# Configuring Rules for Single Sign On

# Configuring Single Sign-on

- User interface pattern recognition
- Web interfaces pattern recognition
- Configuring rules for basic / kerberos authentication

SSO system is configured based on the detection of administrator defined User Interface patterns. The system currently supports native Windows applications, Java applications and Web applications.

The UI Patterns are expressed with XML files associated with each application entry point. They are composed of:

- **Rules** for detecting user interfaces (defined like application attributes or elements).
- **Action** to be taken on user interface recognition. (defined with the action element for the application).

Complementary to the rules defined in Sofifd Console, the synchronization server manages a repository of user accounts and passwords, as well as other information generically known as **secrets**. In general, the system will handle any number of secrets as well as any number of accounts for each managed systems. Anyway, each account for a managed system will have only one password.

All secrets can be used and manipulated using a scripting language fully compatible with ECMA-Script, also known as Javascript.

# User interface pattern recognition

The user interface detection for Windows and Java applications is done using the **Application** tag. This tag will contain one or more Component tagged elements. Each component could have many nested components. Each component could have one or more actions to perform when the

user focus is at a selected component.

Next is a sample to inject the secret name “JconsolePassword” into jconsole application:

| | |
|---|---|
| <Mazinger><br><**Application** cmdLine = '.*jconsole$'><br><**Component** class = 'sun.tools.jconsole.JConsole' title = 'Java Monitoring & Management Console' name = 'frame0'><br><**Component** class = 'sun.tools.jconsole.JConsole\$FixedJRootPane'><br><**Component** class = 'javax.swing.JPanel' name = 'null.glassPane' /><br><**Component** class = 'javax.swing.JLayeredPane' name = 'null.layeredPane'><br><**Component** class = 'javax.swing.JPanel' name = 'null.contentPane'><br>....<br><**Component** class = 'javax.swing.JPasswordField' text ='' ref-as = 'password'> | Patterns to be match |
| <**Action** type='script' event='onFocus'><br><! [CDATA [<br>var account = secretStore.getAccount('soffid');<br>var password = secretStore.getPassword('soffid', account);<br>debug ('user =' + account);<br>debug ('password =' + password);<br><br>password.setText (secretStore.getSecret ('password'));<br>]]><br></ **Action**> | The action you want to be executed |
| .... | |

Thus, when the system detects that the user is within a window that meets the XML specification and the password text box is the focus owner, Soffid will execute the script action that is bound. This one will show the user password in a jconsole application field.

The Application contains in the attribute cmdLine a regular expression that is matched against the process command line. In the example, SSO will only match a running program with a command line that ends with "jconsole". It won't apply to jconsole.exe or “jconsole test”.

The element **Application** accepts the following attributes:

| | |
|---|---|
| cmdLine | Regular expression to match the command line. |

The **Component** element allows the following attributes:

| | |
|---|---|
| class | Regular expression to validate against the kind of visual component, either a Java class or a window class. |
| name | Regular expression to match the name of the component. Applies only to Java components. |
| text | Regular expression to match the content of a text component |
| title | Regular expression to match the title of a java component. |
| dlgId | Regular expression to match window ID dialog on Windows component. |
| optional | If the value is true, the presence of the component is not considered critical to trigger actions associated dialogue. |
| check | When the check attribute has the value “**partial**”, the matcher engine considers the user interface component matches the XML pattern even when it has one or more children components that are not declared at the XML pattern.<br>If you specify the value **full** value or the attribute is missing, the component will not match the pattern if it has children are components not specified in XML descriptor. Thus, the rule will be ignored. |
| ref-as | Specifies a name of a ECMA-Script variable that will refer to this component. |

The **Action** element accepts the following attributes:

| | |
|---|---|
| event | Name of the event that will trigger the action. In the current version must be set to "onFocus" |
| type | Indicates the type of action. Can have the following values:<br>**setText:** Assigns a text value to the owner component.<br>**script.** Run the specified script. |
| text | Text to assign, for setText actions. |
| repeat | If set to true, the action will be executed as many times as necessary.<br>Otherwise, it will only run once per process. |
| delay | Time (in seconds) that must be elapsed before the action is executed again. |

# Web interfaces pattern recognition

The detection is done using the element **WebApplication**. This tag is independent of the browser used, and is based solely on the content of web document. Thus, the same rule will work both on Mozilla Firefox, Google Chrome or Internet Explorer.

| | |
|---|---|
| <Mazinger><br><**WebApplication** url = 'https://www.caib.es. *' title = 'Government of the Balearic Islands'><br><**Form** action = "j_security_check"><br><**Input** name="j_username" ref-as="u"/><br><**Input** name="j_password" type="password" ref-as="p"/><br><**Input** type="Submit" ref-as="b" /> | Patterns to be match |
| <**Action** Type='script' event='onLoad'><br><! [CDATA [<br>debug('User =' + secretStore.getAccount ('user'));<br>debug ('password =' + secretStore.getPassword('soffid', account));<br>u.setAttribute ('value' secretStore.getAccount ('user'));<br>p.setAttribute ('value' secretStore.getPassword('soffid', account));<br>b.click();<br>]]><br></**Action**> | Action you want to be executed |
| </**Form**><br></**WebApplication**><br></Mazinger> | |

Thus, when the system detects that the browser has loaded a page matching the XML specification (url, title, and components), it will run the actions that have been associated.

Mind that despite the actions being coded in Javascript, it is not the Browser javascript engine. Thus, you cannot use browser variables or functions.

The element **WebApplication** accepts the following attributes:

| | |
|---|---|
| url | Regular expression to match the page address |
| title | Regular expression to match the title of the page |
| content | Regular expression to match the HTML content of the page |

The **Form** element will search in the HTML document for a form that matches the specified attributes. It can optionally contain one or more input elements that must be present in the HTML document. It accepts the following attributes:

| | |
|---|---|
| id | Regular expression to match the ID attribute of the element |
| name | Regular expression to match the element name |
| method | Regular expression to match the form element's method attribute. |
| action | Regular expression to match the form element's action attribute. |
| ref-as | Specifies a name of a ECMA-Script variable that will refer to this form. |
| optional | A value of **true** indicates that its presence is not necessary for the execution of actions. |

The **Input** element will search in the HTML document for an input element that matches the specified attributes. Input elements can be located within WebApplication or Form elements. In the first case, you will find there is any input into the document. In the second case, just find the type items included in the input form found.

| | |
|---|---|
| id | Regular expression to match the ID attribute of the element |
| name | Regular expression to match the element name |
| type | Regular expression to match the input type |
| value | Regular expression to match the input value. |
| ref-as | Specifies a name of a ECMA-Script variable that will refer to this form. |
| optional | A value of **true** indicates that its presence is not necessary for the execution of actions. |

The **Action** element accepts the following attributes:

| | |
|---|---|
| event | Name of the event that will trigger the action. In the current version must be set to "onFocus" |
| type | Indicates the type of action. Can have the following values:<br>**setText:** Not supported<br>**script.** Run the specified script. |
| repeat | If set to true, the action will be executed as many times as necessary.<br>Otherwise, it will only run once per process. |
| delay | Time (in seconds) that must be elapsed before the action is executed again. |

# Configuring rules for basic / kerberos authentication

Some web pages are still using basic or kerberos authentication mechanisms. These mechanisms do not present a web page to be filled in by the user. Thus, the ESSO engine cannot detect it using the method described previously.

Instead, starting from Soffid ESSO version 3.0.0, there is a new tag to teach the ESSO which credentials to send in these cases. The rules will be like the next ones:

```
<Mazinger>
  <WebTransport url="https://no-soffid.bubu.lab:4443/" system="OSCM"/>
  <WebTransport url="https://no-ad.bubu.lab/" system="ad" domain="AD"/>
</Mazinger>
```

The tag to use is WebTransport. It has three parameters:

| Attribute | Value |
|---|---|
| url | The base url to use. Include the protocol and port number when needed. Any BASIC, NTLM or Kerberos authentication requested by that server will be automatically answered with the credentials present in the password vault |
| system | The ESSO will send any credential that the user has in that system. Other credentials will be ignored |
| domain | This is an optional attribute. It's required when trying to use Kerberos or NTLM authentication if the account name does not contain the domain name part. If the account contains the domain name, this attribute should not be present. |

Due to the different ways that browsers address this kind of authentication, the user interface will be displayed according to the browser settings. For instance, Edge and Internet Explorer will display a UA dialog box.

---

Revision #8
Created 28 May 2021 08:23:06
Updated 25 April 2024 11:12:43