Explication et solution de contournement sur CSCvt73723 - Le serveur WebRTC fuit les sessions après un grand nombre de sessions placées sur le serveur

Table des matières

Introduction

Conditions préalables

- Exigences
- Composants utilisés

CSCvt73723 - Le serveur WebRTC fuit des sessions après un grand nombre de sessions placées sur le serveur

- Comment identifiez-vous ce bogue ?
- Comment pouvez-vous éviter ce problème ?
- Quand est-il prévu de remédier à cette situation ?

Informations connexes

Introduction

Ce document décrit la détection et la solution de contournement sur l'ID de bogue Cisco CSCvt73723 autour des sessions de fuite du serveur WebRTC après un grand nombre de sessions placées sur le serveur. Cela peut finalement empêcher les utilisateurs de se connecter ou de se joindre en tant qu'invité sur le WebBridge.

Conditions préalables

Exigences

Cisco vous recommande de prendre connaissance des rubriques suivantes :

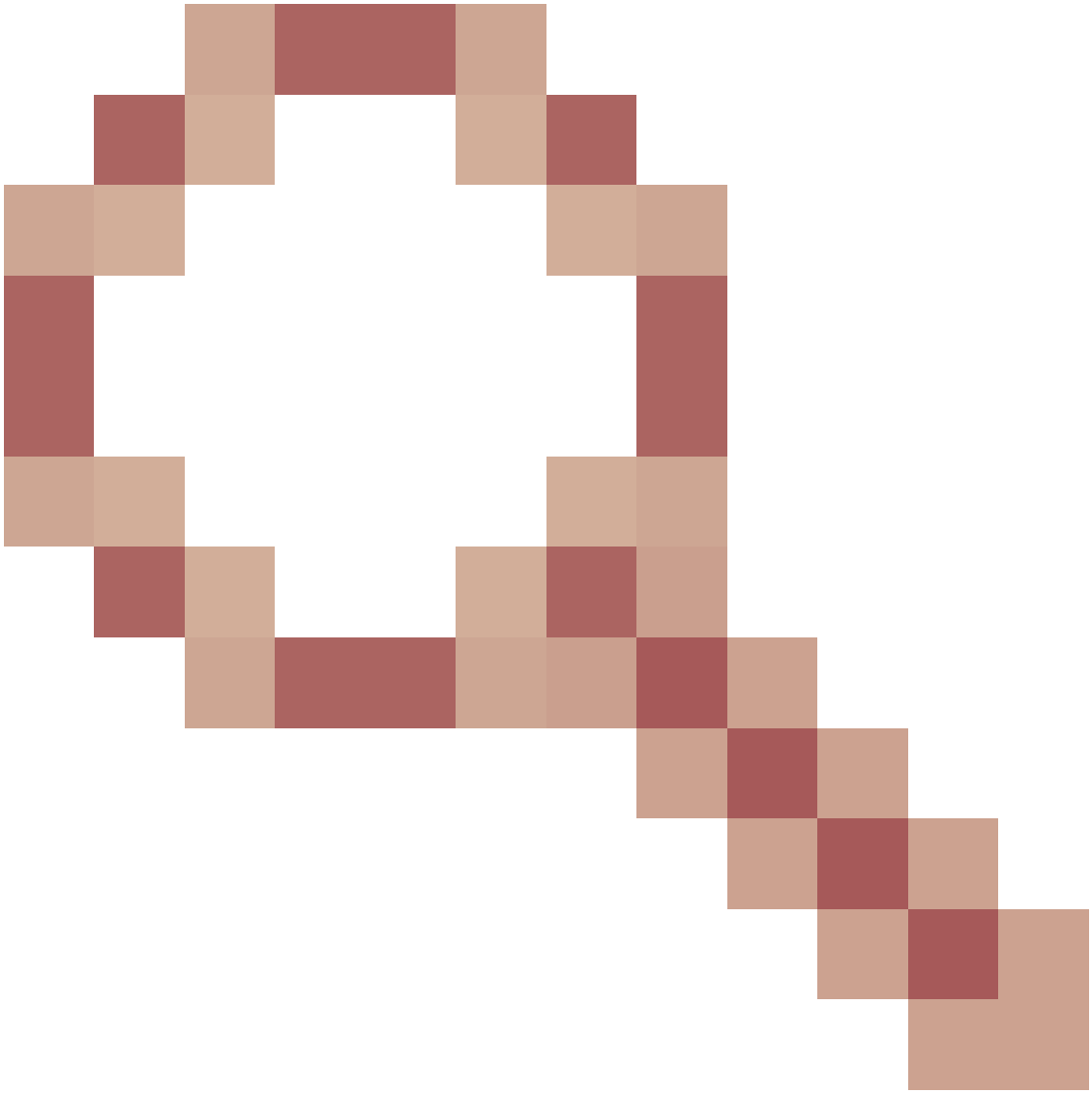
- Cisco Meeting Server (CMS) (composant CallBridge et WebBridge)

Composants utilisés

Les informations contenues dans ce document sont basées sur Cisco Meeting Server et en particulier sur le composant WebBridge 2 / CMA WebRTC. Ce document ne s'applique pas au nouveau composant WebBridge 3 / CMS Web app introduit dans la version 2.9.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau

est en ligne, assurez-vous de bien comprendre l’incidence possible des commandes.



CSCvt73723

- Le serveur WebRTC fuit les sessions après un grand nombre de sessions placées sur le serveur

Comment identifiez-vous ce bogue ?

Du point de vue de l'utilisateur final, le symptôme se manifeste lorsqu'il atteint la limite stricte et qu'aucun autre utilisateur ne peut se joindre à une téléconférence. Dans les journaux, repérer les statistiques de webbridge (comme dans cette FAQ) sont frappant 149 ne signifie PAS nécessairement qu'il s'agit de toutes les sessions divulguées. Cela signifie simplement que le pont Web a atteint sa limite stricte et qu'aucune nouvelle connexion n'est autorisée.

"webbridge" : INFO : [DÉBOGAGE] Stats 149, c:3477, d:3170

Calculer combien de ces sessions sont des fuites est un peu plus compliqué et peut être fait si vous n'utilisez PAS le client de bureau CMA ou le client iOS. À partir de la version 2.8, le pont d'appel signale toutes les 5 minutes le nombre de sessions CMA (CMA WebRTC + CMA client de bureau + CMA client iOS). Notez que ce champ est appelé « CMA » : « X/Y » où X est le nombre

actuel de sessions CMA actives et Y est le pic des 5 dernières minutes.

INFO : STATISTIQUES : {"callLegsPS": 1, "callLegs" : "20/24", "CMA" : "14/17", "sip" : {"std": "0/1", "homologue" : "6/6"}}

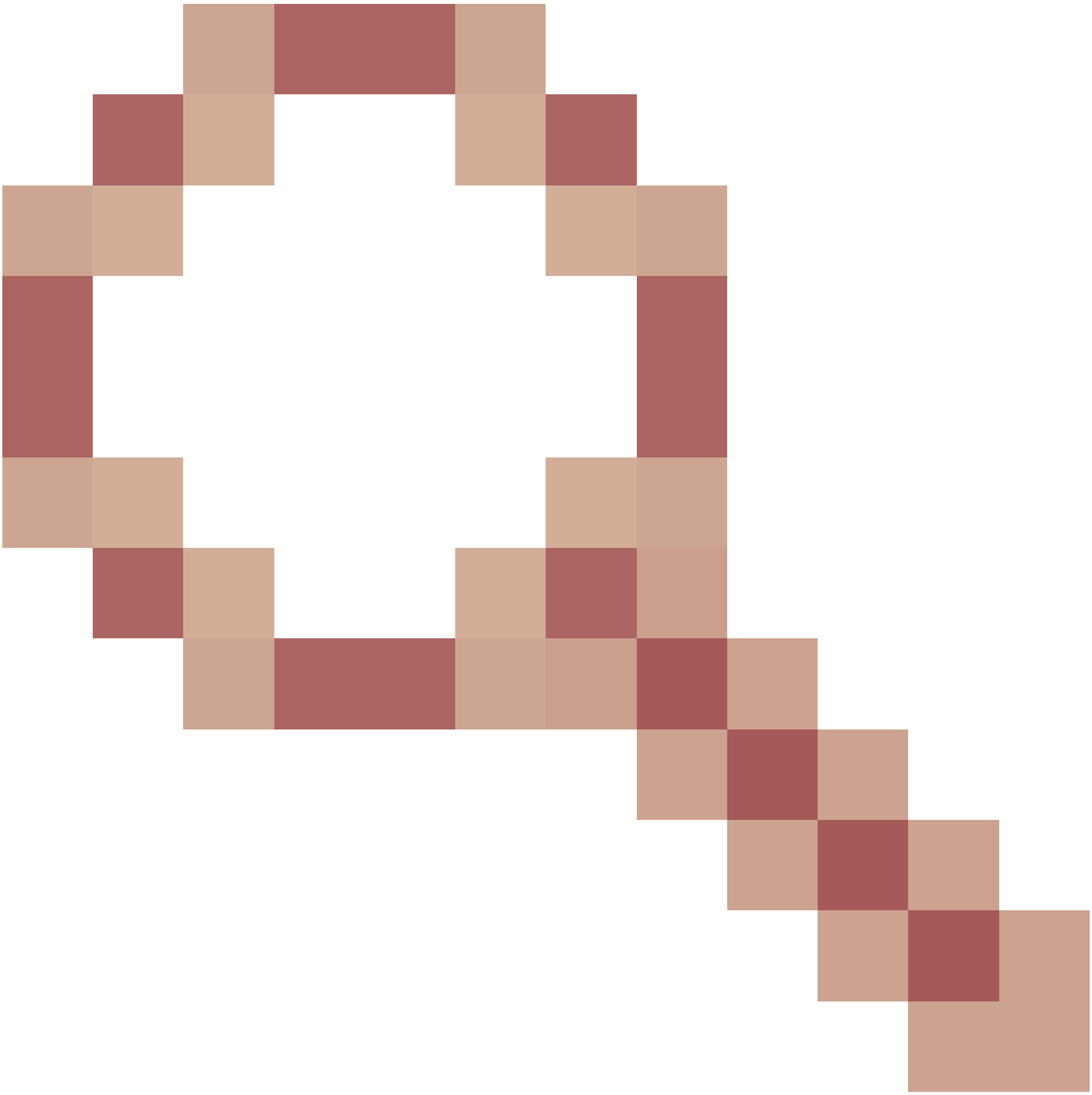
Le simple fait qu'un pont d'appel signale 14 sessions en cours ne signifie pas que le pont Web colocalisé signale également 14 sessions. Ce mappage est 1:1 sur un serveur combiné unique, mais dans un déploiement en cluster, une session Web Bridge peut instancier un appel sur un pont d'appel différent (en particulier lorsque l'équilibrage de charge est activé, ce qui est le cas par défaut pour CMA).

Par conséquent, afin de calculer le nombre total de sessions ayant fuité dans un déploiement, vous avez besoin des sessions actives combinées à partir de TOUTES les statistiques de Web Bridge et comparez ceci avec les statistiques combinées de CMA Call Bridge qui sont rapportées.

Comment pouvez-vous éviter ce problème ?

En fonction de la fréquence à laquelle votre déploiement rencontre cette situation (une fois tous les deux jours ou une fois toutes les deux semaines), vous devez être conseillé de redémarrer leurs ponts Web, ce qui élimine les sessions ayant fui et réinitialise le nombre de sessions actives à 0. Bien entendu, cela peut être fastidieux si cela devient une tâche quotidienne. C'est pourquoi cette tâche peut être facilitée avec un script disponible selon le bloc de code.

```
#################################################################
#### Cisco Meeting Server ####
#### Webbridge restart ####
#### Workaround for CSCvt73723
```



```
####
#### feedback: willwoo@cisco.com ####
#################################################################
#-------------------------------------------------------------
# ---------- DISCLAIMER ----------
#-------------------------------------------------------------
# Please note this script is NOT maintained or supported by Cisco.
# This is to be run at entirely your own risk.
# This script is not intended for redistribution
# Tested with python 3.7.4
#-------------------------------------------------------------
#-------------------------------------------------------------
# ---------- Libraries to import ----------
```

```
#-------------------------------------------------------------
import paramiko
import time
import datetime
#-------------------------------------------------------------
#-------------------------------------------------------------
# ---------- Deployment parameters to change ----------
#-------------------------------------------------------------
# WB Inventory - just extend or modify the below to match your deployment requirements.
# Enter the MMP IP of the server (can differ from interface webbridge service is running)
webbridges ={1:"127.0.0.1",2:"127.0.0.1",3:"127.0.0.1",4:"127.0.0.1"}

mmp_username = "admin" # MMP username
mmp_password = "password" # MMP password
#-------------------------------------------------------------
 def mmp_webbridge_restart(mmp_address,uname,pword):
 conn = paramiko.SSHClient()
 conn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
 try:
 conn.connect(mmp_address, 22, uname, pword)
 stdin, stdout, stderr = conn.exec_command('webbridge restart')
 time.sleep(1)
 conn.close()
 print_log_message('Webbridge on server: ' + mmp_address + ' restarted successfully')
 except Exception as error:
 print_log_message('Failed to restart webbridge on server ' + mmp_address + '. Error:')
 print_log_message(str(error))
 pass
 def print_log_message(message):
 time_stamp = datetime.datetime.now(datetime.timezone.utc)
 time_stamp = str(time_stamp)
 file = open('webbridge_restart_logs.txt', 'a')
 file.write(time_stamp + " " + message + "\n")
 file.close()
 if __name__ == '__main__':
 for wb in webbridges:
 mmp_webbridge_restart(webbridges[wb], mmp_username, mmp_password)
#####################################################################
```

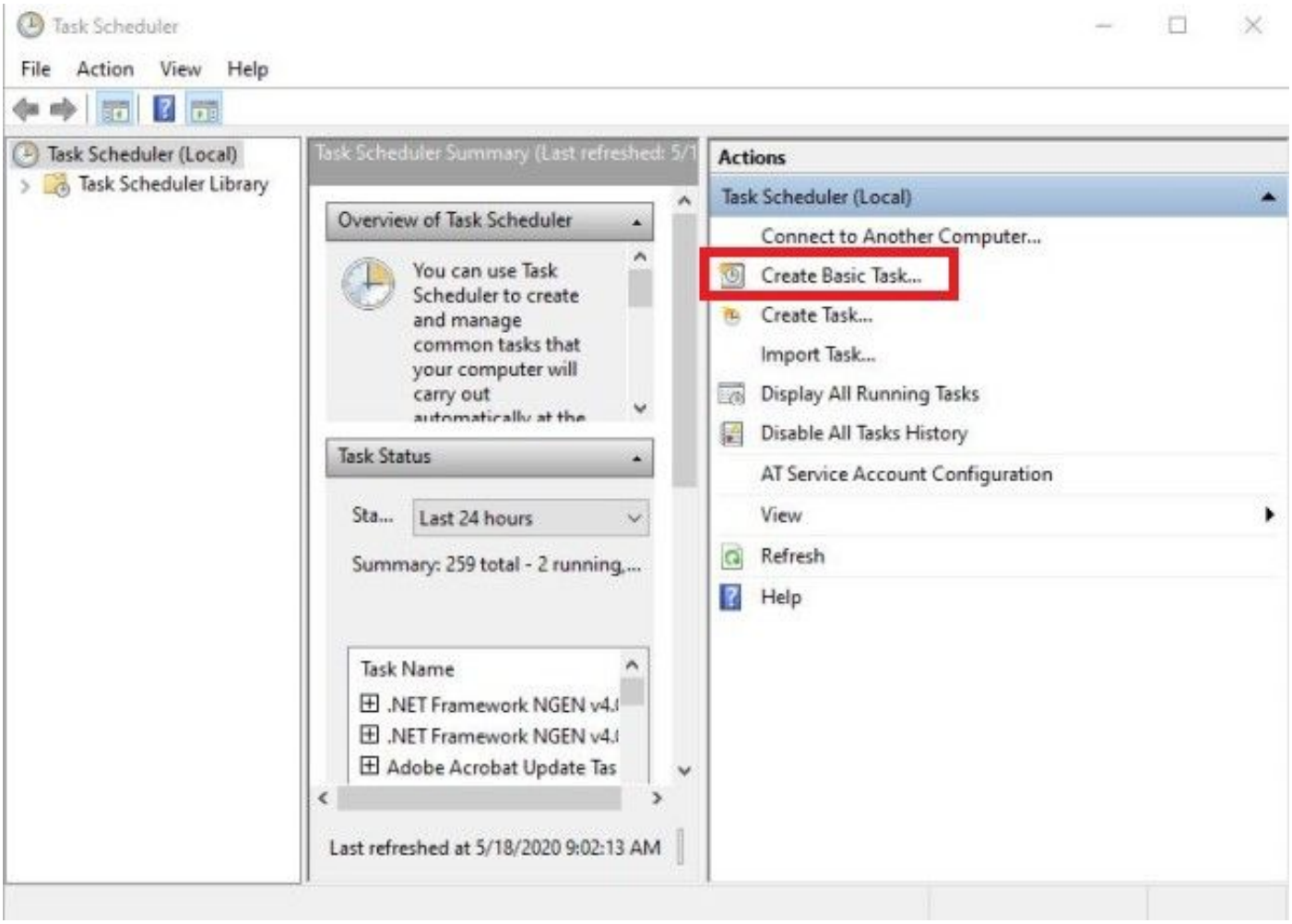
Le script nécessite quelques petites modifications (les informations d'identification sur la ligne 29-30 et les adresses IP des ponts Web dans le déploiement sur la ligne 27) et doit être exécuté UNIQUEMENT quand il n'y a aucune charge attendue ou pendant une fenêtre de maintenance. Le script ne vérifie pas les sessions actives et exécute simplement la commande « webbridge restart » sur tous les serveurs répertoriés, ce qui ne termine aucune session WebRTC active.

Pour automatiser ce script, vous pouvez le faire en configurant une tâche cron ou sur un PC

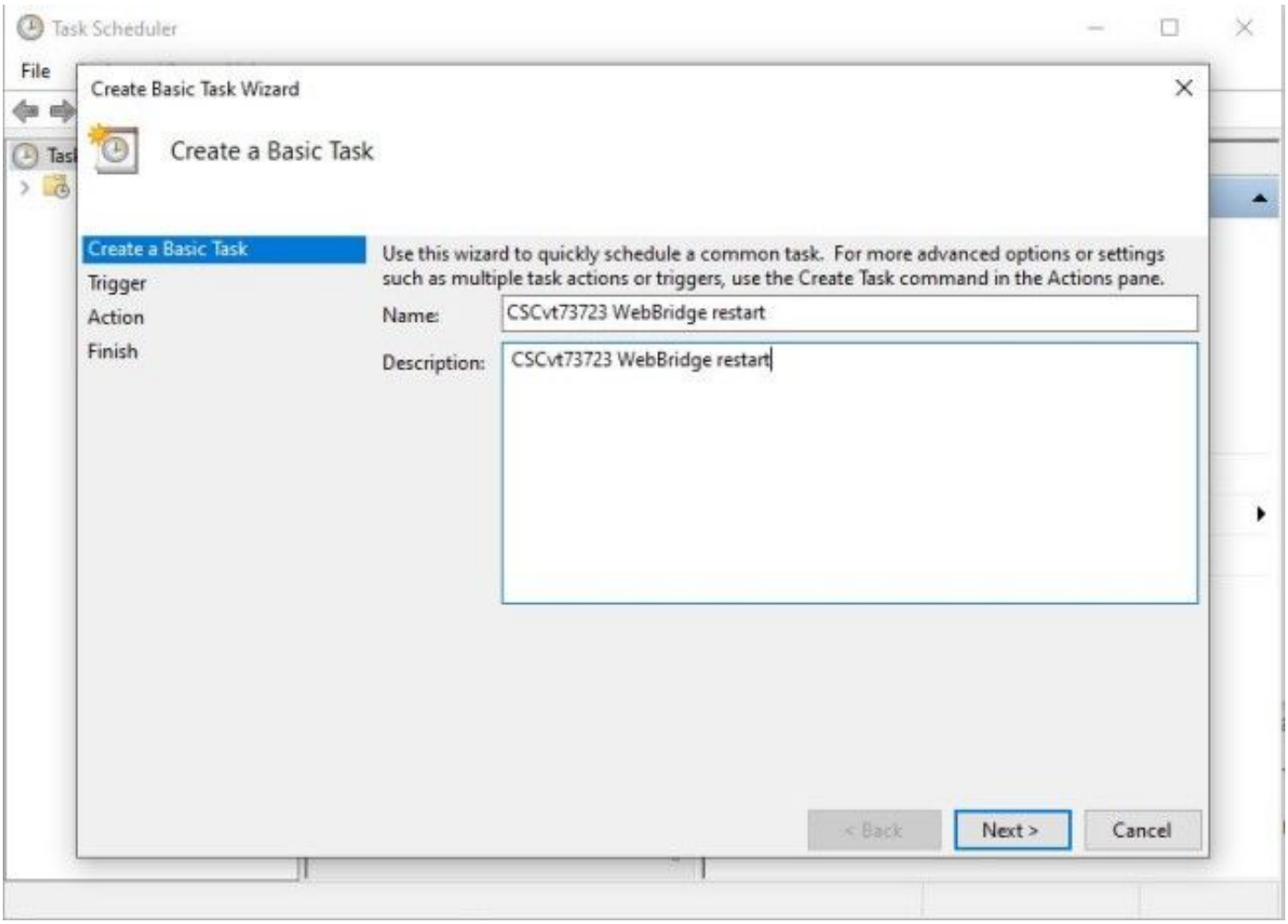
Windows 10 avec le Planificateur de tâches. En supposant que le PC Win 10 ait Python 3.4+ installé, ils peuvent suivre ces étapes :

1. Ouvrir le Planificateur de tâches

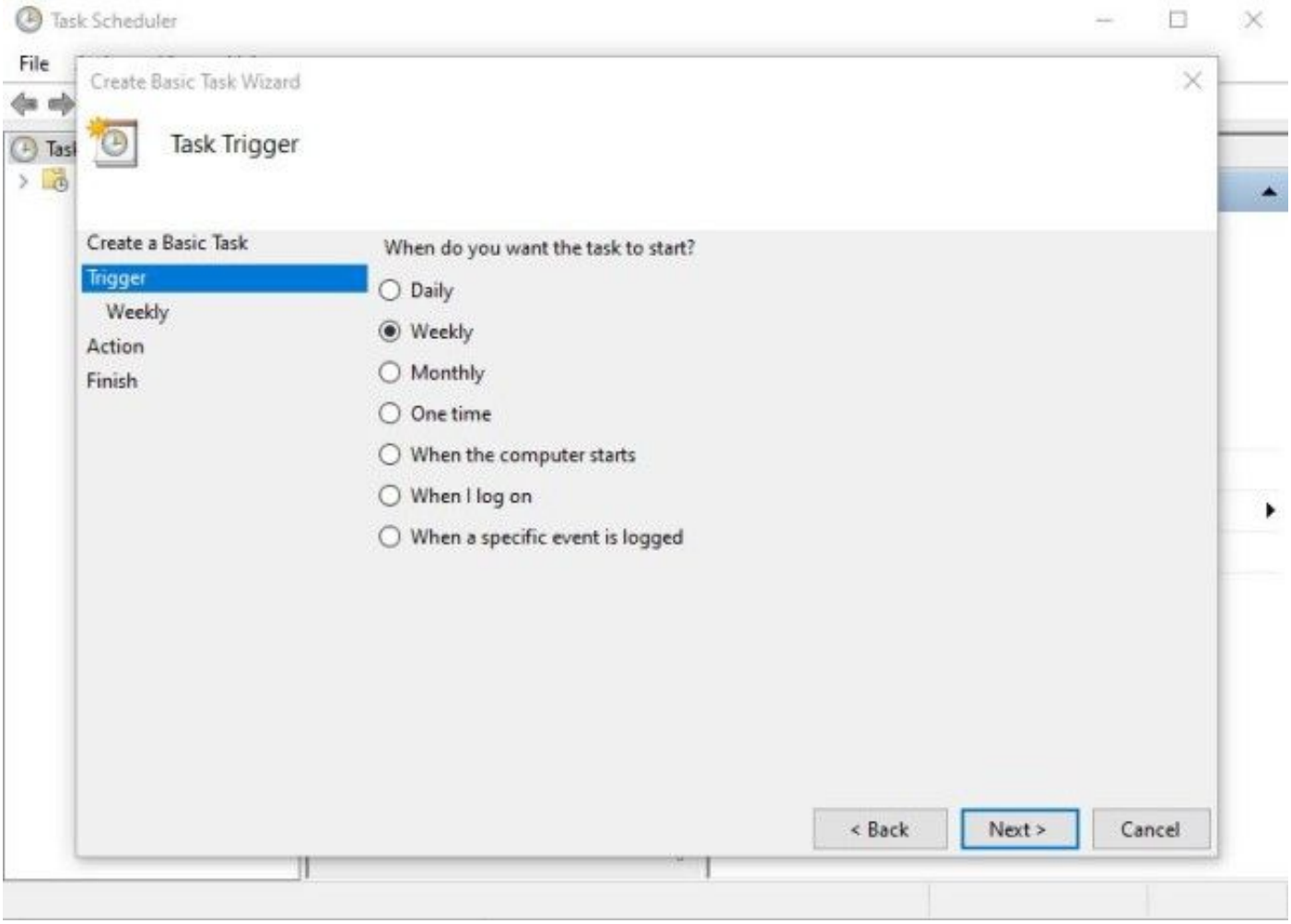
2. Sélectionnez « Créer une tâche de base ».

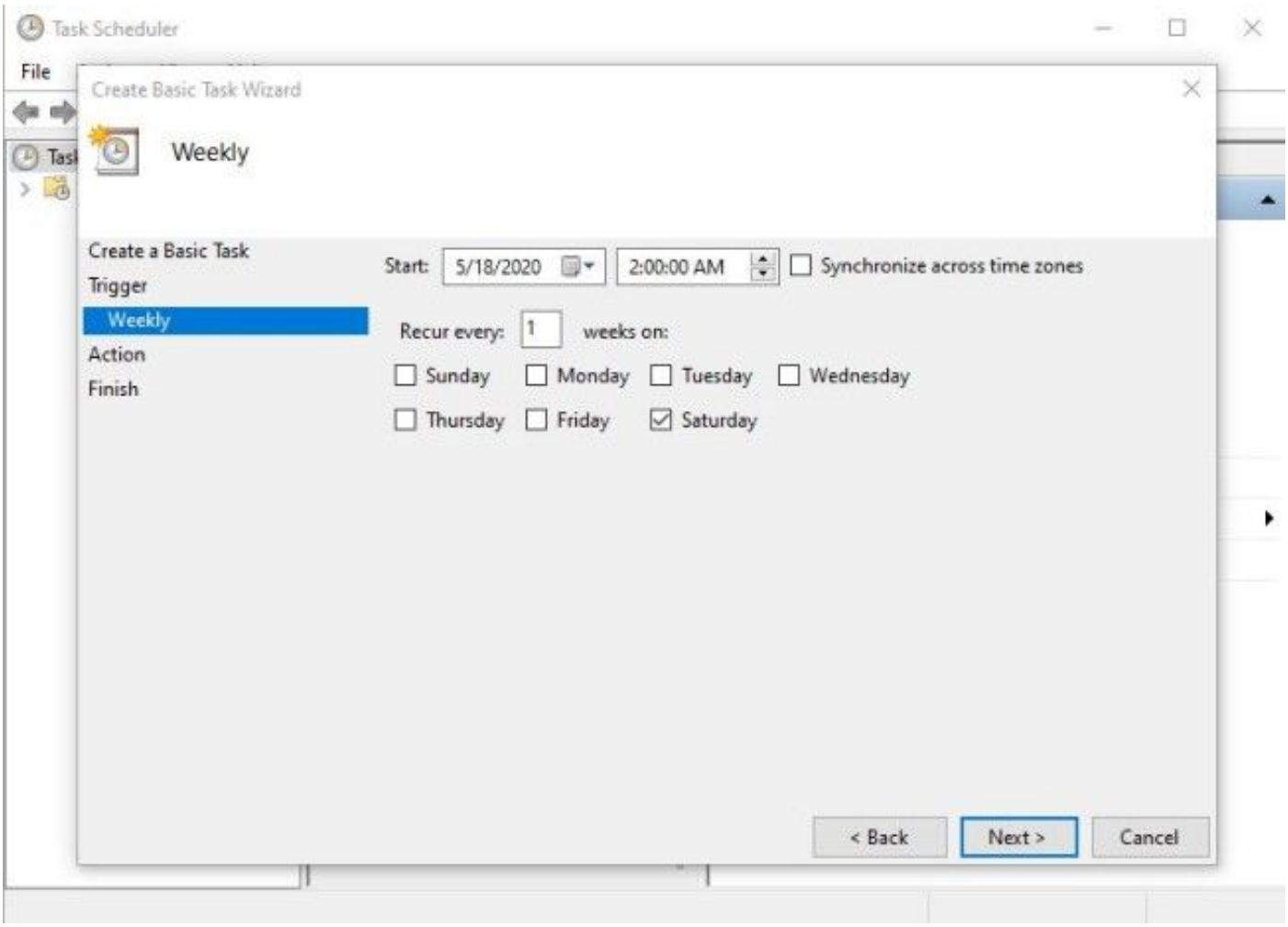


2.1 Entrez un nom/une description pour cette tâche

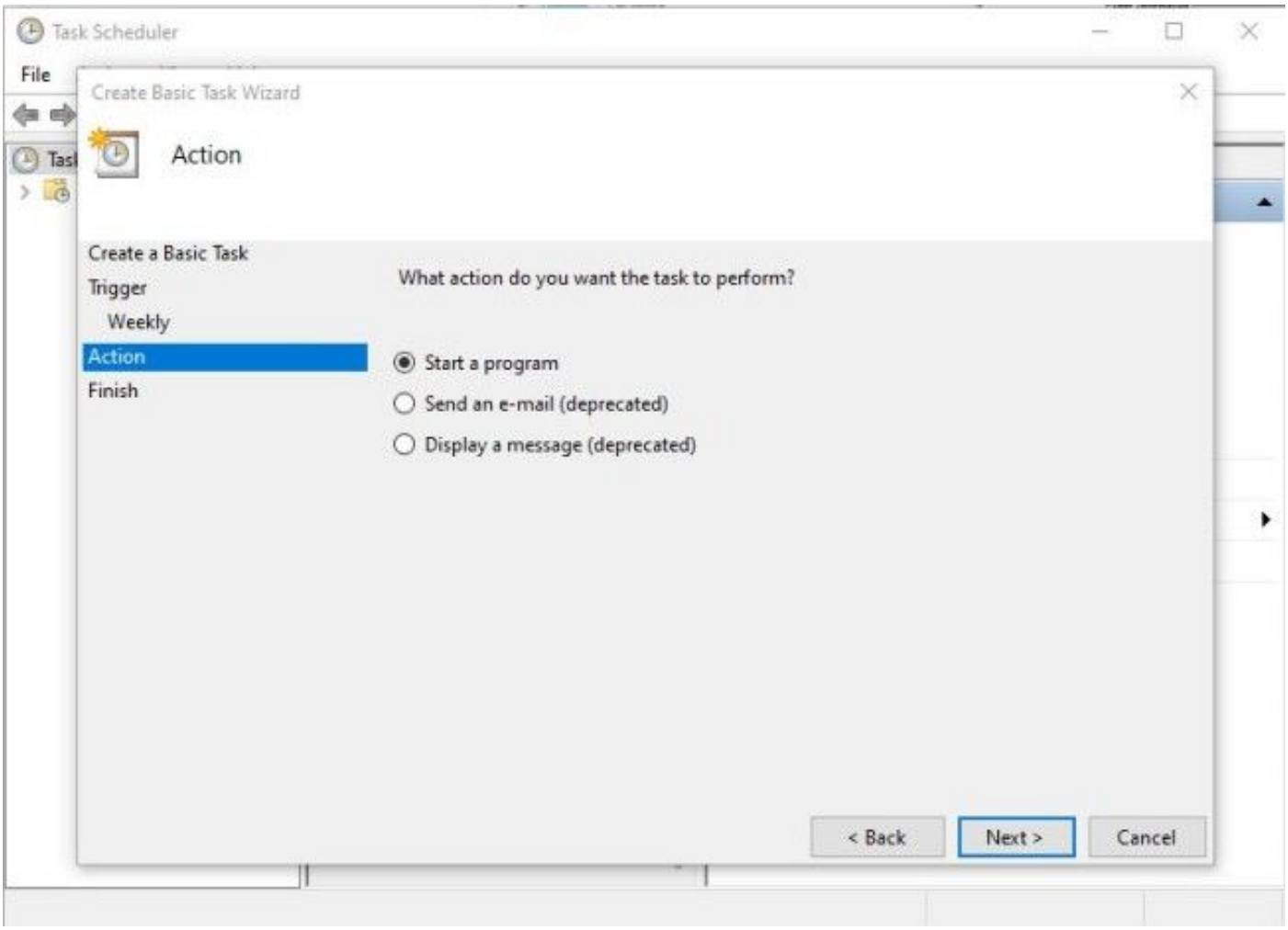


2.2 Sélectionnez la fréquence et les heures d'exécution de cette tâche (recommandé uniquement en dehors des heures de pointe, comme indiqué ici pour chaque samedi à 2 heures du matin)


Task Scheduler
File
Create Basic Task Wizard
Task Trigger
Create a Basic Task
Trigger
Weekly
Action
Finish
When do you want the task to start?
Daily
Weekly
Monthly
One time
When the computer starts
When I log on
When a specific event is logged
< Back
Next >
Cancel



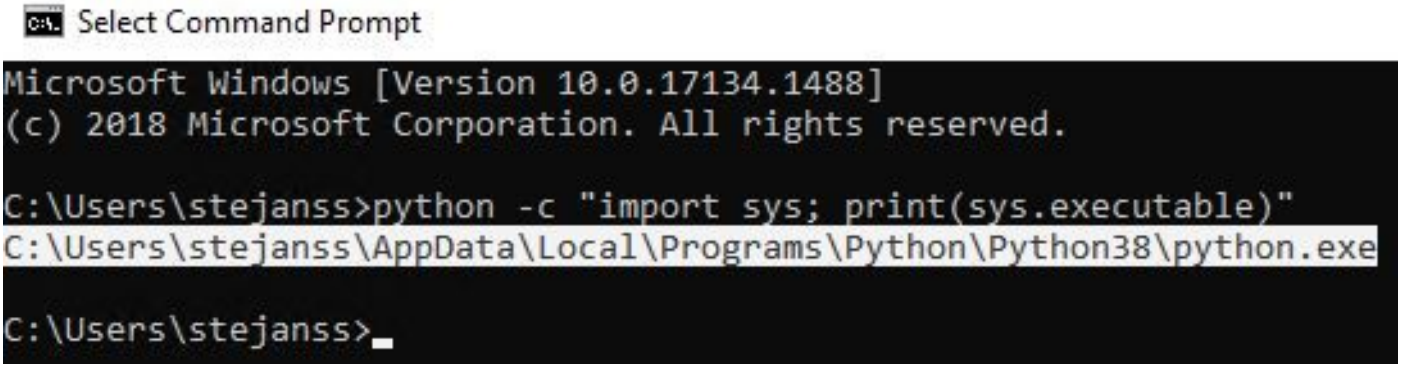
2.3 Action à effectuer, sélectionnez : 'Démarrer un programme'



2.4 Mesures :

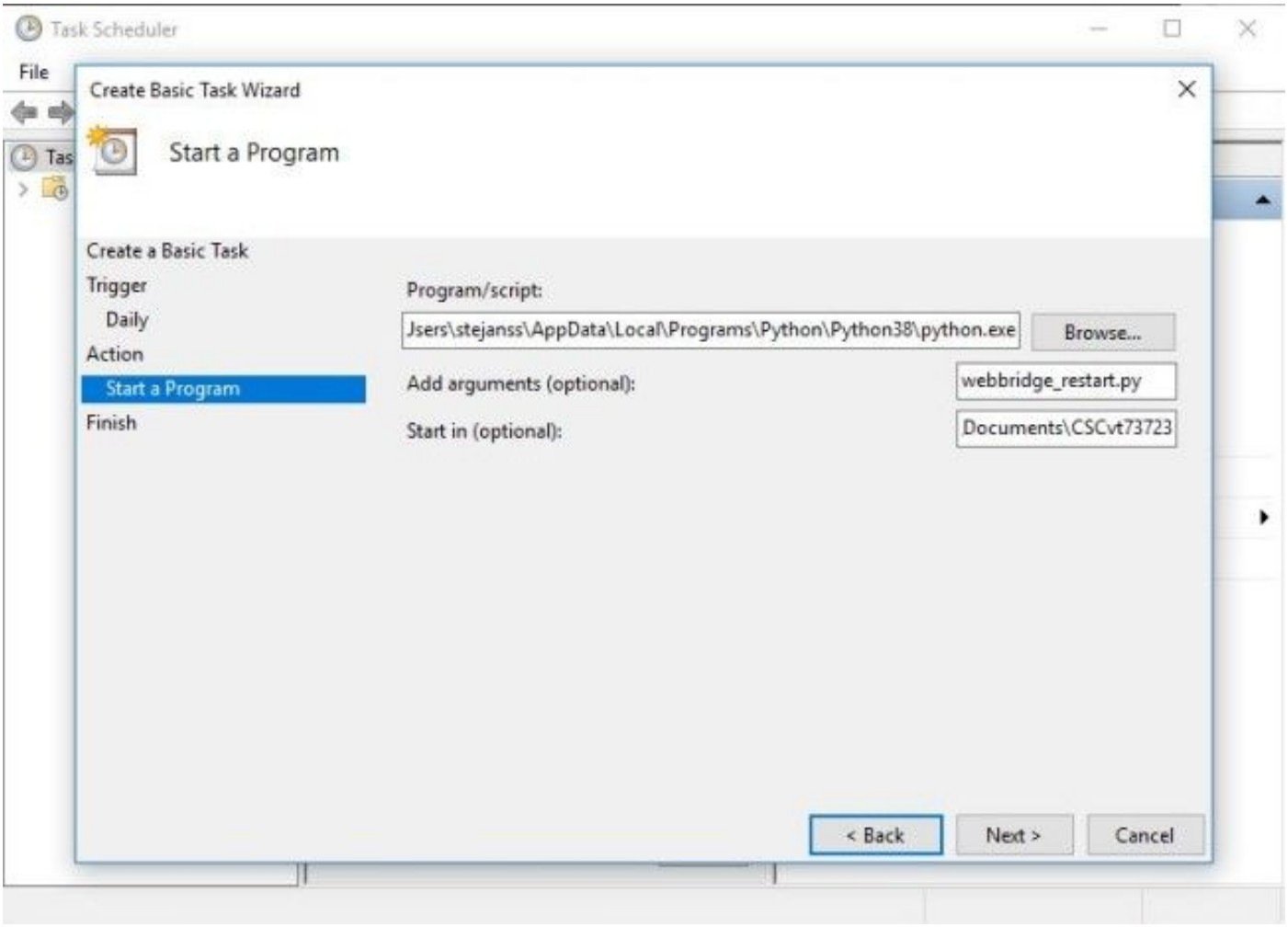
* Programme / Script : C:\<chemin vers python.exe>

(si vous ne connaissez pas le chemin d'accès à python.exe, vous pouvez le trouver en accédant à cmd et en tapant : python -c "import sys ; print(sys.executable)")



* Ajouter des arguments (facultatif) : webbridge_restart.py (ou nom du script python)

* Commencer dans (facultatif) : C:\<chemin vers webbridge_restart.py>



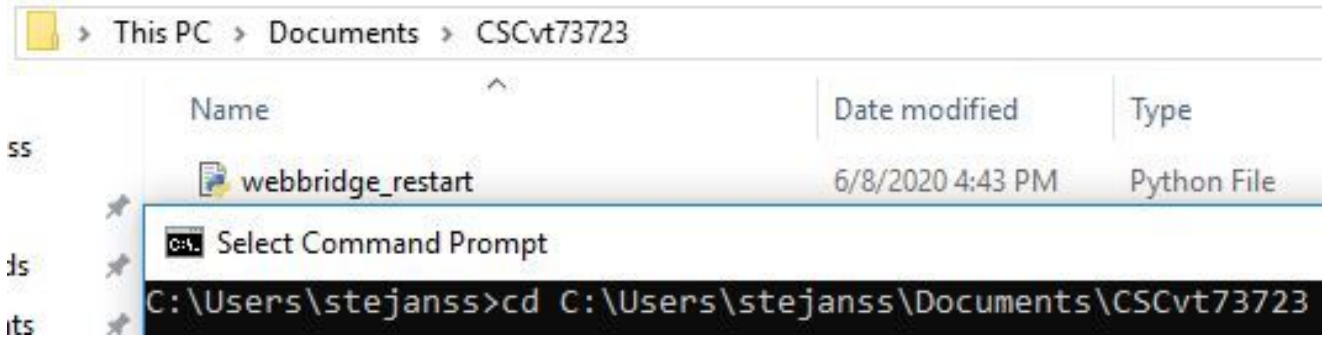
Notez que l'ordinateur qui exécute la tâche cron doit pouvoir accéder au MMP des serveurs CMS configurés. Une fois le script exécuté, il crée un fichier webbridge_restart_logs.txt qui contient des détails sur les redémarrages des différents WebBridges, ainsi que sur les échecs potentiels. Un exemple est illustré avec une connexion réussie à 10.48.79.194 et une connexion défaillante à 127.0.0.1 (comme étant l'adresse de bouclage du PC en fait).

2020-06-08 14:53:18.149915+00:00 Webbridge on server: 10.48.79.194 restarted successfully 2020-06-08 14:53:19.165543+00:00 Failed to restart webbridge on server 127.0.0.1. Error: 2020-06-08 14:53:19.165543+00:00 [Errno None] Unable to connect to port 22 on 127.0.0.1

Comment vérifier que le script fonctionne correctement ?

Si Python a installé le PC à partir duquel vous avez l'intention d'exécuter le script, vous pouvez d'abord l'exécuter manuellement en suivant les étapes suivantes :

1. Ouvrez cmd et accédez à l'emplacement du script à l'aide de la commande « cd »



2. Exécutez le fichier python avec la commande 'python webbridge_restart.py'

```
C:\Users\stejanss\Documents\CSCvt73723>python webbridge_restart.py
Traceback (most recent call last):
  File "webbridge_restart.py", line 18, in <module>
    import paramiko
ModuleNotFoundError: No module named 'paramiko'
```

3. Si vous voyez une erreur indiquant que le module 'paramiko' n'est pas installé, vous devez installer une bibliothèque supplémentaire avec la commande 'pip install paramiko'

```
C:\Users\stejanss\Documents\CSCvt73723>pip install paramiko
Collecting paramiko
  Downloading https://files.pythonhosted.org/packages/06/1e/1e08b
aaaf6c3d3df1459fd85f0e7d2d6aa916f33958f151ee1ecc9800971/paramiko-
2.7.1-py2.py3-none-any.whl (206kB)
    |                                | 215kB 1.3MB/s
Collecting cryptography>=2.5 (from paramiko)
```

4. Une fois terminé, vous pouvez relancer le script avec 'python webbridge_restart.py' (REMARQUE : cela redémarre le webbridge et entraîne la déconnexion des connexions WebRTC en cours)

   S'il s'est exécuté correctement, vous pouvez en vérifier le résultat dans le fichier webbridge_restart_logs.txt.

Quand est-il prévu de remédier à cette situation ?

Il ne s'agit pas d'un nouveau bogue et il n'y a aucun plan pour le corriger sur le Web Bridge 2 / CMA WebRTC. La nouvelle application Web Bridge 3 / CMS (disponible à partir de la version 2.9) n'est pas affectée par ce bogue car elle a été complètement repensée. Les clients fortement touchés par cette situation doivent envisager de passer à la nouvelle application Web CMS (bien qu'il ne s'agisse pas encore de la parité de fonctionnalités avec Web Bridge 2 dans la version 2.9). Consultez les notes de version de CMS 2.9 et de l'application web CMS pour plus de détails.)

Informations connexes

- Notes de défauts : https://bst.cloudapps.cisco.com/bugsearch/bug/CSCvt73723
- Notes de version: https://www.cisco.com/c/en/us/support/conferencing/meeting-server/products-release-notes-list.html
- Forum aux questions
    - https://meeting-infohub.cisco.com/faq/content/43/450/en/how-do-you-check-the-number-of-active-sessions-on-a-web-bridge.html

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.