Lua App

This chapter provides information about the Lua app for Cisco IP cameras. This app lets an IP camera run a script that is created in the Lua programming language.

This chapter includes these topics:

About the Lua App

The Lua app enables the IP camera to run a script that is created in the Lua programming language. A Lua script provides instructions for the camera about how to behave or operate in certain situations.

Cisco provides sample scripts for the for the Lua app. You can run a sample script as provided, edit and then run a sample script, create and run your own script, or run a script that is provided by a third-party.

Configuring the Lua App on an IP Camera

Before you can use the Lua app, you must configure it on each IP camera on which it will run. To configure this app, perform the following steps.

Before You Begin

Install the Lua app on the IP camera on which it will run. See the "Related Documentation" section on page 1-1 for more information.

Procedure

Step 1 From the IP camera web-based user interface, click the **Setup** link, click **Application Manager** to expand the menu, then click **App Setup**.

Step 2 Click the **luaApp** radio button, then click **Configure**.

The Cisco luaApp configuration page appears.

Step 3 (Optional) Click the **Download** button in the Cisco luaApp configuration page. then use the dialog box that appears to save a copy of the existing script.

This step is useful if you want to save a backup copy of a script before you edit it, or save a copy of a completed script for future reference. You can save existing script where you want.

Step 4 In the script area in the Cisco luaApp configuration page, write a script in the Lua programming language, or paste an existing Lua script and edit it as needed.

Step 5 Click **Save** in the Cisco luaApp configuration page, then click **OK** in the Overwrite dialog box.

The script is saved with the name script.lua. The existing script with that name is overwritten.

Running the Lua App

When you run the Lua app on an IP camera, the camera executes the functionality that the current script.lua Lua script defines.

To run the Lua app on an IP camera, follow these steps:

Procedure

Step 1 From the IP camera web-based user interface, click the **Setup** link, click **Application Manager** to expand the menu, then click **App Setup**.

Step 2 Click the **luaApp** radio button.

Step 3 (Optional) If you want the Lua app to run automatically each time the IP camera reboots, in the Installed Application List area, check the **Start on Boot** check box that corresponds to this app.

If you do not check this check box, you must run the app manually each time the IP camera reboots.

Step 4 Click the **Run** button.

Stopping the Lua App

When you stop the Lua app on an IP camera, the camera stops executing the functionality that the current script.lua Lua script defines.

To stop the Lua app on an IP camera, follow these steps:

Procedure

Step 1 From the IP camera web-based user interface, click the **Setup** link, click **Application Manager** to expand the menu, then click **App Setup**.

Step 2 Click the **luaApp** radio button.

Step 3 Click the **Stop** button.

Lua App Sample Scripts

When you install the Lua app on an IP camera, sample scripts for use with the app are placed in the /usr/apps/luaApp/html folder on the camera. Each of these scripts is written in the Lua programming language.

Table 8-1 describes each script and provide references to sections in this document that show the contents of the scripts.

Table 8-1 Sample Lua Scripts

Script File Name	Description	Reference
hello.lua	Prints “Hello World!” to the app log file	Hello World Script, page 8-3
event.lua	Sends a test event to the event manager	Test Event Script, page 8-3
input_event.lua	Subscribes to the camera input triggers and generates an app trigger when the IP camera receives an input trigger	Send Event on Input Trigger Script, page 8-4
motion_event.lua	Subscribes to the motion trigger and generates an app trigger when the IP camera receives a motion trigger	Send Event on Motion Trigger Script, page 8-4

To access a sample Lua script on an IP camera, use an SSH client to access the camera, log in with the password that is configured on the camera for SSH access, then open the desired log file with a text editor. For more information about accessing a camera via an SSH client, see the Configuration Guide for your IP camera model.

Hello World Script

The sample script named hello.lua sends the text “Hello World!” to the app log file. This script contains the following code:

```
log.info("Hello World!")
```

Test Event Script

The sample named script event.lua sends a test event to the event manager. This script contains the following code:

```
app_event_data = [[
<EventPayload>
        <App>EventTest</App>
        <test>1</test>
</EventPayload>
]]

event.send(app_event_data, 0)
```

Send Event on Input Trigger Script

The sample script named input_event.lua subscribes to the camera input triggers and generates an app trigger when the IP camera receives an input trigger. This script contains the following code:

```
input_trigger = "cisco.input1"
max_events = 10

input_event_template = [[
    <CameraAppEventPayload>
    <vendorName>Cisco Systems,Inc</vendorName>
    <eventName>%s</eventName>
    <eventState>%s</eventState>
    <severity>info</severity>
    <description>Cisco luaApp</description>
    </CameraAppEventPayload> ]]

status = trigger.subscribe(input_trigger)
log.info("Subscribed to "..input_trigger.." with status = "..status)

for i = 1, max_events do
        t = trigger.get()
        input_event = string.format(input_event_template, t:name(), t:state())
        event.send(input_event, 0)
--      log.info(input_event)
end
```

Send Event on Motion Trigger Script

The sample script named motion_event.lua subscribes to the motion trigger and generates an app trigger when the IP camera receives a motion trigger. This script is the default script.lua script and contains the following code:

```
motion_trigger = "cisco.motion"
max_events = 10

motion_event_template = [[
    <CameraAppEventPayload>
    <vendorName>Cisco Systems,Inc</vendorName>
    <eventName>%s</eventName>
    <eventState>%s</eventState>
    <severity>info</severity>
    <description>Cisco luaApp</description>
    </CameraAppEventPayload> ]]

status = trigger.subscribe(motion_trigger)
print("Subscribed to "..motion_trigger.." with status = "..status)

for i = 1, max_events do
        t = trigger.get()
        if t:state() == "start" then
        motion_event = string.format(motion_event_template, t:name(), 1)
        else
        motion_event = string.format(motion_event_template, t:name(), 2)
        end
        print(motion_event)
        event.send(motion_event, 0)
end
```