PANDORAFMS

ENTERPRISE

Pandora UX monitoring plugin

ÍNDICE

1. INTRODUCCIÓN

Plugin que permite la ejecución estándar de sesiones pregrabadas PDR y PWR, revisión OUM 745 03/06/2020.

2. PARÁMETROS GENERALES

Configuración de ejecución del plugin

-exe

directorio al binario de tu utilidad de automatización o herramienta de test. También es posible seleccionar el modo interno:

** Solo Windows:
PDR - herramienta compleja de chequeos de escritorio
** windows & linux:
PWR - herramienta compleja de chequeos web (con Selenium)
** requiere curl en el PATH:
stats - monitorización rápida del rendimiento del web server
** solo linux:
goliat – chequeos web basadas en Pandora Webserver

-args *(opcional)*

argumentos para la herramienta de automatización

-script

script(s) a ejecutar
Se pueden definir tantos como se necesiten. Por ejemplo:
-script script1,script2,script3

-folder *(opcional)*

directorio en el que se almacenarán los screenshots

-name *(opcional)*

* PWR: define el nombre de la transacción
* PWD: nombre de la fase, se pueden definir tantas como se necesiten
ejemplo: -name phase1,phase2,phase3
deberían coincidir con los scripts definidos

-pwr_port *(opcional)*

puerto de escucha del servidor PWR
4444 por defecto, requiere -exe PWR

-pwr_host *(opcional)*
host en el que se encuentra el servidor PWR
Localhost por defecto, requiere -exe PWR

-pwr_browser *(opcional)*
navegador en el que se ejecutarán los chequeos PWR
*Firefox por defecto, requiere -exe PWR

-pwr_classic *(opcional)*
usa las librerías antiguas Test::WWW::Selenium en lugar de PandoraFMS::WebDriver, requiere -exe PWR, 0 por defecto

-retries *(opcional)*
número máximo de reintentos en caso de error

-ss_config *(opcional)*
configuración de los screenshots: X,Y,Ancho,Altura
ejemplo: -ss_config 0,0,100,100
obtendrá una imagen de 100x100 desde las coordenadas 0,0
[solo Windows]: activa el parámetro -ss_config para capturar la ventana activa

-checkpoint *(opcional)*
crea un nuevo screenshot de control

-s *(opcional)*
macro de sustitución, Sustituye cada instancia de _texto_ por el valor configurado. Es posible usar now+1 como valor para insertar el día actual. Se pueden configurar tantas macros (-s) como se necesiten, seguidas con el formato (-f) en caso de que sea necesario:
-s macro1=value1 -f format1 -s macro2=value2 -f format2

-f *(opcional)*
formato para las macros de tipo fecha
usa notación std:
ejemplo: -f "%y%m%d" con fecha 2016, Sep 13 para obtener 160913

-k *(opcional)*
usa la salida de errorlevel para los chequeos en caso de fallo. Se comprueba por defecto en caso de que el plugin devuelva alguna fase de '[error]'

-max_differences *(opcional)*
Especifica el porcentaje máximo de diferencia entre la imagen de control y las obtenidas como resultado del test. Por defecto está configurado con el valor numérico 0.5

Configuración de email

-to *(opcional)*
dirección de email de destino que se enviará en caso de error

-from *(opcional)*
remitente del email

-subject *(opcional)*
asunto del email

-content *(opcional)*
contenido del email

-smtp_gw *(opcional)*
gateway del servidor smtp, 127.0.0.1 por defecto

-smtp_port *(opcional)*
puerto del servidor smtp. 25 por defecto

Configuración de los módulos de Pandora FMS

-t name *(opcional)*
(solo modo PDR) habilita y configura un nombre para la transacción "global". De esta manera todos los módulos de los chequeos irán asociados al módulo de transacción global, llamado 'UX XXXX nombre'

-interval *(opcional)*
configura un module interval para los módulos resultantes del chequeo

-g *(opcional)*
configura un module group para los módulos resultantes del chequeo

-post *(opcional)*
comando que se ejecutará al acabar el chequeo, por ejemplo, matar firefox:
-post "taskkill /F /IM firefox.exe"
-post "killall firefox"

-tag_list x *(opcional)*
añade x (separados por comas) tags a los módulos generados, ejemplo: tag1,tag2,...

Campos extra

-exit_on_fail *(opcional)*
termina inmediatamente la transacción si no es posible conectar con el servidor PWR. 0 por defecto

-phase_snapshot *(opcional)*
genera un screenshot por fase. 0 por defecto

-classic_mode *(opcional)*
El token a 0 siempre actualiza la latencia. El token a 1 solo actualiza la latencia si la comprobación ha resultado exitosa. 0 por defecto

-v *(opcional)*
Activa el modo verbose

-agent *(opcional)*
Define la ejecución como plugin de servidor, configura el nombre del agente en el que se almacenarán los módulos de salida

-agent_group *(opcional)*
si se define el token "agent" y crea un agente nuevo, configura el grupo al que pertenecerá

-interval *(opcional)*
si se define el token "agent" y crea un agente nuevo, configura el intervalo que tendrá. 300 por defecto

-tentacle_ip *(opcional)*
ip del servidor tentacle. 127.0.0.1 por defecto

-tentacle_port *(opcional)*
puerto del servidor tentacle. 41121 por defecto

-tentacle_opts *(opcional)*
opciones extra del servidor tentacle

-local_folder *(opcional)*
directorio local de tentacle. /var/spool/pandora/data_in por defecto

-mode *(opcional)*
[local|tentacle] modo de transferencia, "local por defecto"

3. AUTOMATIZACIÓN DE LAS SESIONES

3.1 Ejecución de sesiones PWR

3.1.1 Ejecución estándar

Para lanzar sesiones pregrabadas de PWR, indicaremos que el modo de trabajo es PWR (con el parámetro –exe), y el archivo que contiene las directrices de la sesión (con el parámetro –script):

```
pandora_ux_x64.exe -exe PWR -script tests\std.html
```

Opcionalmente, podremos indicar el directorio en el que guardaremos los screenshots en caso de fallo en el chequeo y la ip y el puerto en el que se aloja el hub de selenium que realizará el chequeo.

```
pandora_ux_x64.exe -exe PWR -script tests\std.html -folder C:\sondas\ -
pwr_host 192.168.80.40 -pwr_port 4445
```

Se devolverán los siguientes módulos:
UX_Time_nombre_proyecto
UX_Status_nombre_proyecto

Si hay alguna fase con error, se creará también el módulo siguiente:
UX_Snapshot_nombre_proyecto

Mostrará una imagen de la web en el momento del error siempre y cuando se esté ejecutando en el mismo equipo en el que está corriendo el servidor de Selenium.

Ejemplo de salida con ejecución correcta:

```
<module>
        <name><![CDATA[UX_Status_std.side]]></name>
        <type>generic_proc</type>
        <data><![CDATA[1]]></data>
        <description><![CDATA[Test OK]]></description>
</module>
<module>
        <name><![CDATA[UX_Time_std.side]]></name>
        <type>generic_data</type>
        <data><![CDATA[16.317]]></data>
        <description><![CDATA[Test OK]]></description>
</module>
```

Ejemplo de salida con ejecución fallida:

```
<module>
        <name><![CDATA[UX_Status_std.side]]></name>
        <type>generic_proc</type>
        <data><![CDATA[0]]></data>
        <description><![CDATA[Failed to execute verifyText]]></description>
</module>
<module>
        <name><![CDATA[UX_Time_std.side]]></name>
        <type>generic_data</type>
        <data><![CDATA[15.463]]></data>
        <description><![CDATA[Failed to execute verifyText]]></description>
</module>


<module>
        <name><![CDATA[UX_Snapshot_std.side]]></name>
        <type>async_string</type>
        <data><![CDATA[
iVBORw0KGgoAAAANSUhEUgAA…JRU5ErkJggg==]]></data>
        <description><![CDATA[Image (last error)]]></description>
</module>
```

3.1.2 Ejecución basada en transacciones

Si se ha realizado la grabación del chequeo con la información adicional basada en transacciones, será el propio sistema quien genere los módulos necesarios para identificar cada una de las fases indicadas.

```
pandora_ux_x64.exe -exe PWR -script tests\std.html -folder C:\sondas
```

Se devolverán los siguientes módulos por fase[1]:

- UX_Time_nombre_proyecto.fase_#orden
- UX_Status_nombre_proyecto.fase_#orden

Si hay alguna fase con error, se creará también el módulo siguiente:

- UX_Snapshot_nombre_proyecto.fase_#orden

Mostrará una imagen de la web en el momento del error.

También se devolverán los módulos de resumen globales identificados con los siguientes nombres:

- UX_Global_Time_nombre_proyecto
- UX_Global_Status_nombre_proyecto

[1] #orden representa el número correspondiente al orden en que se ha declarado (1, 2, 3...)

- UX_Global_Snapshot_nombre_proyecto

Mostrará una imagen de la web en el momento del error.

3.2 Ejecución de sesiones PDR

3.2.1 Ejecución estándar

Para lanzar sesiones pregrabadas de PDR, indicaremos que el modo de trabajo es la ruta al fichero pdr.cmd, el argumento de dicho fichero "-r", el archivo que contiene las directrices de la sesión (-script), el directorio donde almacenar las capturas de pantalla (-folder) terminando en '\'.

En la ejecución siguiente también se personaliza la captura de pantalla para recoger únicamente la ventana activa:

```
pandora_ux_x64      -exe      C:\PDR\pdr.cmd      -args      -r      -script
C:\pandora_ux\calculadora.sikuli -folder C:\pandora_ux\ -ss_config active
```

Se devolverán los siguientes módulos:

- UX_Time_nombre_proyecto
- UX_Status_nombre_proyecto
- *UX_Control_Snapshot_nombre_proyecto*[2]

Si hay alguna fase con error, se creará también el módulo siguiente:

- UX_Snapshot_nombre_proyecto

Mostrará una imagen de la ventana activa (con –ss_config active) del momento del error.

Ejemplo de salida con ejecución correcta:

```
 <module>
        <name><![CDATA[UX_Status_calculadora.sikuli]]></name>
        <type>generic_proc</type>
        <data><![CDATA[1]]></data>
        <description><![CDATA[C:\pandora_ux\calculadora.sikuli       execution
completed
Control snapshot rebuilt
]]></description>
        <tags>UX</tags>
        <module_group>UX</module_group>
</module>
<module>
        <name><![CDATA[UX_Time_calculadora.sikuli]]></name>
```

2 Sólo en la primera ejecución

```
        <type>generic_data</type>
        <data><![CDATA[20.204]]></data>
        <description><![CDATA[C:\pandora_ux\calculadora.sikuli       execution
completed
Control snapshot rebuilt
]]></description>
        <tags>UX</tags>
        <module_group>UX</module_group>
        <module_parent>UX_Status_calculadora.sikuli</module_parent>
</module>
<module>
        <name><![CDATA[UX_Control_Snapshot_calculadora.sikuli]]></name>
        <type>async_string</type>
        <data><![CDATA[
IBCAIAAAAOCnfhAAAAAXNSR…/4x79e/7757f8H2C00s1C73yMAAAAASUVORK5CYII=]]></data>
        <description><![CDATA[Control image rebuilt]]></description>
        <tags>UX</tags>
        <module_group>UX</module_group>
        <module_parent>UX_Status_calculadora.sikuli</module_parent>
</module>
```

Ejemplo de salida con ejecución fallida:

```
<module>
        <name><![CDATA[UX_Status_std.html]]></name>
        <type>generic_proc</type>
        <data><![CDATA[0]]></data>
        <description><![CDATA[Failed to execute verifyText]]></description>
        <tags>UX</tags>
        <module_group>UX</module_group>
</module>
<module>
        <name><![CDATA[UX_Time_std.html]]></name>
        <type>generic_data</type>
        <data><![CDATA[15.463]]></data>
        <description><![CDATA[Failed to execute verifyText]]></description>
        <tags>UX</tags>
        <module_group>UX</module_group>
        <module_parent>UX_Status_std.html</module_parent>
</module>
<module>
        <name><![CDATA[UX_Snapshot_std.html]]></name>
        <type>async_string</type>
        <data><![CDATA[
iVBORw0KGgoAAAANSUhEUgAA…JRU5ErkJggg==]]></data>
        <description><![CDATA[Image (last error)]]></description>
        <tags>UX</tags>
        <module_group>UX</module_group>
        <module_parent>UX_Status_std.html</module_parent>
</module>
```

3.2.2 Ejecución basada en transacción

Si tenemos grabados diferentes procesos con PDR y hemos probado que funcionan al reproducirlos de forma continuada, haremos la ejecución del siguiente modo:

```
pandora_ux_x64.exe -exe C:\PDR\pdr.cmd -args -r -t calculadora_trans -script
C:\PDR\calc.sikuli,C:\PDR\savecalc.sikuli,C:\PDR\savefile.sikuli -folder
C:\PDR\ -ss_config active
```

Como puede observarse, simplemente indicaremos la ruta del nuevo script en el parámetro -script separada por una coma del script anterior y utilizaremos el parámetro -t con el nombre de la transacción que englobará las diferentes fases. Si tuviésemos un proceso con más fases seguiríamos la misma lógica; por ejemplo:

```
pandora_ux_x64.exe -exe C:\PDR\pdr.cmd -args -r -t proceso_transaccional -
script C:\PDR\script1,C:\PDR\script2,C:\PDR\script3,C:\PDR\script4 -folder
C:\PDR\ -ss_config active
```

La línea a añadir al fichero de configuración del agente, para este caso, será la siguiente:

```
module_plugin pandora_ux_x64.exe -exe C:\PDR\pdr.cmd -args -r -t
calculadora_trans -script
C:\PDR\calc.sikuli,C:\PDR\savecalc.sikuli,C:\PDR\savefile.sikuli -folder
C:\PDR\ -ss_config active -checkpoint -post "taskkill /F /IM calc.exe"
```

Gracias al parámetro -checkpoint podremos ver capturas del resultado final de cada fase en la consola de Pandora FMS.

Se devolverán los siguientes módulos por fase[3]:

- UX_Time_nombre_proyecto.fase_#orden
- UX_Status_nombre_proyecto.fase_#orden

Si hay alguna fase con error, se creará también el módulo siguiente:

- UX_Snapshot_nombre_proyecto.fase_#orden

Mostrará una imagen de la web en el momento del error.

También se devolverán los módulos de resumen globales identificados con los siguientes nombres:

- UX_Global_Time_nombre_proyecto

[3] #orden representa el número correspondiente al orden en que se ha declarado (1, 2, 3...)

- UX_Global_Status_nombre_proyecto
- UX_Global_Snapshot_nombre_proyecto

Mostrará una imagen de la web en el momento del error.

Ejemplo de captura de imagen de web con transferencia incompleta:

Observación: Las capturas de error sólo se mostrarán cuando el cliente UX (PWR) y el servidor PWR estén corriendo en la misma máquina. En caso contrario, el directorio de entrega de las imágenes por parte del servidor PWR deberá ser accesible por el cliente para poder mostrar la imagen en Pandora.