Remote Custom API Server Configuration

Overview

Remote execution of custom code facilitates the execution of custom code and libraries in the remote server outside VXML Server. This feature allows the separation of core IVR application (business logic) and extended business logic (custom code not shipped with the Call Studio application) and operates on a distinct instance that is not shared by Call or VXML Server. This improves system stability and performance because the fundamental services are functioning exclusively for their respective applications. This in turn provides the sufficient resources and reduces the application instability caused by excessive resource utilization. A component is introduced in Call Studio to facilitate the communication and separation between external applications and core applications.

The table below provides the list of elements and whether those elements have the remote execution option or not. For more information on the configuration details, see the Programming Guide for Cisco Unified CVP VXML Server and Cisco Unified Call Studio.

Element Type	Element Name	Remote Execution Option
Audio	Audio	No
	Custom_Audio	Yes
Call Control	Transfer	No
Cisco	ReqICMLabel	No

Element Type	Element Name	Remote Execution Option
Callback	Callback_Add	No
	Callback_Disconnect_Caller	
	Callback_Enter_Queue	
	Callback_Get_Status	
	Callback_Reconnect	
	Callback_Set_Queue_Defaults	
	Callback_Update_Status	
	Calback_Validate	
	Callback_Wait	
	Callback_Ready	
Commerce	Currency	No
	Currency_with_confirm	
Context	Application_Modifier	No
Date & Time	Date	No
	Date_With_Confirm	
	Time	
	Time_With_Confirm	
Form	Form	No
	Custom Form	Yes
	Form_With_Confirm	No
	Custom Form_With_Confirm	Yes
Integration	Database	Yes
	FTP_Client	No
	REST_Client	No
Math	Counter	No
	Math	
	Set Value	Yes

Element Type	Element Name	Remote Execution Option
Menu	Yes_No_Menu	No
	Custom Yes_No_Menu	Yes
	2_Option_Menu	No
	Custom 2_Option_Menu	Yes
	3_Option_Menu	No
	Custom 3_Option_Menu	Yes
	4_Option_Menu	No
	Custom 4_Option_Menu	Yes
	5_Option_Menu	No
	Custom 5_Option_Menu	Yes
	5_Option_Menu	No
	Custom 4_Option_Menu	Yes
	6_Option_Menu	No
	Custom 6_Option_Menu	Yes
	7_Option_Menu	No
	Custom 7_Option_Menu	Yes
	8_Option_Menu	No
	Custom 8_Option_Menu	Yes
	9_Option_Menu	No
	Custom 9_Option_Menu	Yes
Notification	Alert	No
	Email	

Element Type	Element Name	Remote Execution Option
Number Capture	Digits	No
	Custom Digits	Yes
	Digits_With_Confirm	No
	Custom Digits_With_Confirm	Yes
	Number	No
	Custom Custom Number	Yes
	Number_With_Confirm	No
	Custom Number_With_Confirm	Yes
	Phone	No
	Custom Phone	Yes
	Phone_With_Confirm	No
	Custom Phone_With_Confirm	Yes
Record	Record	No
	Record_With_Confirm	
Video	Video Connect	No
Virtual Agent	Dialogflow	No
	DialogflowCX	
	DialogflowIntent	
	DialogflowParam	
	Transcribe	
	VirtualAgentVoice	
Wxm	WxM_PCS	No
Say it Smart Plugin	Say it Smart Plugin	Yes
Logger	Remote Custom Logger	Yes

Installation and Configurations

Set Up Remote Server

Before you begin

You must set up the Remote Server VM on Windows OS similar to the VXML Server OVA configuration.

Procedure

Step 1 In the Remote Server VM, install OpenJDK 8 (version 1.8.0_271 or higher).

Download OpenJDK at: https://www.openlogic.com/openjdk-downloads?field_java_parent_version_target_id=416&field_operating_system_target_id=436&field_architecture_target_id=All&field_java_package_target_id=All.

Step 2 Configure the `JAVA_HOME` environment variable under **System Variables**, with the respective Java installed path. For example, the path may be `C:\Program Files\OpenLogic\jdk-8.0.372.07-hotspot`.

Step 3 Install Apache Tomcat 9 (version 9.0.60 or higher).

Download Tomcat at: https://tomcat.apache.org/download-90.cgi.

Step 4 Stop the Tomcat server.

Step 5 Copy the `customapis.war` file to the `webapps` folder of Tomcat (for example, `C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps`).

Note The spring boot SDK `customapis.war` file is located at `%CVP_HOME%\util\remoteexecution` folder of the VXML Server. To run the `customapis.war` file, you need Apache Tomcat as the web server.

Step 6 Start the Tomcat server. A folder named `customapis` is created in `%Apache Software Foundation%\Tomcat 9.0\webapps`.

Templates for `web.xml` and `server.xml` are bundled in the `customapis` folder in the following locations:

- `\customapis\WEB-INF\classes\tomcatConfig\conf\web.xml`
- `\customapis\WEB-INF\classes\tomcatConfig\conf\server.xml`

Use these files only as a reference and configure them properly according to your requirements.

The existing `web.xml` and `server.xml` files are available at the following locations:

- `%Apache Software Foundation%\Tomcat 9.0\conf\server.xml`
- `%Apache Software Foundation%\Tomcat 9.0\conf\web.xml`

You can replace or modify the above files.

Running Custom Code Using Remote Server

Procedure

Step 1 Bundle all the custom code that you want to run remotely in a `.jar` file.

Step 2 Copy the `.jar` file and all the dependencies to the `%Apache Software Foundation%\webapps\customapis\WEB-INF\lib` folder.

Step 3 Restart the Tomcat server.

The HTTP port listens at **8080** and the gRPC port listens at **8090**.

You can change the port for gRPC in the `application.properties` file located at `%Apache Software Foundation%\Tomcat 9.0\webapps\customapis\WEB-INF\classes`.

Name of the property:

```
#server.grpc.port =8090
```

You can change the port for HTTP in the `server.xml` file located at `%Apache Software Foundation%\Tomcat 9.0\conf`.

Name of the property:

```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443"
maxParameterCount="1000"
 />
```

For secure connection, configure the port **8080** in the `server.xml` file. For more information, see Enable Security over HTTP (Self-Signed Certificate) in Remote Server.

Note To confirm if the application is running, check the spring boot starter logs in the `cvp.log` file located in the `%Apache Software Foundation%\logs` folder.

Step 4 Check the status of the application at: *http://remote_ip:8080/customapis/actuator/health*. **UP** status denotes that the application is running.

Step 5 To check if the gRPC server is up and running, run the command **netstat -a | findstr 8090**.

Step 6 Configure the **dynamic configuration** and **remote execution** URLs for the Call Studio application in the **Remote URL Settings** tab and redeploy the application.

Step 7 Restart the VXML Server for the changes to be effective.

Step 8 To run the loggers remotely, see the *Loggers* chapter in the Programming Guide for Cisco Unified CVP VXML Server and Cisco Unified Call Studio.

Step 9 In a failure scenario:

a) Check the error logs in VXML Server: `%CVP_HOME%\VXMLServer\applications\{App name}\logs\ErrorLog`.

b) To check the lifecycle of an element, check the activity log: `%CVP_HOME%\VXMLServer\applications\{App name}\logs\ActivityLog`.

Remote Server Application Properties

The following table lists the properties in the `application.properties` file that is located at `%Apache Software Foundation%\Tomcat 9.0\webapps\customapis\WEB-INF\classes`.

Property Name	Usage
loggerPath = C:\\<any folder>\\CustomLogger	Specifies the path for running the application loggers in the remote server.
server.grpc.port = 8090 server.grpc.keyStorePath = C:\\Program Files\\OpenLogic\\jdk-8.0.372.07-hotspot\\jre\\lib\\security\\cacerts	Specifies the port and keystore path configured for gRPC.
server.grpc.keyStoreType = JCEKS	Specifies the KeyStoreType for gRPC connection.
server.grpc.keyAlgorithm = SunX509	Specifies the key algorithm used.
server.grpc.transport = TLS	Specifies the incoming secure protocol.
server.grpc.outgoing.secure.Transport = TLS	Specifies the outgoing secure protocol.
server.grpcciphers = TLS_RSA_WITH_AES_128_CBC_SHA	Colon (;) seperated secure ciphers, for example TLS_RSA_WITH_AES_128_CBC_SHA.
server.grpctlsv1dot2Enabled = true	Secure TLS versions flags, for example TLSv1.
server.grpc.protocol = TLS	Specifies the secure protocol used.
server.grpc.useClientAuth = true	Specifies whether a client certificate is needed or not.
server.grpc.enableRemoteAuthentication = false	Specifies whether gRPC authentication is needed or not.
server.grpc.maxAllowedRequests = 1000	Specifies the maximum allowed calls at a time.
restapi.security.enabled = false	Specifies whether HTTP authentication is needed or not.

Heartbeat Settings in VXML Server

The heartbeat mechanism monitors each remote endpoint URL, be it HTTP or RPC.

For the End point heartbeat control following properties have been added in the `vxml.properties` .

These three properties have been added in the `%CVP_HOME%\conf\vxml.properties`.

- VXML.EndpointHeartbeatEnabled = true
- VXML.EndpointPingInterval = 30000
- VXML.EndpointMaxPingFailure = 1

After you update the `vxml.properties` file, restart the VXML Server.

Configuring HTTP Proxy Settings in VXML Server

To configure HTTP proxy settings in VXML Server, perform the following steps:

Procedure

Step 1 Open Windows Registry Editor (regedit) in the VXML Server.

Step 2 Go to `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Apache Software Foundation\Procrun 2.0\VXMLServer\Parameters\Java\Options`.

Step 3 Add the following entries:

- `Dhttps.proxyHost=<proxy-server hostname or fqdn or IP>`
- `Dhttps.proxyPort=<port>`

Step 4 Restart the Unified CVP VXML Server from Windows services.

Firewall Port Settings

To allow remote execution of custom code, add the following ports to the firewall exclusion list:

- HTTP port - 8080
- gRPC port - 8090

Security Configuration

This section covers the steps which need to be configured for enabling a secure connection between the remote server and VXML Server.

Authentication for Remote Server

Create Credentials for Authentication in Remote Server

Before you begin

It is necessary to create credentials for the remote server before activating authentication on the remote server. Use the **add-user-credentials** API both for creating and updating the credentials.

Procedure

In a REST client, for example Postman, enter the following details to create credentials:

a. In the *POST* request, add the URL *http://<remote_machine_IP>:8080/customapis/actionapi/add-user-credentials*.

In the above URL, replace remote server, IP address, and port as needed.

b. In **Request Body**, add *userid* and *secret* as key-value. Make sure the provided *userid* and *secret* to be non-null values.

c. In the **Headers** tab, you must enable **Content-Type** as *application/x-www-form-urlencoded*.

Note After authentication is enabled, you must change the credentials. Use basic authentication if the credentials were updated in the same POST request and the username and password were changed previously.

Enable gRPC Authentication in Remote Server

Procedure

Step 1 Log in to the remote server machine.

Step 2 Navigate to `%Apache Software Foundation%\Tomcat 9.0\webapps\customapis\WEB-INF\classes`.

Step 3 Open the `application.properties` file and set the *server.grpc.enableRemoteAuthentication* flag to true.

Step 4 Restart Apache Tomcat Server.

Enable HTTP Authentication in Remote Server

Procedure

Step 1 Log in to the remote server machine.

Step 2 Navigate to `%Apache Software Foundation%\Tomcat9.0\webapps\customapis\WEB-INF\classes`.

Step 3 Open the `application.properties` file and set the *restapi.security.enabled* flag to true.

Step 4 Restart Apache Tomcat Server.

Enabling Authentication in Call Studio and VXML Server

Procedure

Step 1 Go to the Call Studio application.

Step 2 Right-click on the application, which needs to be in secure connection, and select **Properties**.

Step 3 In the **Call Studio Properties** tab, click **Remote URL Settings** .

Step 4 Choose **HTTP** or **RPC** as required.

Step 5 Provide the user ID and password.

Step 6 Save and deploy the Call Studio application and restart the VXML Server.

Secure Connection Setup Between Remote Server and VXML Server

Create Keystore Password for Remote Server

Before you begin

Before activating the security, you must create the remote server keystore password.

Procedure

In a REST client, for example Postman, enter the following details to create credentials:

a. In the *POST* request, add the URL *http://<remote_machine_IP>:8080/customapis/actionapi/add-keystore-password*.

In the above URL, replace remote server, IP address, and port as required.

b. In **Request Body**, add *keyStorePassword* as the *key-value*. Ensure that the provided *keyStorePassword* is a non-null value.

c. In the **Headers** tab, you must enable **Content-Type** as *application/x-www-form-urlencoded*.

Note After authentication is enabled, you must change the credentials. Use basic authentication if the credentials were updated in the same POST request and the username and password were changed previously.

Generate Self-Signed Certificate for Remote Custom Server HTTP or gRPC

Before you begin

It is recommended to change the default keystore password (which is usually 'changeit'). To change the password, run the command: **%JAVA_HOME%\jdk-8.0.372.07-hotspot\jre\bin>keytool -storepasswd -new <newpassword> -keystore ..\lib\security\cacerts -storetype JCEKS -storepass <defaultpassword>**

Procedure

Step 1 Open the command prompt and execute the following command in `cmd.exe` with proper alias: **%JAVA_HOME%\jdk-8.0.372.07-hotspot\jre\bin>keytool -genkey -keyalg RSA -alias customcode_certificate -keystore ..\lib\security\cacerts -storetype JCEKS -keysize 2048**

Step 2 Once you execute the command, answer the questions.

For example:

```
Enter keystore password: *************
What is your first and last name?
  [Unknown]:  CustomRemoteServer----------------→ The CN should same as the FQDN of the
machine
What is the name of your organizational unit?
  [Unknown]:  CCBU
What is the name of your organization?
  [Unknown]:  CISCO
What is the name of your City or Locality?
  [Unknown]:  KA
What is the name of your State or Province?
  [Unknown]:  BLR
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=CustomRemoteServer, OU=CCBU, O=CISCO, L=KA, ST=BLR, C=IN correct?
  [no]:  yes
Enter key password for <customcode_certificate>
        (RETURN if same as keystore password):
done
```

Step 3 Now, you can import the self-signed certificate.

Generate Remote Server ECDSA Certificate with Open SSL

Before you begin

The Remote Server enables a variant of the Digital Signature Algorithm that are known as an Elliptic Curve Digital Signature Algorithm (ECDSA). Remote Server supports either ECDSA or RSA. RSA remains the default cryptography algorithm. However, looking at the requirements, we can enable or disable ECDSA.

For disabling ECDSA, you must delete the existing ECDSA aliases and generate RSA certificates again.

Procedure

Step 1 Download OpenSSL (64-bit) and install on your remote computer.

Step 2 Add OpenSSL bin path to the Windows environment path variable.

For example, `path=C:\Program Files\OpenSSL-Win64\bin`

Step 3 Go to `C:\Cisco\CVP\conf\security`

Step 4 From the command prompt, run the following command to generate the private keys for the remote server: **openssl ecparam -name prime256v1 -genkey -noout -out remoteserver-private-key.pem**

Step 5 Run the following command to generate the self-signed certificates for the remote server: **openssl req -new -key remoteserver-private-key.pem -x509 -nodes -days 365-out remoteserver-cert.pem**

Step 6 Enter the values for the following fields when prompted:

```
Country Name (2 letter code) []: < >
State or Province Name (full name) []: < >
Locality Name (for example, city) []: < >
Organization Name (for example, company) []: < >
Organizational Unit Name (for example, section) []: < >
Common Name (for example, server FQDN or your name) []: < >
Email Address []: < >
Please enter the following extra attributes to be sent with your certificate request:
```

```
A challenge password []: < >
An optional company name []: < >
```

Note Enter a period (.) to leave the following fields blank:

- **Common Name**
- **Email Address**
- **Challenge password**
- **An optional company name**

You can generate a certificate after entering all the details.

Step 7 Run the following command to append the keys and certificates in one file: **cat remoteserver-private-key.pem remoteserver-cert.pem > remoteserver-certificate-private.pem**

Step 8 Run the following command to export the certificates to the Remote server: **openssl pkcs12 -export -inkey remoteserver-private-key.pem -in remoteserver-certificate-private.pem -out cert_remoteserver.p12 -name remoteserver_certificate**

```
Enter Export Password:<CVP keystore password>
Verifying - Enter Export Password:<CVP keystore password>
```

Step 9 Go to `C:\Cisco\CVP\conf\security` and run the following command to delete the existing RSA certificates for the remote server: `C:\Cisco\CVP\jre\bin\keytool.exe -storetype JCEKS -keystore .keystore -delete -alias remoteserver-certificate -storepass <CVP keystore password>`

Step 10 Run the following command to import the ECDSA certificates to the keystore:
`C:\Cisco\CVP\jre\bin\keytool.exe -v -importkeystore -srckeystore cert_remoteserver.p12 -srcstoretype PKCS12 -destkeystore .keystore -deststoretype JCEKS -alias remoteserver_certificate`**Importing keystore cert_remoteserver.p12 to .keystore...**

```
Enter destination keystore password:
Enter source keystore password:
[Storing.keystore]
```

Step 11 Restart the remote server.

Step 12 In the new browser tab, type the following and download the certificates: `https://<remote ip>:8080`

Enable Security over gRPC (Self-Signed Certificate) in Remote Server

Procedure

Step 1 Log in to the remote server machine.

Step 2 Make sure the the self-signed certificate for the remote server machine is generated.

Step 3 Navigate to the `%Apache Software Foundation%\Tomcat 9.0\webapps\customapis\WEB-INF\classes` directory.

Step 4 Launch the `application.properties` file and set the **server.grpc.secure** flag to true.

Step 5 Provide the `.keystore` path for the `server.grpc.keyStorePath` flag.

For example: `C:\\Program Files\\OpenLogic\\jdk-8.0.372.07-hotspot\\jre\\lib\\security\\cacerts` (make sure that keystore exists in that machine).

Step 6 Restart the Remote Apache Tomcat server.

Enable Security over HTTP (Self-Signed Certificate) in Remote Server

Procedure

Step 1 Log in to the remote server machine.

Step 2 Ensure that the self-signed certificate for the remote server machine is generated.

Step 3 Navigate to the `%Apache Software Foundation%\Tomcat 9.0\webapps\customapis\WEB-INF\classes` directory.

Step 4 Open the `application.properties` file, set the **restapi.security.enabled** flag to true, and save the file.

Step 5 Navigate to the `%Apache Software Foundation%\Tomcat 9.0\conf` directory.

Step 6 Open the `server.xml` file and provide the connector with the port number, and mention the respective keystore password.

For Example:

```
<Connector SSLEnabled="true" acceptCount="1500"
ciphers="TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA256"

clientAuth="false" disableUploadTimeout="true" enableLookups="false"
executor="tomcatThreadPool" keyAlias="customcode_certificate"
keystoreFile="C:\Program Files\Java\jre1.8.0_271\lib\security\cacerts" keystorePass="changeit"
 keystoreType="JCEKS" maxHttpHeaderSize="8192"
port="8080" protocol="org.apache.coyote.http11.Http11NioProtocol" scheme="https" secure="true"

sslImplementationName="org.apache.tomcat.util.net.jsse.JSSEImplementation" sslProtocol="TLS"
 sslEnabledProtocols="TLSv1.2"/>
```

Step 7 Save the `server.xml` file and restart the remote Apache Tomcat server.

Import Self-Signed Certificate of Remote Server in VXML Server for HTTP or gRPC

To import the certificate:

Procedure

Step 1 Launch the URL `https://<remote_server_IP>:8090/` to download the certificate.

Step 2 Upload the certificate in the `%CVP_HOME%\conf\security` folder where the VXML Server is hosted.

Step 3 Import the certificate using the following command:
%CVP_HOME%\jdk-8.0.372.07-hotspot\jre\bin\keytool.exe -import -trustcacerts -keystore %CVP_HOME%\conf\security\.keystore -storetype JCEKS -alias <any_alias> -file %CVP_HOME%\conf\security\<FQDN_remote_server.cer>

Where, `<FQDN_remote_server.cer>` is the certificate downloaded in step 1, which is named after the FQDN of remote server.

Note Use FQDN instead of IP address for a secure gRPC connection. The common name (CN) of the certificate should be same as the address mentioned in the studio settings.

Step 4 Enter the keystore password when prompted.

Run the **DecryptKeystoreUtil.bat** file located at `%CVP_HOME%\bin` to view the keystore password.

Step 5 Make sure that the connection type in the Call Studio application is changed to HTTP or gRPC accordingly. Also, ensure the **Secure Connection** checkbox is enabled.

Step 6 Save and deploy the Call Studio application and restart the VXML Server.

Enable Secure Connection in Call Studio and VXML Server

Before you begin

Make sure that the Call Studio console is used to establish the necessary secure connection.

Procedure

Step 1 Go to the Call Studio application.

Step 2 Right-click on the application, which needs to be in secure connection, and select **Properties**.

Step 3 In the **Call Studio Properties** tab, click **Remote URL Settings** .

Step 4 Choose **HTTP** or **RPC** as required

Step 5 Click the **Secure Connection** checkbox to provide a secure connection.

Note You must provide FQDN for HTTP and gRPC connection type in the address field for secure connection.

Step 6 For FQDN, add **<IP><space><FQDN(hostname)>** in the `%drivers%\etc\hosts` file in the VXML Server.

Step 7 Save and deploy the application. Restart the VXML Server.

Note The same port cannot be used simultaneously in several applications as secure and non-secure ports.

For example: All the applications utilizing a port on the VXML Server for gRPC must be re-deployed with the same secure or non-secure configurations if the port is used for both secure and non-secure calls.

Note When several remote servers or load end points are added for load balancing, all of those servers must have the same secure or non-secure configurations when used simultaneously. This is because all the added servers have the same remote URL settings.

(Optional) Enabling Mutual TLS for gRPC and HTTP in Remote Server

Follow the steps for each VXML Server.

Procedure

Step 1 Import the certificate for VXML Server in the `RemoteServer java keystore`.

Step 2 For gRPC, in the `application.properties` file, available at `%Apache Software Foundation%\Tomcat 9.0\webapps\customapis\WEB-INF\classes`, change the following property to true:

```
server.grpc.useClientAuth = true
```

Step 3 For HTTP, in the `server.xml` file, available at `%Apache Software Foundation%\Tomcat 9.0\conf\server.xml`, change the following property to true:

```
"clientAuth" flag = true
```

Step 4 Restart the Apache Tomcat server.

Monitoring and Serviceability

Spring boot provides enhanced serviceability and has a set of APIs for monitoring and serviceability.

HTTP port listens at **8080** and gRPC port listens at **8090**.

Spring Boot Starter Actuator: *http://<remote_IP>:<Port>/customapis/actuator*. This API provides monitoring facilities around the services:

- **beans**: *http://<IP>:<Port>/customapis/actuator/beans*
- **health-path**: *http://<IP>:<Port>/customapis/actuator/health/{*path}*
- **health**: *http://<IP>:<Port>/customapis/actuator/health*
- **info**: *http://<IP>:<Port>/customapis/actuator/info*
- **shutdown**: *http://<IP>:<Port>/customapis/actuator/shutdown*
- **loggers**: *http://<IP>:<Port>/customapis/actuator/loggers*
- **loggers-name**: *http://<IP>:<Port>/customapis/actuator/loggers/{name}*
- **heapdump**: *http://<IP>:<Port>/customapis/actuator/heapdump*
- **thread dump**: *http://<IP>:<Port>/customapis/actuator/threaddump*
- **Prometheus**: *http://<IP>:<Port>/customapis/actuator/prometheus*
- **metrics-requiredMetricName**: *http://<IP>:<Port>/customapis/actuator/metrics/{requiredMetricName}*
- **metrics**: *http://<IP>:<Port>/customapis/actuator/metrics*
- **scheduled tasks**: *http://<IP>:<Port>/customapis/actuator/scheduledtasks*

To check the health of the application, access the URL: *http://remote_ip:8080/customapis/actuator/health*. The **UP** status denotes that the application is running.

To check the memory used by the application, access the URL: *http://remote_ip:8080/customapis/actuator/metrics/jvm.memory.used*.

To check the CPU usage, access the URL: *http://remote_ip:8080/customapis/actuator/metrics/process.cpu.usage*.

To check the current number of live threads, access the URL: *http://remote_ip:8080/customapis/actuator/metrics/jvm.threads.live*.

To analyze the state of all the threads of an application at a given time, access the URL: *http://remote_ip:8080/customapis/actuator/threaddump*.

A heap dump is a snapshot of all the objects that are in memory in the JVM at a certain moment. It is useful for troubleshooting memory-leak problems and optimising memory usage in Java applications.

To download the heap dump, access the URL: *http://remote_ip:8080/customapis/actuator/heapdump*.

HTTP and gRPC Statistics

To know the statistics of HTTP and gRPC, use the following actuator URL:

/prometheus: *http://remote_ip:8080/customapis/actuator/prometheus*.

HTTP Statistics

HTTP statistics shows the summary of the requests handled along with type, status, and other attributes.

Example

```
# TYPE http_server_requests_seconds summary
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator",}
 1.0
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator",}
 1.095763557
```

gRPC Statistics

gRPC statistics mentioned below represents the total count of gRPC requests handled with attributes. It shows that there is one bidirectional streaming request (BIDI_STREAMING) for the **RemoteVoiceService** method of **com.audium.core.protobuf.RemoteExecutionService** and the response code is "OK".

Example

```
# HELP grpc_server_handled_total Total number of RPCs completed on the server, regardless
of success or failure.
# TYPE grpc_server_handled_total counter
grpc_server_handled_total{grpc_type="BIDI_STREAMING",grpc_service="com.audium.core.protobuf.RemoteExecutionService",
grpc_method="RemoteVoiceService",code="OK",grpc_code="OK",} 1.0
```

VXML Server Statistics

Use the following URL for gRPC registry collection statistics: *http://<IP/hostname of VXML Server>:7000/CVP/Server?stats=true*.

SNMP and Syslog Alerts

The Remote Server application displays SNMP and Syslog alerts when it encounters an exception or error. The alerts contain messages along with the stack trace for monitoring and serviceability of the application.

Following is the sample syslog and SMNP alerts displayed when the class name is not valid.

Syslog Alert

```
98: IP: Oct 05 2023 10:45:40.726 -0700:
%com.cisco.ccbu.infra.serviceability.ServiceabilityManager_VXML-3-VXML_SERVER_SYSTEM_ERROR:
  In application TestSNMP encountered SYSTEM_ERROR_EVENT with message:
'com.cisco.cvp.customaction' is not a valid action element class.
<011 = Level: ERR - error conditions (3)>
```

SNMP Alert

```
Trap OID - 1.3.6.1.4.1.9.9.590 In application TestSNMP encountered SYSTEM_ERROR_EVENT with
 message: 'com.cisco.cvp.customaction' is not a valid action element class.
SNMP    880    trap iso.3.6.1.4.1.9.9.590
```

Similar messages are logged for other scenarios encountered after SNMP and Syslog are properly configured for the VXML Server used.

Logging

VXML Server Configuration for Remote Server

For enhanced VXML logging for custom code, make the required configuration in the VXML Server.

In the `log4j_vxml.xml` file located at `%CVP_HOME%\conf\`, navigate to the **logger** section and add the following **AsyncLogger** tag.

```
<AsyncLogger name="com.cisco.cvp.callserver" level="info" additivity="false">

    <AppenderRef ref="rootUniversalAppender" />

</AsyncLogger>

<AsyncLogger name="io.grpc" level="info" additivity="false">

    <AppenderRef ref="rootUniversalAppender" />

</AsyncLogger>

<AsyncLogger name="com.cisco.cvp.ivr" level="info" additivity="false">

    <AppenderRef ref="rootUniversalAppender" />

</AsyncLogger>
```

You can set the level to **debug** or **info** according to the requirement of logging level.

The log files monitored are:

- `%CVP_HOME%\logs\VXML\CVP <timestamp>.log`
- `%CVP_HOME%\logs\VXML\ERROR <timestamp>.log`

To monitor the health of **RPC end point status**, check the logs in the `VXML\CVP <timestamp>.log` file.

Sample Log

```
RpcEndPoint-6-com.cisco.cvp.callserver.grpc.endpoint.RpcEndPoint:  status of
healthcheckgrpc.health.v1.HealthCheckResponse.ServingStatus.SERVINGEndPoint=
url=<FQDN>/<IP>:8090, statusUrl=cvv, key=<FQDN>:8090:null, status=true
```

Remote Server Configuration for Logging

In the `log4j2.xml` file located at `%Apache Software Foundation%\Tomcat 9.0\webapps\customapis\WEB-INF\classes`, change the level of logging from **info** to **debug** for enhanced logging.

```
<Logger name="com.cisco.cvp.customapi" level="debug"
additivity="false">
<AppenderRef ref="LogToFile" />
</Logger>
```

Log file monitored: `%Apache Software Foundation%\Tomcat 9.0\logs\cvp.log`.