.
Password/Old Password followed by 'Enter' character.
```

5. In the form properties dialog that appears, do one of the following:
    - If you need to perform additional configuration, see Advanced Form Configuration.
    - If you are satisfied with the configuration, click **OK** to dismiss the dialog.



Form name: My Mainframe Application (Scrolling)

Window titles: [Match all windows]

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

## Advanced Form Configuration

This section describes the additional configuration you can perform after you have defined a form using the wizard. These are:

- Defining Window Title Matching Rules
- Defining Additional Text Matching Rules
- Defining Additional Fields
- Adding Keystrokes, Pauses, or Text to the Logon Sequence
- Configuring a Form for Dynamic Column Positioning (Scrolling-Screen Only)
- Adjusting Injection Timing
- Adjusting the Emulator Polling Interval
- Adjusting the Credential Request Delay Interval

Before starting the procedures below, open the properties dialog for the desired form as follows:

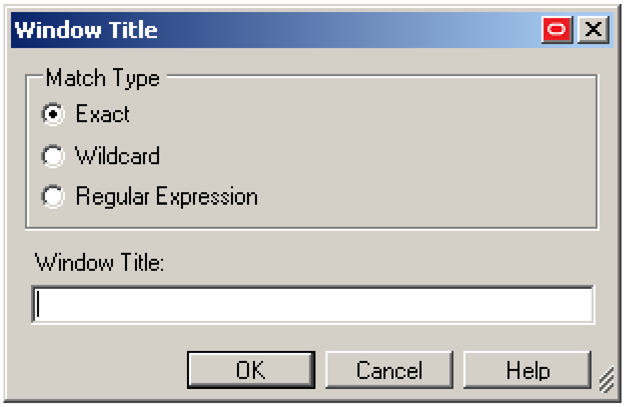
1. In the Logon Manager Administrative Console's left-hand tree, expand the **Applications** pane and select the desired template.
2. In the right-hand pane, select the **General** tab.
3. In the **General** tab, double-click the desired form definition.
4. Continue to the desired procedure.

### Defining Window Title Matching Rules

To limit Agent response to a specific window title (or multiple variations of thereof), define one or more specific window title matching rules as follows:

> **Note:** Oracle recommends keeping your matching rules to a minimum while retaining the desired matching behavior. Matching on large portions of text can slow down the logon process.

1. In the **Identification** tab of the form properties dialog, click **Add**.
2. In the "Window Title" dialog that appears, do the following:
   a. Select the desired matching type.
   b. Enter the desired match value.
   c. Click **OK** to save your changes.



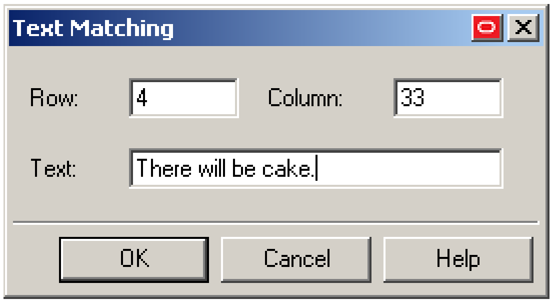
3. When you have finished, click **OK** to dismiss the form properties dialog.

### Defining Additional Text Matching Rules

To define additional text matching rules, do the following:

> **Note:** Oracle recommends keeping your matching rules to a minimum while retaining the desired matching behavior. Matching on large portions of text can slow down the logon process.

1. In the **Matching** tab of the form properties dialog, click **Add**.
2. In the dialog that appears, enter the desired match text and its starting coordinates.



Row: 4 Column: 33

Text: There will be cake.

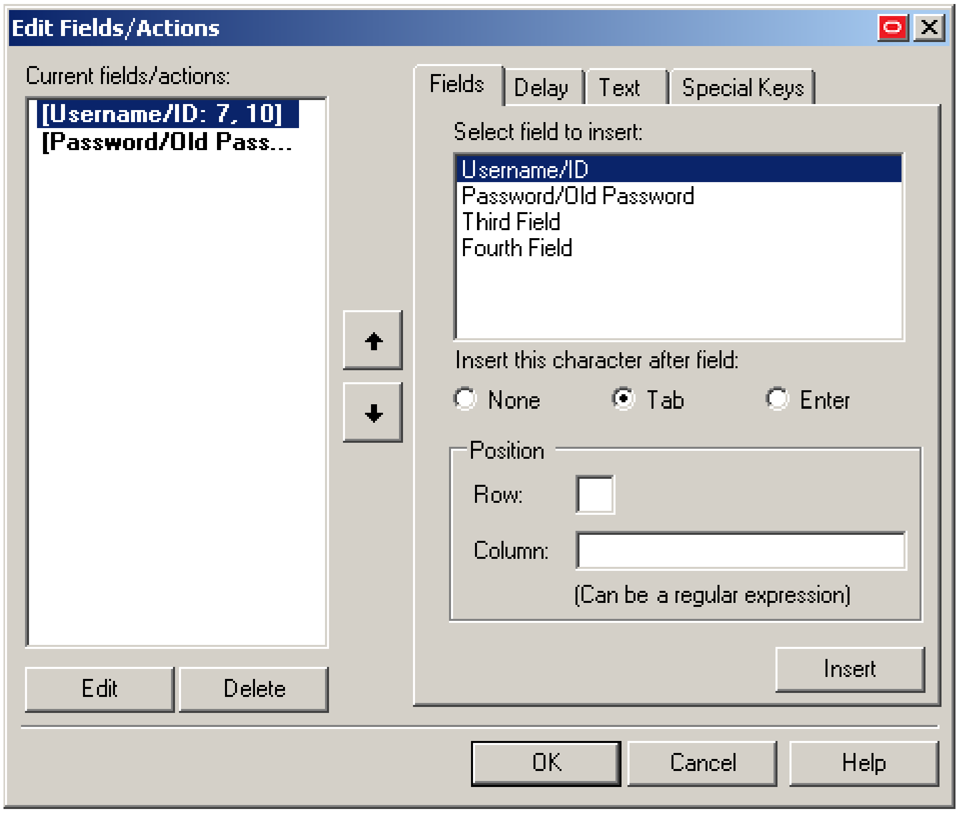
3. Repeat steps 1-2 for each additional rule you want to define.
4. When you have finished, click **OK** to dismiss the form properties dialog and publish your changes to the repository if applicable.

### Defining Additional Fields

To define fields additional to the ones you have configured in the wizard, do the following:

**Warning:** If you begin this procedure, the template will be converted from a native HLLAPI template to a SendKeys template – in other words, Logon Manager will no longer inject credentials programmatically via HLLAPI, but instead will emulate user input by sending keystrokes to the emulator. This conversion is not reversible – to revert to native HLLAPI injection, you must delete the template and re-create it. For more information on native HLLAPI vs. SendKeys-based injection, see Supported Credential Injection Methods.

1. In the **Fields** tab of the form properties dialog, click **Edit**.
2. In the dialog that appears, select the **Fields** tab.



Edit Fields/Actions

Current fields/actions:

[Username/ID: 7, 10]
[Password/Old Pass...

Select field to insert:

Username/ID
Password/Old Password
Third Field
Fourth Field

Insert this character after field:

○ None ● Tab ○ Enter

Position

Row:

Column:

(Can be a regular expression)

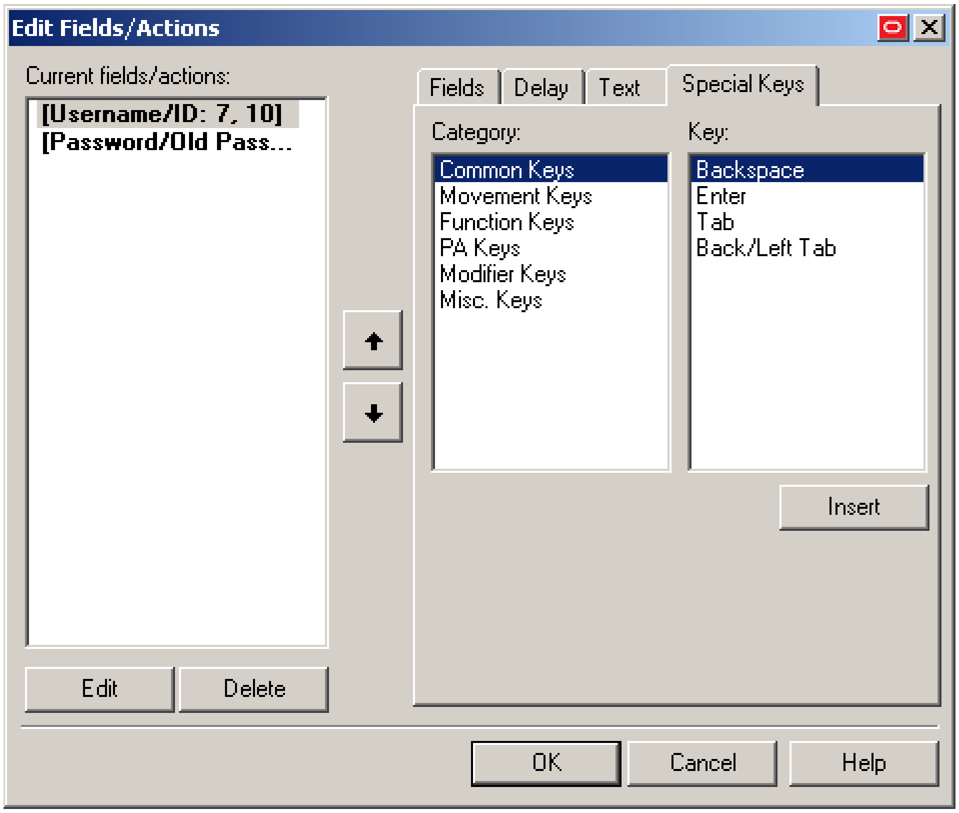
3. In the tab, select the desired field type, and enter its coordinates (fixed-screen only).
4. Use the up/down arrows to adjust the order of the logon sequence.
5. Repeat steps 2-3 for each additional field.
6. When you have finished, click **OK** to dismiss the form properties dialog and publish your changes to the repository, if applicable.

### Adding Keystrokes, Pauses, or Text to the Logon Sequence

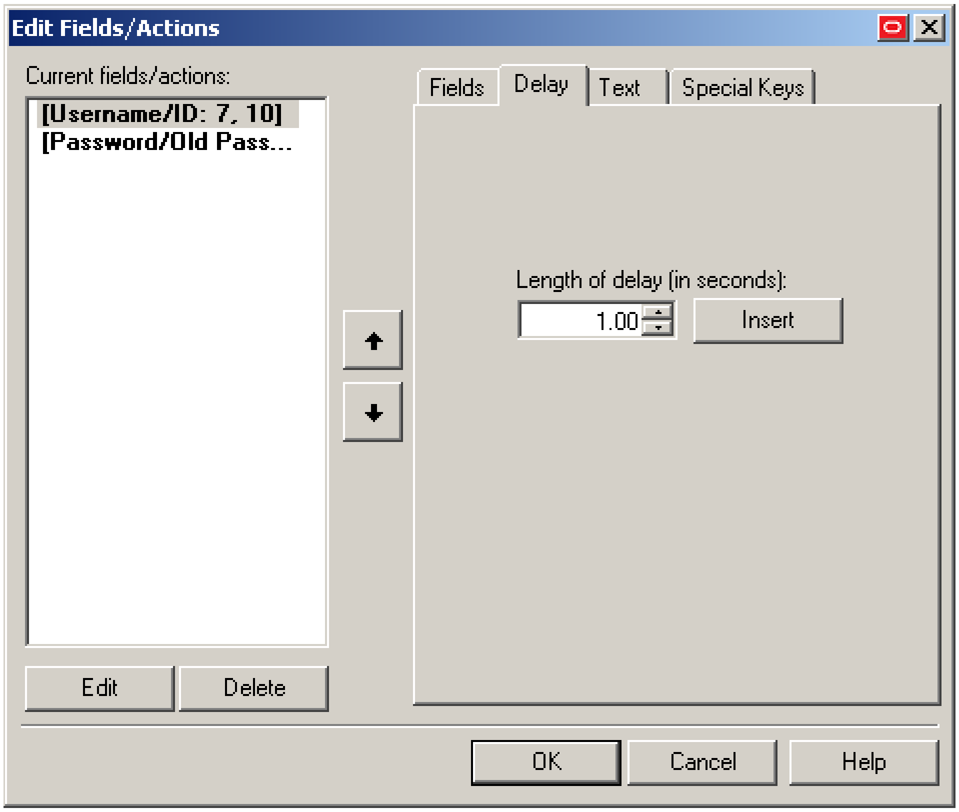
To add keystrokes, pauses, or text to the logon sequence, do the following:

**Warning:** If you begin this procedure, the template will be converted from a native HLLAPI template to a SendKeys template – in other words, Logon Manager will no longer inject credentials programmatically via HLLAPI, but instead will emulate user input by sending keystrokes to the emulator. This conversion is not reversible – to revert to native HLLAPI injection, you must delete the template and re-create it. For more information on native HLLAPI vs. SendKeys-based injection, see Supported Credential Injection Methods.

1. In the **General** tab of the form properties dialog, click **Edit** in the "Fields" section.
2. In the dialog that appears, do one of the following:
   - To add a keystroke:
     a. Select the **Special Keys** tab.
     b. In the tab, select the keystroke category and then the keystroke itself.
     c. Click **Insert**.



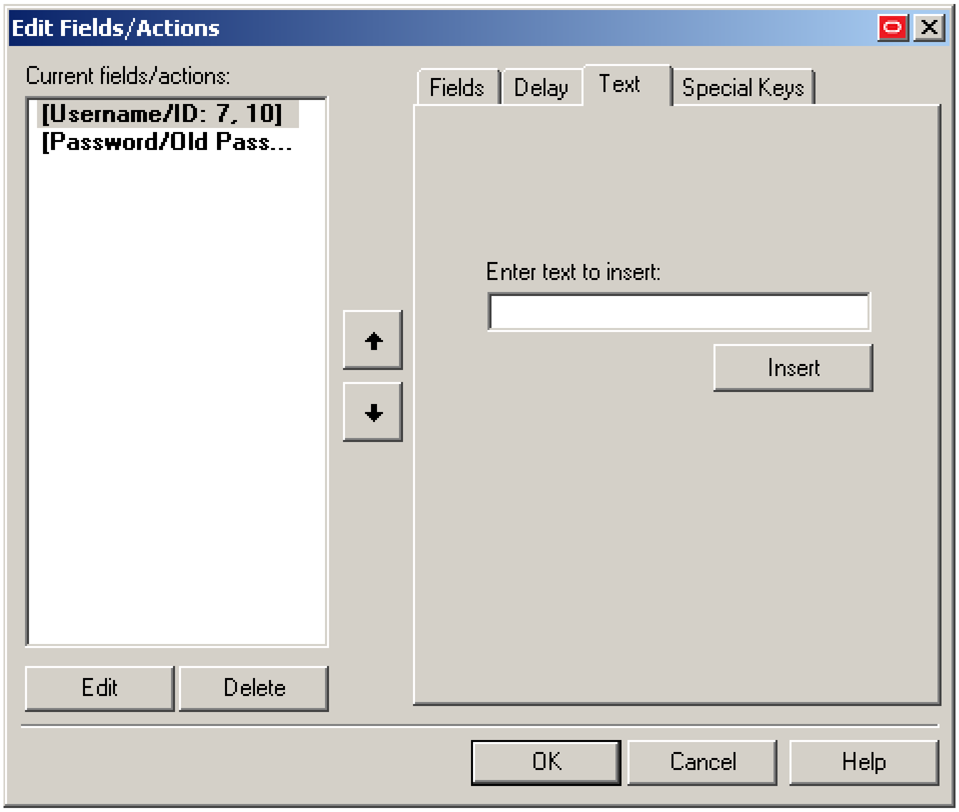
- To add a pause:
  a. Select the **Delay** tab.
  b. In the tab, specify the length of the pause in seconds.
  c. Click **Insert**.



Current fields/actions:
[Username/ID: 7, 10]
[Password/Old Pass...

Length of delay (in seconds): 1.00

- To add text:
    a. Select the **Text** tab.
    b. In the tab, enter the desired text string.
    c. Click **Insert**.

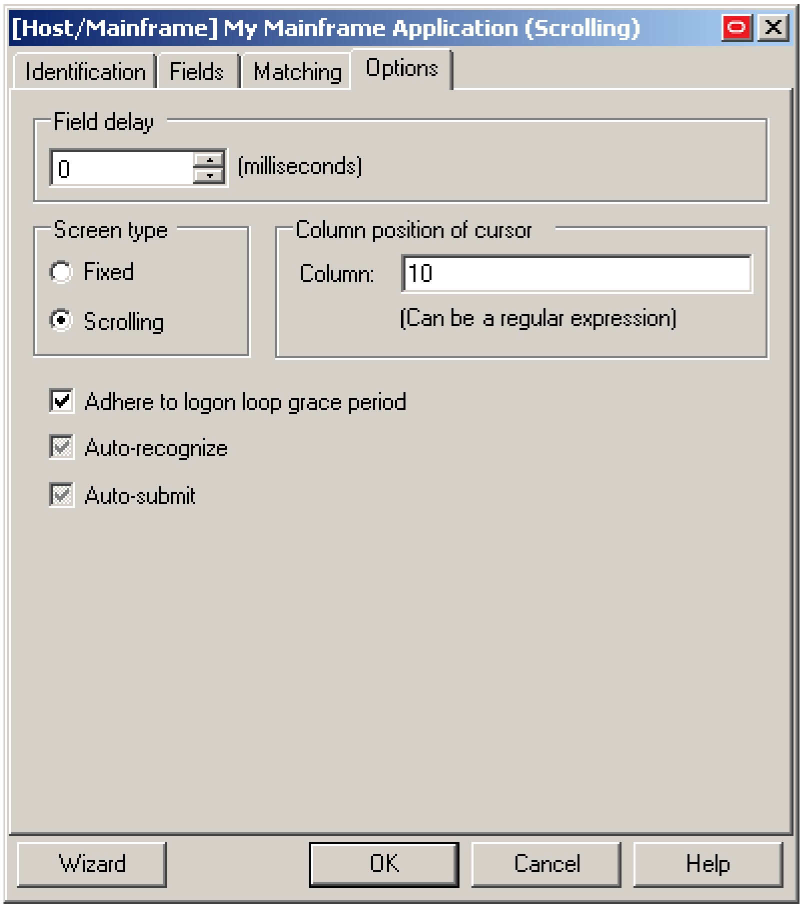


3. Select the newly inserted action(s) and use the up/down arrows to adjust their order in the logon sequence.
4. Repeat steps 2-3 for each additional field.
5. When you have finished, click **OK** to dismiss the form properties dialog and publish your changes to the repository, if applicable.

### Configuring a Form for Dynamic Column Positioning (Scrolling-Screen Only)

To configure a scrolling-screen form for dynamic column positioning, do the following:

1. Select the **Options** tab of the form properties dialog.
2. In the **Column position of cursor** field, enter a number or regular expression that defines the possible horizontal position(s) of the cursor in the form.



[Host/Mainframe] My Mainframe Application (Scrolling)

Field delay

0 (milliseconds)

Screen type

○ Fixed

● Scrolling

Column position of cursor

Column: 10

(Can be a regular expression)

☑ Adhere to logon loop grace period

☑ Auto-recognize

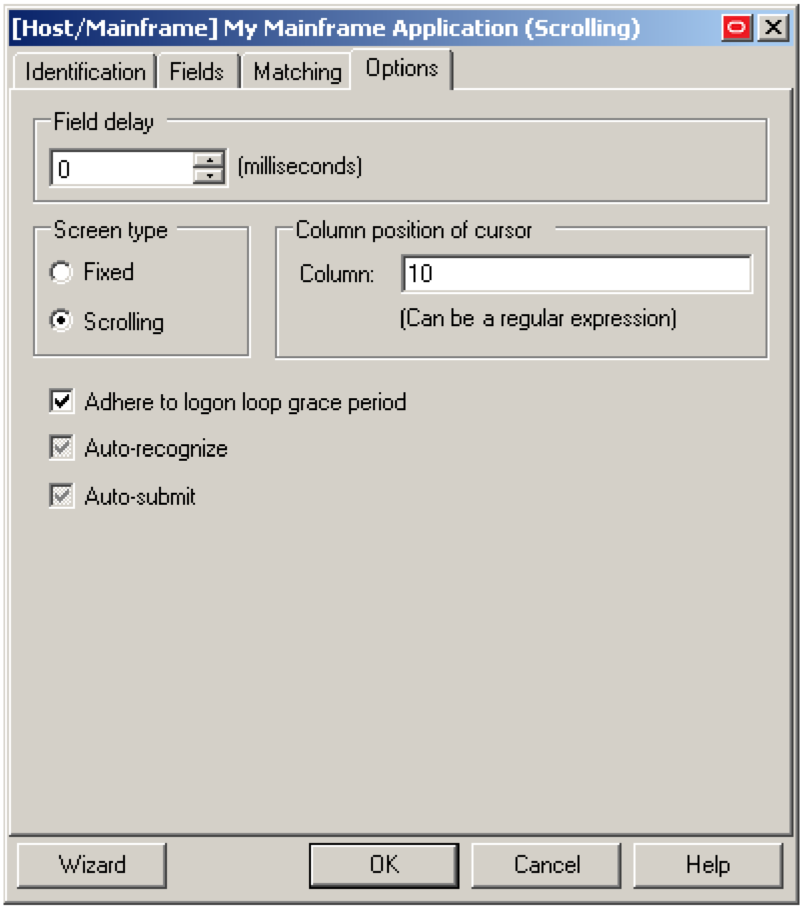
☑ Auto-submit

3. When you have finished, click **OK** to dismiss the form properties dialog and publish your changes to the repository, if applicable.

### Reducing Injection Speed

By default, Logon Manager injects and submits credentials with no delay between those actions. To reduce the injection speed, do the following:

1. Select the **Options** tab of the form properties dialog.
2. In the **Field Delay** field, enter the desired amount of time Logon Manager should wait after populating each field, in milliseconds.



[Host/Mainframe] My Mainframe Application (Scrolling)

Field delay: 0 (milliseconds)

Screen type: ○ Fixed ● Scrolling

Column position of cursor: Column: 10 (Can be a regular expression)

☑ Adhere to logon loop grace period

☑ Auto-recognize

☑ Auto-submit

3. When you have finished, click **OK** to dismiss the form properties dialog and publish your changes to the repository, if applicable.

### Adjusting the Emulator Polling Interval

By default, if polling is enabled, Logon Manager queries the emulator via HLLAPI for new events every 700 ms. To adjust the length of this polling interval, do the following:

1. Open the Logon Manager Administrative Console and load the current set of your global Agent settings.
2. In the left-hand tree, expand **Global Agent Settings → <current settings set> → End-User Experience → Response → Host Mainframe Apps**.
3. In the right-hand pane, select the check box next to the **Polling Interval** option and enter the desired interval length, in milliseconds, into the field.
4. When you have finished, publish your changes to the repository, if applicable.

### Adjusting the Credential Request Delay Interval

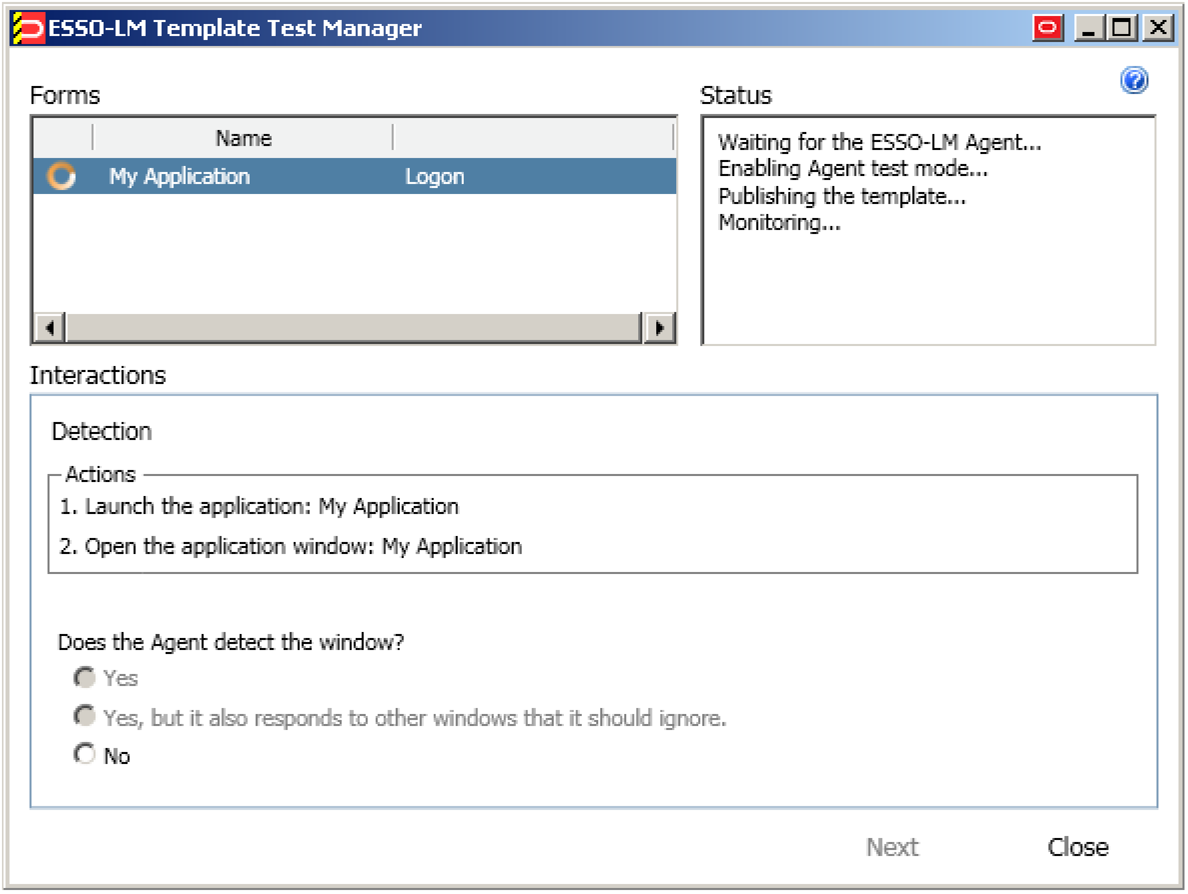
When a user logs on to a mainframe application for which a template exists but no credentials have yet been stored, Logon Manager prompts the user to store the credentials. If the user dismisses the dialog without storing the credentials, Logon Manager will wait for a default of 60 seconds and the next time the user interacts with the application, Logon Manager will prompt the user to store the credentials again. To change how long Logon Manager waits before re-prompting the user to store credentials for the application, do the following:

1. Open the Logon Manager Administrative Console and load the current set of your global Agent settings.
2. In the left-hand tree, expand **Global Agent Settings → <current settings set> → End-User Experience → Response → Host Mainframe Apps**.
3. In the right-hand pane, select the check box next to the **Credential Request Interval** option and enter the desired interval length, in milliseconds, into the field.
4. When you have finished, publish your changes to the repository, if applicable.

## Testing the Configuration of a Form

Once you have created and configured a form, complete the following steps to test it before publishing:

1. In the left-hand tree, expand the **Applications** node and navigate to the target template.
2. Right-click the target template and select **Test** from the context menu.
3. In the "Logon Manager Template Test Manager" window that appears, do the following:
   d. In the **Forms** pane, select the target form.
   e. Follow the instructions displayed in the **Interactions** pane.



ESSO-LM Template Test Manager

Forms

| Name | |
|---|---|
| My Application | Logon |

Status

Waiting for the ESSO-LM Agent...
Enabling Agent test mode...
Publishing the template...
Monitoring...

Interactions

Detection

Actions
1. Launch the application: My Application
2. Open the application window: My Application

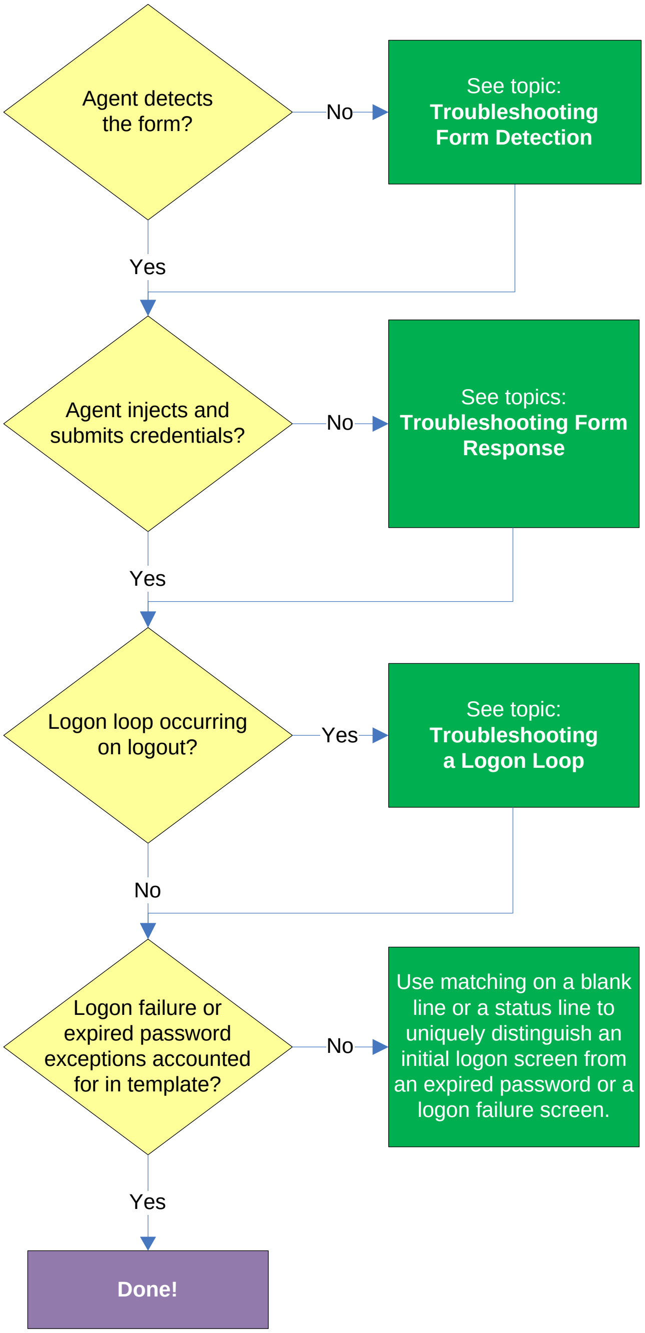
Does the Agent detect the window?

○ Yes
○ Yes, but it also responds to other windows that it should ignore.
○ No

f. Do one of the following:

- If the Agent responds to the application as desired and the test has completed successfully, click **Finish**.
- If the Agent is not responding to the application as desired, click **Close** and follow the troubleshooting flowcharts in this section to determine and correct the problem, then repeat steps 1-3 to test the corrected configuration.

## Testing the Configuration of a Logon Form


Agent detects the form?
No
See topic: Troubleshooting Form Detection
Yes
Agent injects and submits credentials?
No
See topics: Troubleshooting Form Response
Yes
Logon loop occurring on logout?
Yes
See topic: Troubleshooting a Logon Loop
No
Logon failure or expired password exceptions accounted for in template?
No
Use matching on a blank line or a status line to uniquely distinguish an initial logon screen from an expired password or a logon failure screen.
Yes
Done!

### Agent detects the form?

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

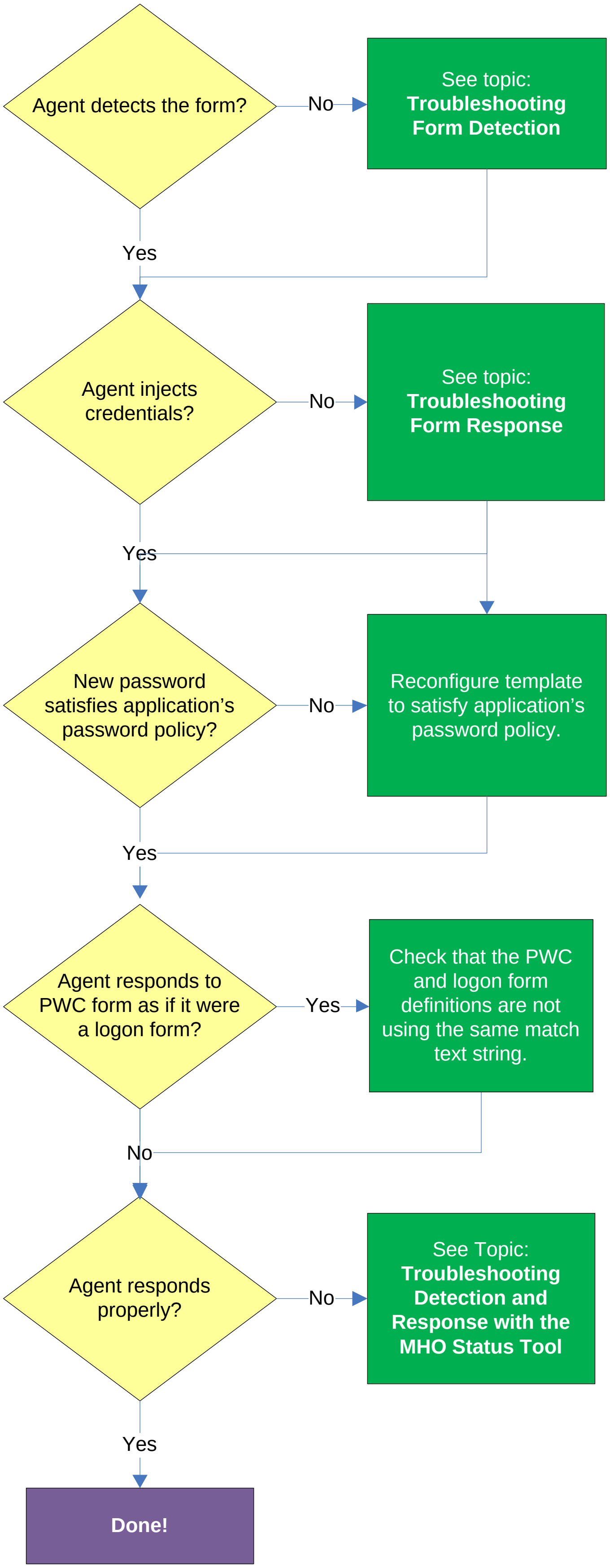
### Logon loop occurring on logout?

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.

### Logon failure or expired password exceptions accounted for in the template?

If the logon attempt fails, (for example, due to bad credentials), or the stored password has expired, most applications will display an appropriate message, but Logon Manager will not be by default aware of what happened and will attempt to log on again. In order to avoid this, set up a matching rule that will expect either a blank (no text) or space characters in the exact location of the error message. This way, whenever the message is displayed, the match rule will not be satisfied and Logon Manager will not attempt logon.

## Testing the Configuration of a Password Change Form


Agent detects the form?
No
See topic: **Troubleshooting Form Detection**
Yes
Agent injects credentials?
No
See topic: **Troubleshooting Form Response**
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 that the PWC and logon form definitions are not using the same match text string.
No
Agent responds properly?
No
See Topic: **Troubleshooting Detection and Response with the MHO Status Tool**
Yes
**Done!**

### Agent detects the form?

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

### New password satisfies application's password policy?

If the new password generated by Logon Manager 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), make sure that your password change form definition is not using the same match text string as your logon form definition – if it does, either match on a different string of text or, if possible, modify either of the application screens to include a piece of text against which Logon Manager can match uniquely.

If the Agent is still not responding to the form, see Troubleshooting Detection and Response with the SSO MHO Status Tool.

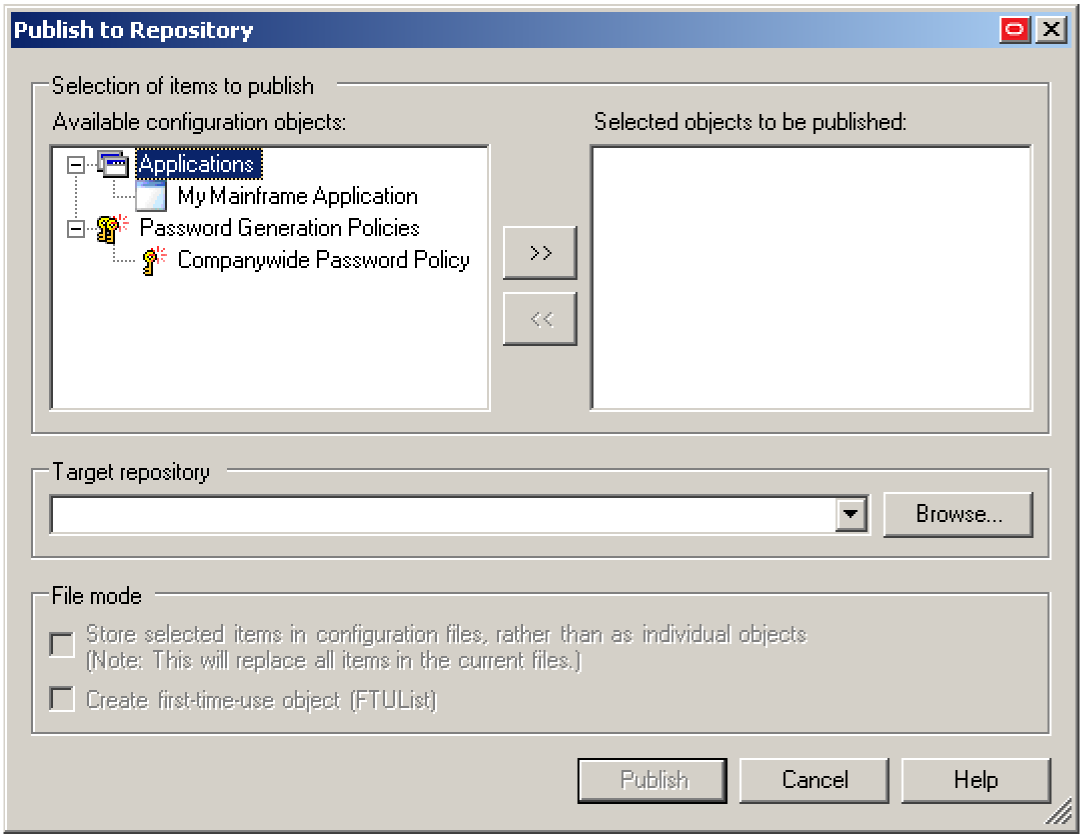
## Publishing a Template to the Repository

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 Logon Manager with a repository and best practices for structuring the repository tree, see the *Logon Manager Best Practices* guide for your platform.
>
> **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 Logon Manager 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 Mainframe Application
Password Generation Policies
Companywide Password Policy
>>
<<
Selected objects to be published:
Target repository
Browse...
File mode
Publish
Cancel
Help

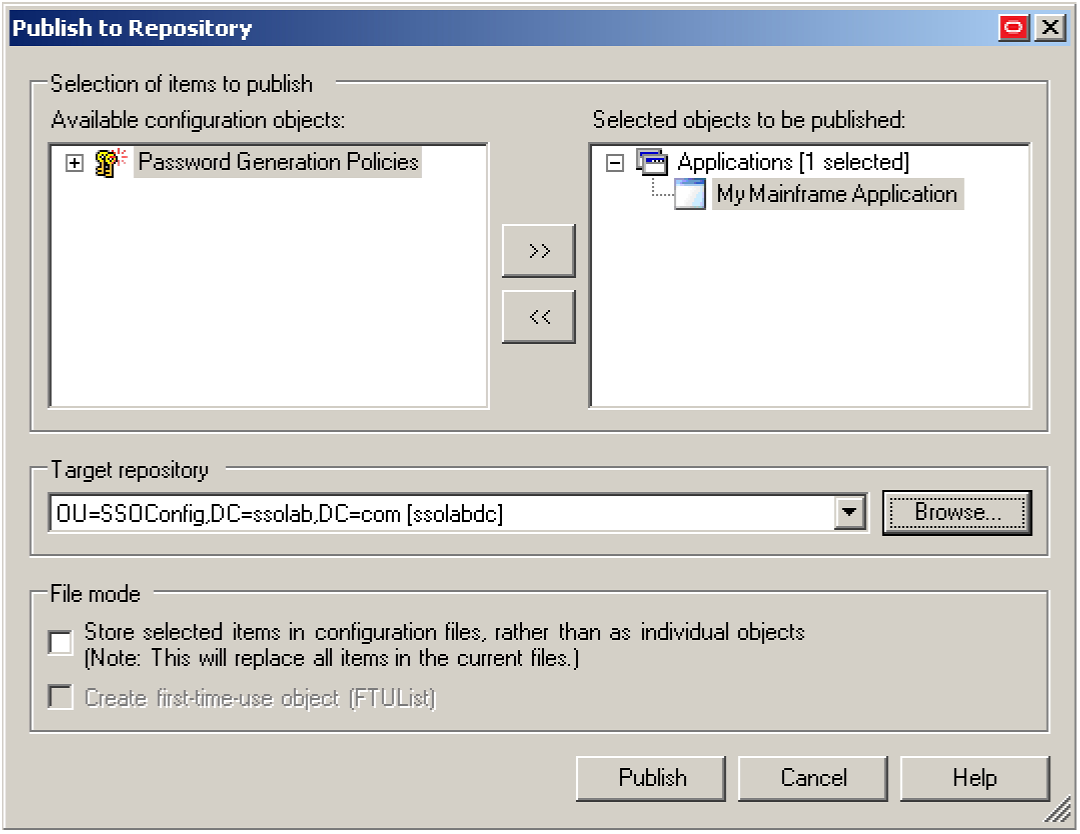
☐ Store selected items in configuration files, rather than as individual objects (Note: This will replace all items in the current files.)

☐ 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 Mainframe Application

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

File mode

□ 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:

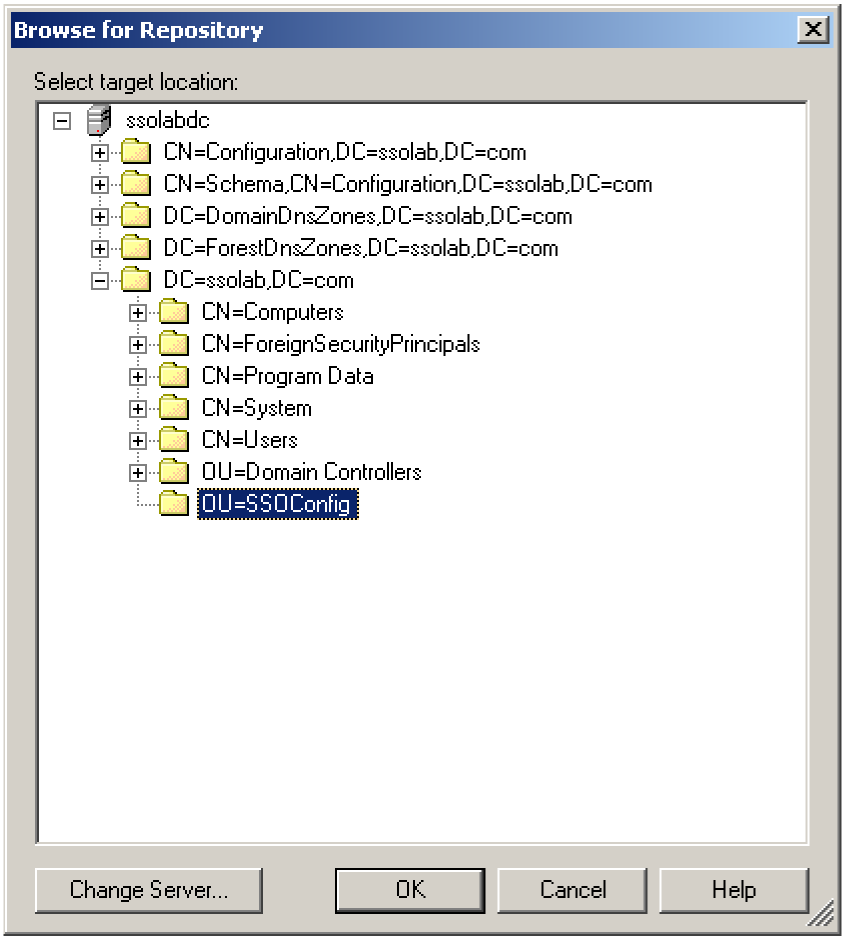
   - If you have previously published to the desired container, select it from the **Target Repository** drop-down list.

   - 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 Logon Manager Administrative Console help.

## Viewing and Modifying Emulator Configuration

The `mfrmlist.ini` file contains configuration information that allows it to detect and interact with terminal emulators.

**Note:** This `mfrmlist.ini` file is located in:

```
    %PROGRAMFILES%\Passlogix\v-GO SSO\Helper\Emulator
```

The most important configuration parameters for each emulator are:

- Full path and file name of the HLLAPI library file
- Session short –name
- Window title
- Window class
- Polling toggle (for emulators that do not properly notify the MHO of HLLAPI events)

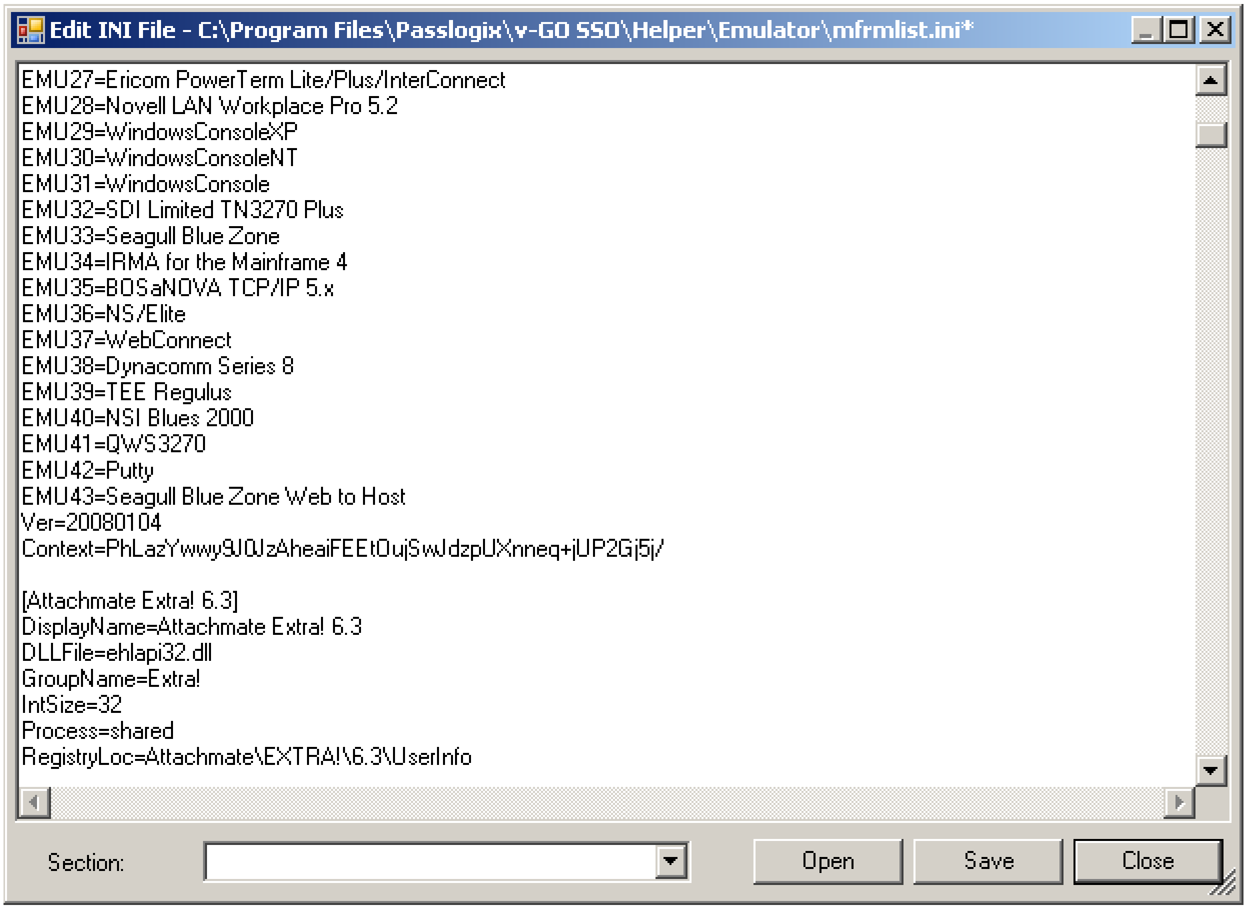
For a full list of parameters and their descriptions, see Emulator Configuration Parameter Reference.

To view or edit an emulator's configuration stored in the `mfrmlist.ini` file, do the following:

**Warning:** Do not modify the `mfrmlist.ini` file in an external editor – doing so will render mainframe support inoperable for any emulator whose configuration you edit in the external editor. This is because the Console's built-in editor automatically recalculates and updates the "magic number" stored in the Context parameter for each emulator when that emulator's configuration changes – an external editor will not update this magic number.

1. Launch the Logon Manager Administrative Console.
2. From the **Tools** menu, select **Modify Configuration → MfrmList**.

The Console displays the `mfrmlist.ini` in a built-in editor.


Edit INI File - C:\Program Files\Passlogix\v-GO SSO\Helper\Emulator\mfrmlist.ini*
Section:
Open
Save
Close

```
EMU27=Ericom PowerTerm Lite/Plus/InterConnect
EMU28=Novell LAN Workplace Pro 5.2
EMU29=WindowsConsoleXP
EMU30=WindowsConsoleNT
EMU31=WindowsConsole
EMU32=SDI Limited TN3270 Plus
EMU33=Seagull Blue Zone
EMU34=IRMA for the Mainframe 4
EMU35=BOSaNOVA TCP/IP 5.x
EMU36=NS/Elite
EMU37=WebConnect
EMU38=Dynacomm Series 8
EMU39=TEE Regulus
EMU40=NSI Blues 2000
EMU41=QWS3270
EMU42=Putty
EMU43=Seagull Blue Zone Web to Host
Ver=20080104
Context=PhLazYwwy9J0JzAheaiFEEtOujSwJdzpUXnneq+jUP2Gj5j/

[Attachmate Extra! 6.3]
DisplayName=Attachmate Extra! 6.3
DLLFile=ehlapi32.dll
GroupName=Extra!
IntSize=32
Process=shared
RegistryLoc=Attachmate\EXTRA!\6.3\UserInfo
```

At the top of the file there is a list of all emulators whose configuration data is stored as sections in the remainder of the file.

3. Do one of the following:
   - To modify existing configuration data, navigate to that section (to quickly jump to a specific section, select it from the **Section** drop-down list), locate the desired parameter and modify its value as desired.

     **Warning:** Do not modify the value of the `Context` parameter.

   - To add a section for a new emulator, do the following:
     a. Add an entry at the end of the list at the top of the file in the following format:

        `EMUxx=NewSectionName`

        where `xx` is one more than the number of the last entry in the list.

b. Create the new section in the following format:

```
[NewSectionName]
Parameter1=Value1
Parameter2=Value1
…
ParameterN=ValueN
```

**Caution:** The name of the new section *must* match the name you specified in the `EMUxx` entry at the top of the file in step 2a.

4. When you have finished, click **Save** to commit your changes and recalculate the `Context` parameters for all entries in the file.
5. Launch the newly added emulator and connect to an application session.
6. Use the MHO Status Tool to confirm that the MHO has successfully attached to the emulator's HLLAPI interface and detected the open session. See Troubleshooting Detection and Response with the SSO MHO Status Tool for instructions.

## Emulator Configuration Parameter Reference

The table below lists the available emulator configuration parameters (and their values) that you can use to create an entry in the `mfrmlist.ini` file.

> **Note:** Boolean-type parameters support the following values: `true, y, yes, 1, false, n, no, 0.`

| Setting | Description |
|---|---|
| `GroupName`<br>**Type:** text | (Groups emulator definitions and only allows one in a group to load<br>**Type:** text |
| `DisplayName`<br>**Type:** text | Used internally on the command line to other MHO processes |
| `RegistryName`<br>**Type:** text | Specifies an INI file name to check.<br>Windows directory will be searched first, then the PATH variable.<br>If this value is specified, *RegistryLoc* specifies a section name and *ValueName* specifies an entry name in the section and must not be omitted.<br>**Special Values:**<br>***WinIni*** – Indicates that the Windows INI file should be checked. |
| `RegistryLoc`<br>**Type:** text | Registry location path (see also: *RegistryName*, *ValueName*)<br>Specifies a registry location to search for *ValueName* in HKEY_LOCAL_MACHINE\Software tree.<br>**Special Values:**<br>**\*HKEY_CURRENT_USER*\*Software*\**"– If the entry starts with this text (case-sensitive), HKEY_CURRENT_USER\Software will be searched instead of HKEY_LOCAL_MACHINE\Software. |

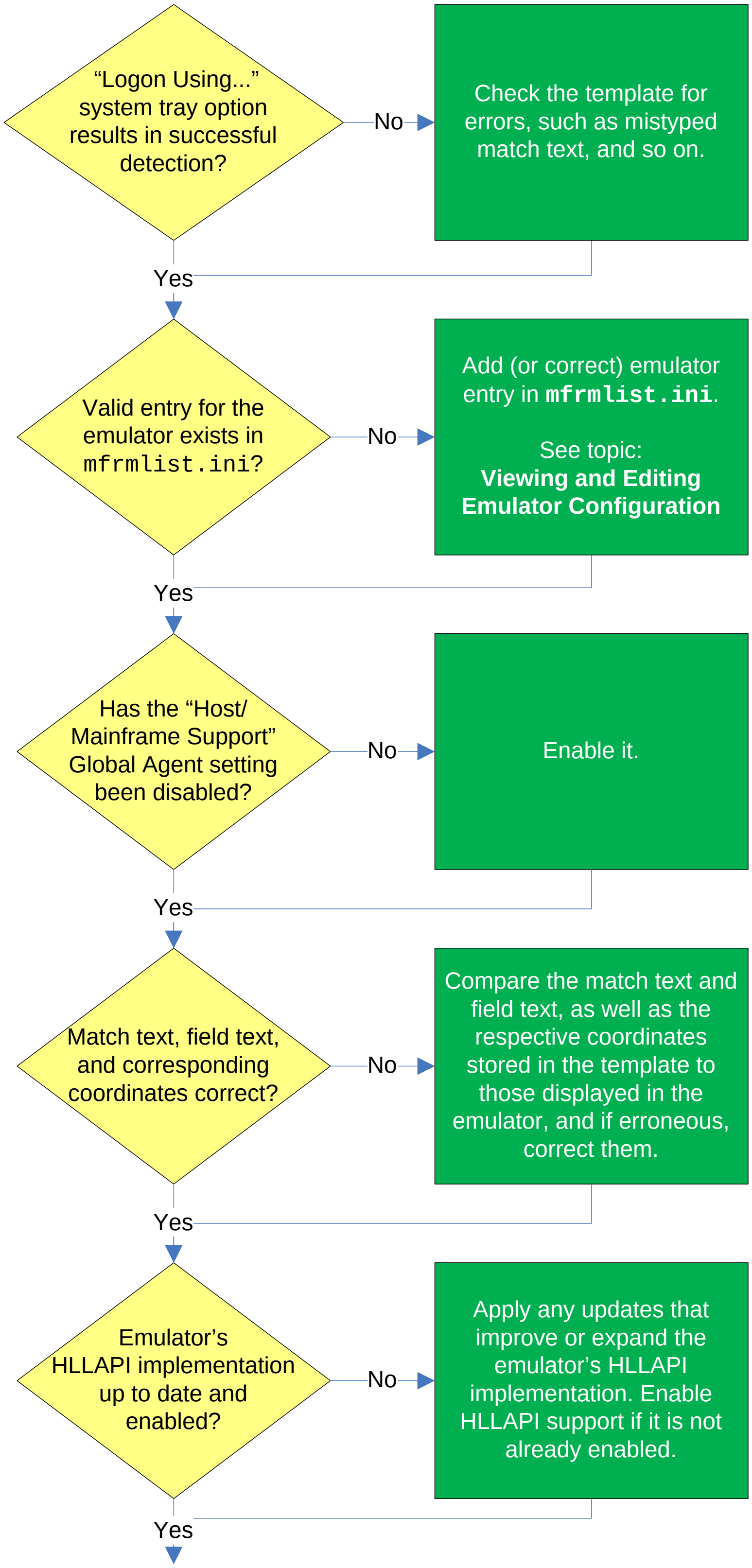
| | |
|---|---|
| `ValueName`<br>**Type:** text | Registry key name (see also: *RegistryName*, *StripFileName*, *DLLFile*)<br>The value that specifies a registry or section name that specified the location of *DLLFile*.<br>If *RegistryLoc* is omitted, this value directly specifies where to load *DLLFile* from.<br>**Special Values:**<br>***WinDir*** – Load dll from Windows directory.<br>***SysDir*** – Load dll from Windows System directory.<br>***HomeDir*** – Load dll from Logon Manager Installation directory.<br>***EmulDir*** – Load dll from Logon Manager Emulator Installation directory.<br>"**\**" – If the path starts with this value, load dll from the specified directory on the drive where Windows is installed. |
| `DLLFile`<br>**Type:** text | Name of the dll to load. |
| `StripFileName`<br>**Type:** boolean | Strip the filename from the retrieved registry string. |
| `PresentationSpaceSharing`<br>**Type:** special | ***SUPER_WRITE*** to put emulator in a supervisor mode (not all emulators support this) |
| `Process`<br>**Type:** special | ***shared*** causes the emulator to be loaded in the original copy of the mho.  Anything else and a new process is spawned. |
| `ConvertPosType`<br>**Type:** special | ***long***, ***short*** - Use enhanced or standard HLLAPI interface for Convert Position and Convert RowCol (99) function.<br>Default: *HLLAPIType* setting. |

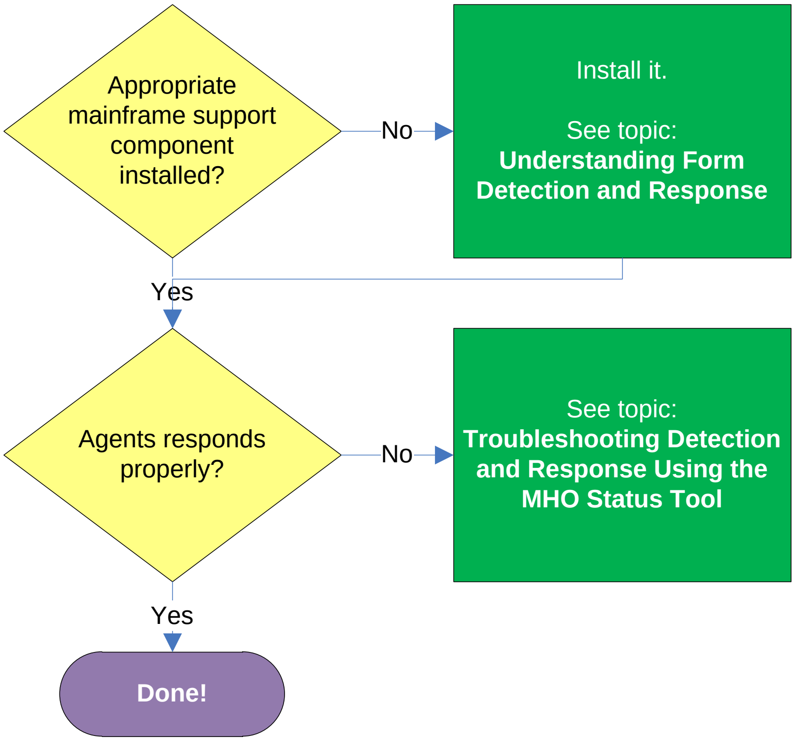
| | |
|---|---|
| `QuerySessionsType`<br>**Type:** special | ***long32***, ***long***, ***short***, ***both*** – *short* and *long* will use standard or enhanced HLLAPI interface for Query Sessions (10) function. *both* will attempt to use enhanced HLLAPI interface first and then standard, if enhanced fails (doesn't always work, specifying correct value is better.) *long32* is used for the Windows command prompt HLLAPI emulation.<br>Default: *HLLAPIType* setting. |
| `ValidateQuerySessionsData`<br>**Type:** boolean | Validates the data returned from QuerySessions before using it. |
| `QuerySessionStatusType`<br>**Type:** special | ***long***, ***short*** - Use enhanced or standard HLLAPI interface for Query Session Status (22) function.<br>Default: *HLLAPIType* setting. |
| `QueryHostUpdateType`<br>**Type:** special | ***long***, ***short*** - Use enhanced or standard HLLAPI interface for Query Host Update (24) function.<br>Default: *HLLAPIType* setting. |
| `StartNotificationType`<br>**Type:** special | ***long***, ***short*** - Use enhanced or standard HLLAPI interface for Start Host Notification (23) function.<br>Default: *HLLAPIType* setting. |
| `PlatformSize`<br>**Type:** numeric | **16**, **32** – 16-bit or 32-bit HLLAPI.<br>Default: 32 |
| `IntSize`<br>**Type:** numeric | **16**, **32** – Integer size used by the emulator.<br>Default: 32 |
| `QuerySystemType`<br>**Type:** special | ***long***, ***short*** - Use enhanced or standard HLLAPI interface for Connect Presentation Space (20) function.<br>Default: *HLLAPIType* setting. |

| | |
|---|---|
| `UpdateNotificationHandling`<br>**Type:** special | Special handling for return codes when update notification has occurred.<br>For example:<br>1.ReconnectSession<br>for return code 1, call internal function ReconnectSession<br>Functions available:<br>RecheckResetConnection - Calls reset, reattempts login<br>CheckLoginStatusAndReset - If no logins are pending, reset HLLAPI subsystem<br>ReconnectSession - Attempt to connect to the session<br>FirstLogin - Some emulators refuse to admit the screen has changed when first connecting<br>StartNotification - Attempt a login to the current session |
| `WindowClass`<br>**Type:** regexlist | Semicolon-separated list of regular expressions.<br>Window class matching. Includes regular expressions. Matches substrings. |
| `WindowTitle`<br>**Type:** regexlist | Semicolon-separated list of regular expressions.<br>Window title matching. Matches substrings.<br>Special Values:<br>$s – substituted by a session's short name. Substitution occurs before regular expression matching. |
| `UseSendKeys`<br>**Type:** regexlist | Use HLLAPI Send Key (3) to send keystrokes to the emulator (not to be confused with Windows SendKeys) instead of HLLAPI Copy String to Presentation Space (15) |
| `QuerySystemRequired`<br>**Type:** boolean | Make sure that the system is queried before doing a reset. |
| `ChangeNotiticationBroken`<br>**Type:** boolean | Some emulators did not implement change notification. |

| | |
|---|---|
| `RequireVisibleWindow`<br>**Type:** boolean | Don't log in if the window is hidden. |
| `EmulatorFixedRowWidth`<br>**Type:** boolean | Some emulators are incapable of determining how wide a row is. |
| `HllapiExportedCallName`<br>**Type:** text | The DLL's exported function name. Required if it's not "hllapi." |
| `LogonUsingTimeout`<br>**Type:** numeric | How long to wait for logon screen to appear in milliseconds.<br>Default: 1000 |
| `LoginDelay`<br>**Type:** numeric | Do not log in sooner than this many milliseconds following the last attempt.<br>Default: 1 |
| `WinHLLAPIStartup`<br>**Type:** version | If specified, calls WinHLLAPIStartup with specified requested version after loading the DLL. Calls WinHLLAPICleanup before unloading the dll.<br>If the call fails, dll will not be loaded.<br>Ex: 1.0 or 1.1 |
| `HllapiParamCommand`<br>**Type:** boolean | Emulator requires all settings to be passed to it |
| `IgnoreErrorCount`<br>**Type:** boolean | After a few errors, some emulators need to be reset, others stop working if you do [true, false] |
| `SupportsHwnd`<br>**Type:** boolean | Logon Manager extension to HLLAPI spec. Only implemented in SsoConNT and SSoConCP [true, false] |
| `HLLAPIType`<br>**Type:** special | ***long***, ***short*** – Use enhanced or standard HLLAPI interface for all HLLAPI functions.<br>Default: short |
| `ConnectSessionType`<br>**Type:** special | ***long***, ***short*** - Use enhanced or standard HLLAPI interface for Connect Presentation Space (5) function.<br>Default: *HLLAPIType* setting. |

## Troubleshooting Form Detection


"Logon Using..." system tray option results in successful detection?
No
Check the template for errors, such as mistyped match text, and so on.
Yes
Valid entry for the emulator exists in mfrmlist.ini?
No
Add (or correct) emulator entry in mfrmlist.ini.
See topic: Viewing and Editing Emulator Configuration
Yes
Has the "Host/ Mainframe Support" Global Agent setting been disabled?
No
Enable it.
Yes
Match text, field text, and corresponding coordinates correct?
No
Compare the match text and field text, as well as the respective coordinates stored in the template to those displayed in the emulator, and if erroneous, correct them.
Yes
Emulator's HLLAPI implementation up to date and enabled?
No
Apply any updates that improve or expand the emulator's HLLAPI implementation. Enable HLLAPI support if it is not already enabled.
Yes


Appropriate mainframe support component installed?
No
Install it.
See topic:
Understanding Form Detection and Response
Yes
Agents responds properly?
No
See topic:
Troubleshooting Detection and Response Using the MHO Status Tool
Yes
Done!

### Agent detects the form when invoking the "Logon Using Logon Manager" tray icon option?

Manually invoke window detection by using the **Logon Using Logon Manager** option from the Agent's system tray icon, then do one of the following:

- If the Agent detects the window, the configuration data for your emulator may be missing from the `mfrmlist.ini` file, or it might be erroneous. See Viewing and Modifying Emulator Configuration for instructions on understanding and editing the contents of the `mfrmlist.ini` file.
- If the Agent does not detect the target window even when you manually invoke detection by using the **Logon Using Logon Manager** option from the Agent's tray icon, do the following:
  - Make sure that the **Auto-recognize** option in the form definition's **Options** tab is enabled.
  - Review the template for common configuration errors, such as a mistyped window title or class; also, determine whether the window title and/or class are dynamic, and reconfigure the template as appropriate.

### Valid entry for the emulator exists in `mfrmlist.ini`?

If the emulator has been tested by Oracle and found to be compatible with Logon Manager, it is supported by Logon Manager out of the box and a corresponding entry exists for the emulator in the `mfrmlist.ini` file. The list of currently supported emulators can be found in the latest Logon Manager release notes, and can also be determined by looking for an entry for the emulator in question in the `mfrmlist.ini` file, as described in Viewing and Modifying Emulator Configuration.

If the emulator is not supported by Logon Manager out of the box and it is HLLAPI-compliant, you may add it to the `mfrmlist.ini` file (as described in Viewing and Modifying Emulator Configuration) in order to determine whether it is fully compatible with Logon Manager. While Oracle Support will aid you in this effort should you run into configuration issues, Oracle is unable to guarantee that an untested emulator will properly function with Logon Manager in all expected capacities.

**Note:** If you add an unlisted emulator to the mfrmlist.ini file and find it to be fully functional with Logon Manager, Oracle requests that you submit the emulator name and version to Oracle Support so that we can consider including official support for that emulator in the next release of Oracle.

### Has "Host/Mainframe Support" global Agent setting been disabled?

Check whether previously made changes to Logon Manager's configuration erroneously disabled the "Host/Mainframe Support" global Agent setting, located in the Logon Manager Administrative Console tree under **Global Agent Settings → End-User Experience → Response → Host/Mainframe Apps**. If the option is disabled, re-enable it and publish your changes to the repository, if applicable.

### Match text, field text, and corresponding coordinates correct?

Check whether the match text, field text, and the respective coordinates stored in the template match what's displayed by the application and correct any mismatches, if found. For more information Understanding Form Detection and Response.

### Emulator's HLLAPI implementation up to date and enabled?

In some cases, the emulator's initial HLLAPI implementation might be incomplete or defective. If possible, Oracle recommends researching and applying any vendor updates that improve or expand the emulator's HLLAPI implementation in a way that improves the emulator's compatibility with Logon Manager. Also, make sure that the emulator's HLLAPI support is enabled and that the emulator is exposing session short-names via HLLAPI. If it doesn't, Logon Manager will be unable to detect a session within the emulator.

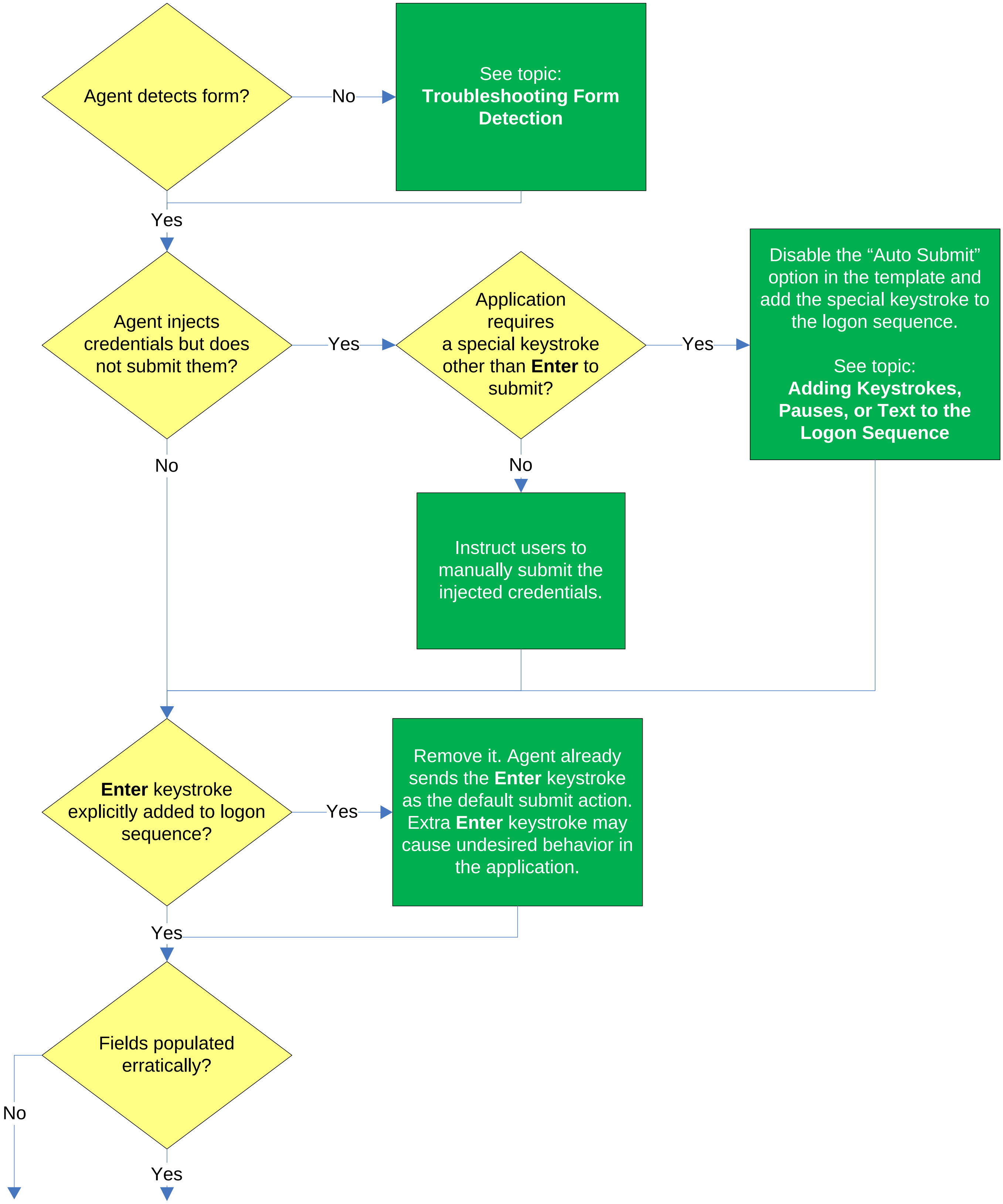
For more information on the level of your emulator's HLLAPI implementation, as well as instructions for configuring it, consult the emulator vendor. The Console help also provides quick tips on enabling HLLAPI support in most commonly used emulators.

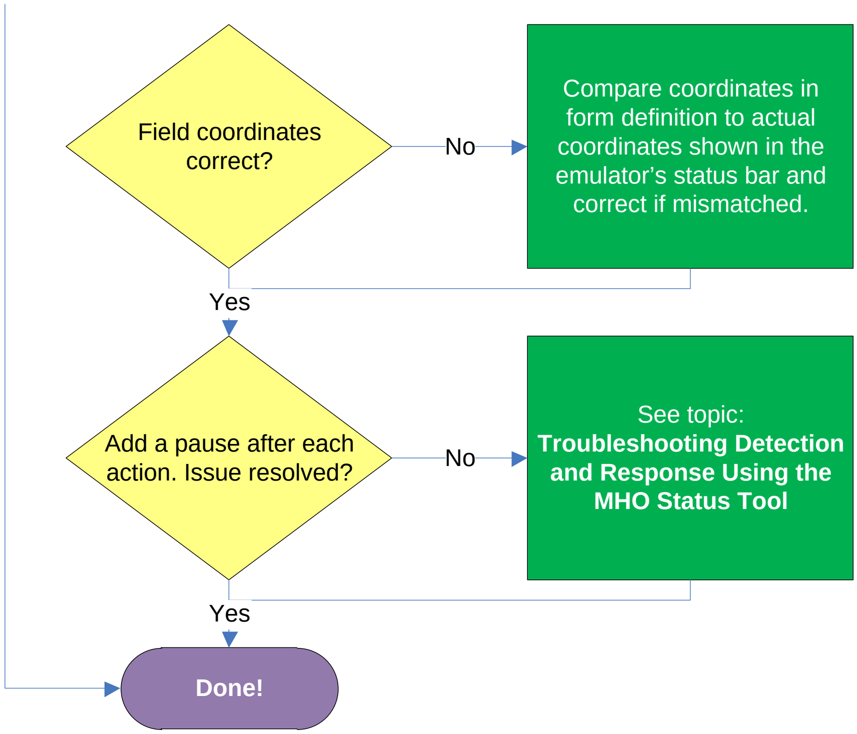
### Appropriate mainframe support component installed?

In order to interface with terminal emulators via HLLAPI, Logon Manager requires that the **Mainframe Support** (available under the **Extensions** node in the Logon Manager installer) is installed. Additionally, depending on your emulator, you might have to install additional helper components to fully enable Logon Manager with the emulator. (For more information on these helper components, see Understanding Form Detection and Response.)

If the Agent still does not detect the form, see Troubleshooting Detection and Response with the SSO MHO Status Tool.

## Troubleshooting Form Response


Agent detects form?
No
See topic: **Troubleshooting Form Detection**
Yes
Agent injects credentials but does not submit them?
Yes
Application requires a special keystroke other than **Enter** to submit?
Yes
Disable the "Auto Submit" option in the template and add the special keystroke to the logon sequence.
See topic: **Adding Keystrokes, Pauses, or Text to the Logon Sequence**
No
No
Instruct users to manually submit the injected credentials.
**Enter** keystroke explicitly added to logon sequence?
Yes
Remove it. Agent already sends the **Enter** keystroke as the default submit action. Extra **Enter** keystroke may cause undesired behavior in the application.
Yes
Fields populated erratically?
No
Yes


Field coordinates correct?
No
Compare coordinates in form definition to actual coordinates shown in the emulator's status bar and correct if mismatched.
Yes
Add a pause after each action. Issue resolved?
No
See topic: Troubleshooting Detection and Response Using the MHO Status Tool
Yes
Done!

### Agent detects the form?

If the Agent does not detect the form, check whether the **Auto-Recognize** feature has been disabled, and if so, re-enable it. If the feature has not been disabled, see Troubleshooting Form Detection.

### Agent injects credentials but does not submit them?

If the credentials are injected not automatically submitted, first check that the **Auto-Submit** option is enabled for the application – this causes the Agent to send the Enter keystroke as the default submit action. If **Auto Submit** is enabled but the Agent still does not submit them to the application, the application might require a special keystroke other than Enter (such as PF4) in order to submit the credentials. In such case, disable the Auto Submit feature and add the special keystroke to the logon sequence as described in Adding Keystrokes, Pauses, or Text to the Logon Sequence.

### Enter keystroke explicitly added to the logon sequence?

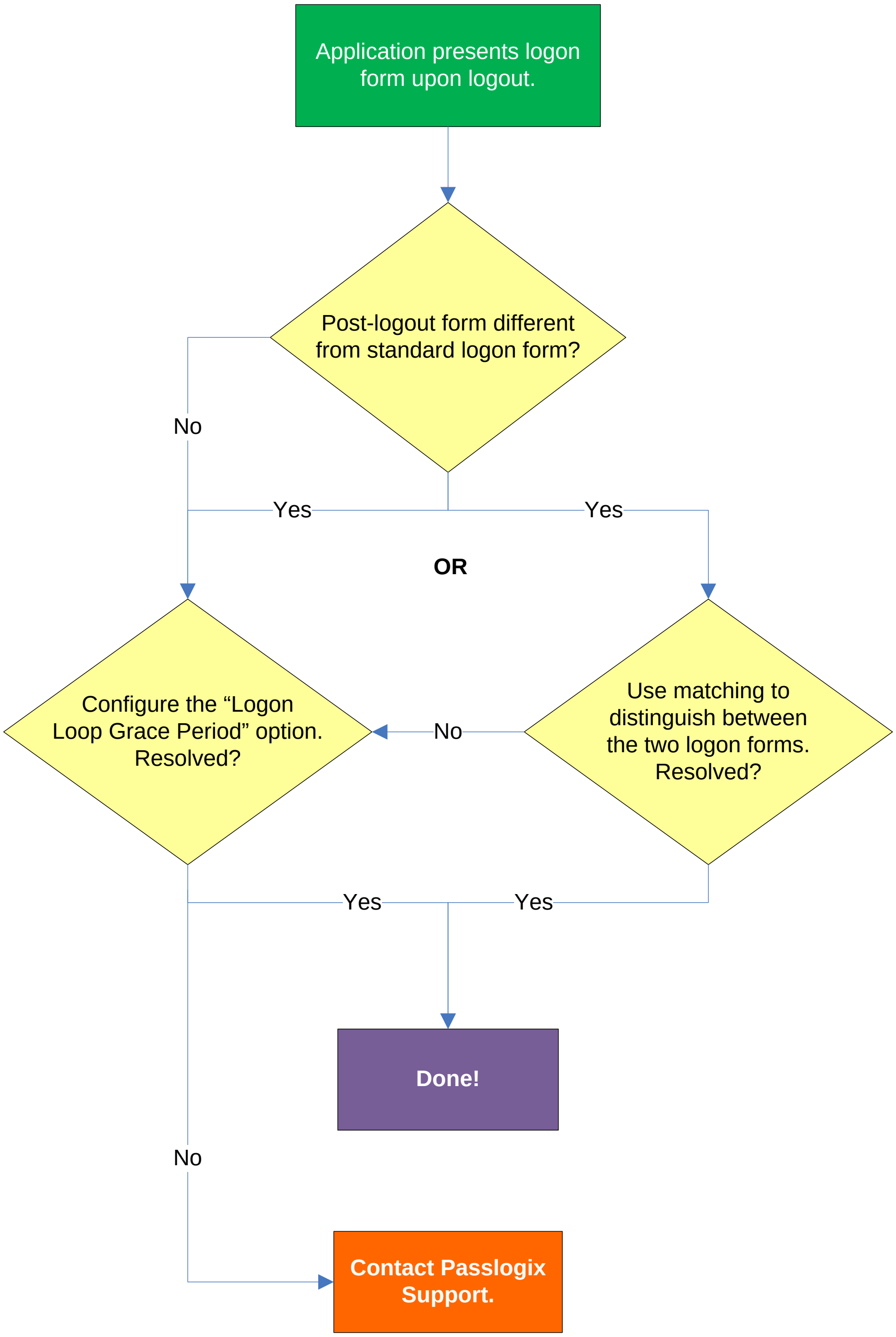
If you have explicitly added the Enter keystroke to the logon sequence and the application is exhibiting erroneous behavior during logon, remove the **Enter** keystroke from the logon sequence. When the **Auto Submit** feature is enabled, the Agent automatically sends the Enter keystroke as the default submit action – together with the explicit Enter keystroke you have added, the application is receiving two **Enter** keystrokes instead of the expected one, which causes undesired behavior after logon.

### Fields populated erratically?

If the Agent populates the fields erratically, i.e., inserts wrong, truncated, garbled, or blank values, check whether the field coordinates defined in the template are correct (by comparing them to the actual display shown in the emulator), and if not, correct them. If the coordinates are correct, add a pause after each action, as described in Adding Keystrokes, Pauses, or Text to the Logon Sequence.   If the issue persists even after adding the pauses, see Troubleshooting Detection and Response with the SSO MHO Status Tool to determine the point of failure.

## Troubleshooting a Logon Loop

Some applications display their logon form upon logout, which causes Logon Manager 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 Logon Manager from logging on to an application within set time period since the last logon.


Application presents logon form upon logout.
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 Passlogix Support.

### Post-logout form different from standard logon form?

Oracle recommends that you consider the “Logon Loop Grace Period” as well as matching if the logon form presented upon logout is sufficiently different from the application’s standard logon form. If the forms cannot be uniquely distinguished, use the “Logon Loop Grace Period” feature described below.

### 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 Logon Manager 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.

**Note:** If you have configured the logon grace period timer, and logon loop is still occurring for a specific form definition, make sure that the **Adhere to logon loop grace period** option in the form definition’s **Options** tab is enabled.

If this does not resolve the logon loop for the application, contact Oracle Support for assistance.

## Troubleshooting Detection and Response with the SSO MHO Status Tool

The MHO provides a HLLAPI status interface that displays real-time information describing MHO's interaction with terminal emulators. To display this interface, execute the MHO executable (`ssomho.exe`) with the `/showmho` switch:

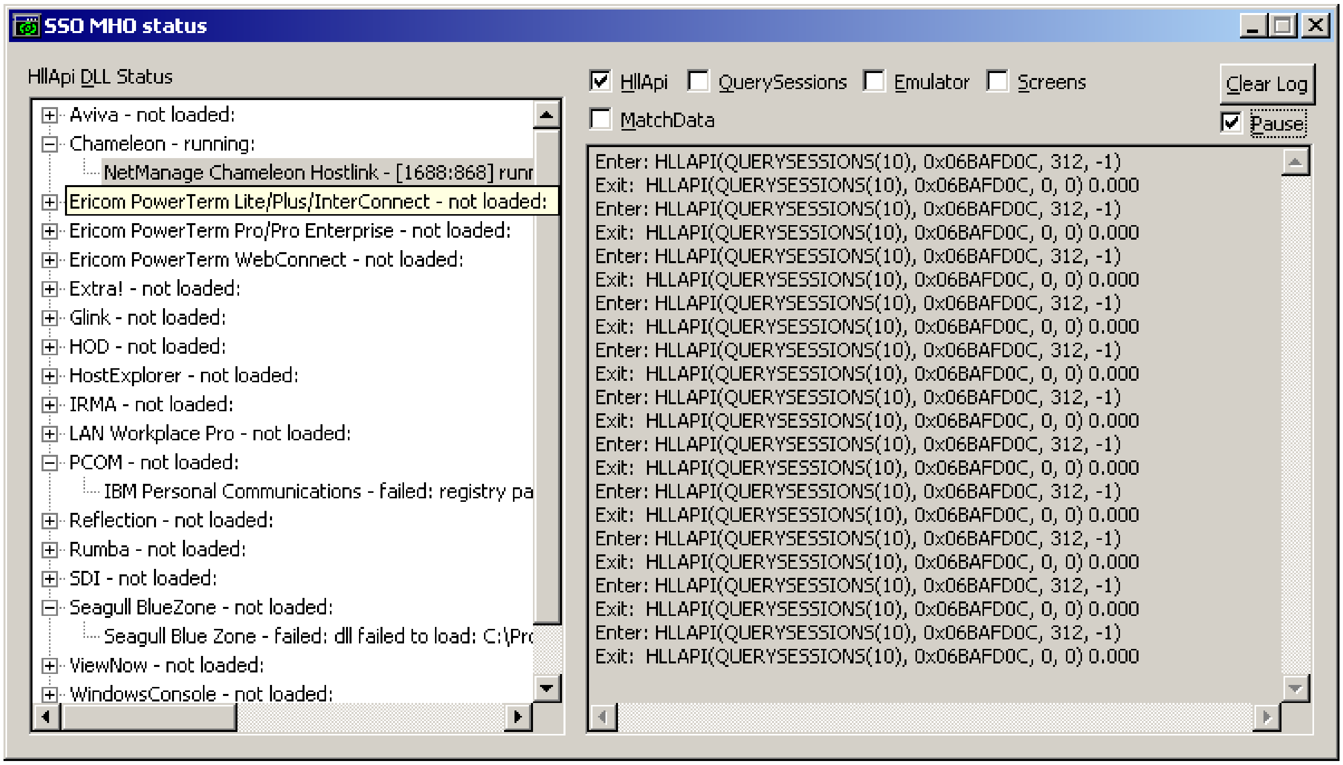
```
<program_files_folder>\Passlogix\v-GO-SSO\Helper\Emulator\ssomho.exe /showmho
```

Oracle highly recommends creating a desktop shortcut to the tool so that you can launch the tool easily at your convenience without having to remember the exact syntax.

> **Note:** If the MHO does not detect at least one active HLLAPI interface on the system, it will shut down to conserve resources, which may impair the troubleshooting process. In order to keep the MHO running continuously, install the "DOS Window Support" component (described in Understanding Form Detection and Response), which will always expose its active HLLAPI interface to the MHO.

### Understanding the MHO Status Tool Interface

The MHO status interface looks as follows:



☑ HllApi ☐ QuerySessions ☐ Emulator ☐ Screens

☐ MatchData ☑ Pause

```
⊞ Aviva - not loaded:
⊟ Chameleon - running:
    NetManage Chameleon Hostlink - [1688:868] runr
⊞ Ericom PowerTerm Lite/Plus/InterConnect - not loaded:
⊞ Ericom PowerTerm Pro/Pro Enterprise - not loaded:
⊞ Ericom PowerTerm WebConnect - not loaded:
⊞ Extra! - not loaded:
⊞ Glink - not loaded:
⊞ HOD - not loaded:
⊞ HostExplorer - not loaded:
⊞ IRMA - not loaded:
⊞ LAN Workplace Pro - not loaded:
⊟ PCOM - not loaded:
    IBM Personal Communications - failed: registry pa
⊞ Reflection - not loaded:
⊞ Rumba - not loaded:
⊞ SDI - not loaded:
⊟ Seagull BlueZone - not loaded:
    Seagull Blue Zone - failed: dll failed to load: C:\Prc
⊞ ViewNow - not loaded:
⊞ WindowsConsole - not loaded:
```

```
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
Enter: HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 312, -1)
Exit:  HLLAPI(QUERYSESSIONS(10), 0x06BAFD0C, 0, 0) 0.000
```

The interface is comprised of the following sections:

- **HLLAPI DLL Status pane** – displays all the emulator entries present in the `mfrmlist.ini` file and the status of the associated HLLAPI interfaces: o **Running** – the emulator's interface has been detected and hooked on to by the MHO. Expanding a node with a **Running** status displays the detected session. In our example above, the **NetManage Chameleon Hostlink** emulator is in this state.
- **Not loaded** – the emulator's interface was not detected by the MHO. Expand this node to view the reason for the load failure is displayed as a sub-node under the emulator node. Typical reasons include:
    - **Registry path not found** – the registry key which defines the path to the emulator's HLLAPI Dynamically Linked Library (DLL) cannot be found at the location specified in the `mfrmlist.ini` file. Consult the emulator vendor to obtain the correct value for this parameter and modify the `mfrmlist.ini` file entry for the emulator to correct this problem. In our example above, the **IBM Personal Communications** emulator is showing this error.
    - **DLL failed to load** – the emulator's HLLAPI library was not found at the location specified by the registry path described above, or the file that exists at the specified location is corrupted and cannot be hooked on to by the MHO. In our example above, the **Seagull BlueZone** emulator is showing this error.
- **Log data pane** – this pane displays real-time information describing the interaction between the MHO and the emulator selected in the "HLLAPI DLL Status" pane.

The check boxes above the log data pane enable or disable the display of the following classes of events:

- **HLLAPI** – all HLLAPI communication between the MHO and the detected emulator interfaces. Leave this checked at all times or no log data will be captured.
- **Emulator** – emulator events such as user input or window instantiation.
- **QuerySessions** – session-related events, such as list of open sessions, detected session short-names, and so on.
- **Screens** – content of the emulator's display. Every time the MHO receives (or detects, if polling is enabled) a screen update event, the contents of the updated screen are captured.
- **MatchData** – attempted matches against the match text defined in the template, as well as the outcome of those attempts.

As the information displayed in the log data pane scrolls by quickly, whenever you notice something of interest, select the **Pause** check box to temporarily pause the data capture and copy the data in question into a text editor for more thorough analysis.

## Using the MHO Status Tool to Diagnose Session Detection and Response

If you have created a mainframe application template but Logon Manager is not detecting the terminal session, the most common causes are:

- **The emulator is not exposing a session short-name via HLLAPI.** Logon Manager uses the session short-name to detect the session and initiate text matching. Some emulators require explicit configuration to expose the session short-name, either globally, or on a per-session basis. In this case, you would see the emulator listed in the “HLLAPI DLL Status” pane as **Running** but expanding the emulator’s node would reveal no active session even though the session was connected in the emulator itself. Consult the vendor’s documentation for information on how to enable this required feature.
- **The emulator’s HLLAPI support is disabled or not installed.** In this case, you would see the emulator listed in the “HLLAPI DLL Status” pane as “Not loaded” and expanding its node would reveal either the “Registry path not found” or the “DLL failed to load” error described in the previous section. Consult the vendor’s documentation for information on how to enable or install HLLAPI support.
- **Bug or HLLAPI-noncompliance in the emulator’s code.** Consult the vendor to determine if an update is available. For example, with all event types enabled in the “Log data pane,” you would not see any screen update events or session-related information (such as session opening or closing) even though the emulator would appear as “Running” in the “HLLAPI DLL Status” pane and expanding its node revealed a detected session.