[AUD SERVER] API

Start Release No : 7.0.500.20250213-11 / Update Release No : 7.0.500.20251002-11 : / :

(2026.01.29) Sample

- https .

- : _API sh .zip_20251002-11 / _ _SAMPLE.json_20251002-11

(2025.09.24) Sample (Release No. 7.0.500.20251002-11)

- UserComponent .

- Application Token , (SSH Key) .

- : _API sh .zip_20250924 / _ _SAMPLE.json_20250924

1.
   a. Admin / UI Rest API . (.)
   b. AUD Platform API Application Access Token . (: **Application Access Token**)
   c. API DB .

2. **API**

   a. / Json ( )
      i. / : _ _**SAMPLE.json**
   b. / (3-a)
      i. {Context Path}/WEB-INF/classes/matrix/**matrix_cm.properties** , API / , Json

   [5. API sh] , .

3. **API**

   a. **matrix_cm.properties ( )**

      - : {AUD Context Path}/WEB-INF/classes/matrix/matrix_cm.properties
      - matrix.cm.properties.extend true matrix.cm.properties.extend.path .
      i.

matrix.cm.properties.extend		matrix_cm.properties (: false, : true / : false)	false
matrix.cm.properties.extend.path		,	C:/AUDPlatform_7/conf/matrix
sql.fetch.size		SQL fetch size (: 1000)	1000
matrix.cm.outer		(: true, : false / : false)	true
matrix.cm.outer.path		ROOT	C:/AUDPlatform_7/reports/cm/ext
matrix.cm.outer.tableinfo.path		ROOT	C:/AUDPlatform_7/reports/cm/ext/tableinfo

   b. **( )**

      i. CM_SAMPLE.json , matrix_cm.properties > matrix.cm.outer.tableinfo.path . ( )
      ii. .

   c. **API**

      - : MTX_CM_EXT_LIST
      -

Column Name	Description	Type	Length	Null	P.K	default	
SEQ		NUMBER	-	N	1	-	-
API_URL	URL	VARCHAR2	500	N	-	-	-
REQUEST_DATE		TIMESTAMP	-	-	-	-	-
REQUEST_PARAM		CLOB	-	-	-	-	-
RESULT_STATUS		VARCHAR2	256	-	-	-	-
RESULT_MSG		VARCHAR2	1000	-	-	-	-
REQUEST_IP	IP	VARCHAR2	100	-	-	-	-

      i. Admin > Repository > MTX_CM_EXT_LIST > [Repository Script] .
      ii. API , matrix_service.log .

4. **API**
   a. URL: {AUD Context Path}/api/cm/backup, restore

API	API URI	METHOD					
	api/cm/backup	POST	reportCode	String	Y		{
			tableInfoFileName	String	Y	json	"reportCode": "REP05B91F3B73F1416096B4D004741C0A08" , "tableInfoFileName": "CM_SAMPLE.json" }
	api/cm/restore	POST	restoreFileName	String	Y	( )	{
			authRestore	String		(: false)	"restoreFileName": "REP05B91F3B73F1416096B4D004741C0A08" , "authRestore": "true" }

5. **API sh ()**
   1) API Java sh .
   2)
   - Java .
   - sh API . (**[4.API]** .)
   - sh Java Java .
   - Java Application Access Token / API .
   4) , .

   (Release No. 7.0.500.20251002-11)
   a. **sh**
      i. : ./backup.sh {} {}
      ii. backup.sh JAVA_HOME Java

Sample_backup.sh

```
#!/bin/sh

#
#    JAVA
JAVA_HOME="/home/AUDPlatform/apps/openjdk"

rm -rf CmBackupApi.class
$JAVA_HOME/bin/javac CmBackupApi.java -encoding UTF-8
chmod -R 755 *

#
# 1.    (  .zip  )
# 2.   json
$JAVA_HOME/bin/java CmBackupApi $1 $2
```

      iii. CmBackupApi.java AUD_SERVER_URL, AUD_SECRET_KEY, AUD_SECRET_KEY_FILE_PATH, AUD_AP_ID

Sample_CmBackupApi.java

```
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import java.io.BufferedReader;
import java.io.IOException;
```

```
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.Signature;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class CmBackupApi {
    /**
     * AUD_SERVER_URL: AUD
     * AUD_SECRET_KEY: SSH Key
     * AUD_SECRET_KEY_FILE_PATH: private_key.pem
     * AUD_AP_ID: Ap token    (AUD    )
     */
    private static final String AUD_SERVER_URL = "http://192.168.0.59:8087";
    private static final String AUD_SECRET_KEY = "";
    private static final String AUD_SECRET_KEY_FILE_PATH = "";
    private static final String AUD_AP_ID = "";

    public static void main(String[] args) {
        if (args.length != 2) {
            System.err.println("backup parameter check please....");
            System.exit(1);  //   1
        }

        try {
            // Ap token
            String apAccessToken = getAPToken();

            //     .
            String backupReportCode = args[0];
            String tableInfoFileName = args[1];
            String data = "{\"reportCode\" : \"" + backupReportCode + "\" , \"tableInfoFileName\" : \"" + tableInfoFileName + "\"}";
            try {
                String returnData = execute("/api/cm/backup", data, "POST", apAccessToken);
                System.out.println("======backup data : " + returnData);

                //    api
                String returnLogout = execute("/api/logout", "", "GET", apAccessToken);
                System.out.println("======returnLogout data : " + returnLogout);
            } catch (Exception ex) {
                ex.printStackTrace();
                throw ex;
            }
        } catch(Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);  //   1
        }
    }

    public static String getAPToken() throws Exception {
        HttpURLConnection connection = null;
        String apiUrl = AUD_SERVER_URL + "/api/auth/sign/ap/token";
        String apAccessToken = "";
        try{
            if ((AUD_AP_ID == null || AUD_AP_ID.isEmpty()) || (AUD_SECRET_KEY == null || AUD_SECRET_KEY.isEmpty())){
                System.err.println("AUD  Application   id, pw   .");
                System.exit(1);
            }

            // aud7   secret key ssh private key  .
            PrivateKey privateKey = loadPrivateKey(AUD_SECRET_KEY_FILE_PATH);
            // aud7 secret key
            String signedMessage = signMessage(AUD_SECRET_KEY, privateKey);

            // Header
            Map<String, String> requestHeaders = new HashMap<>();
            requestHeaders.put("X-AUD-AP-Id", AUD_AP_ID);
            requestHeaders.put("X-AUD-AP-Secret-SSH", signedMessage);
            requestHeaders.put("X-AP-UPDATE-ADDR" , "127.0.0.1");
            requestHeaders.put("X-AUD-USER", AUD_AP_ID);

            try{
                // URL
                URL url = new URL(apiUrl);
                boolean isSecure = AUD_SERVER_URL.toLowerCase().startsWith("https:");
                if(isSecure){
                    // https
                    connection = (HttpsURLConnection)url.openConnection();
                    SSLContext sslContext = getSslContext();
                    ((HttpsURLConnection) connection).setSSLSocketFactory(sslContext.getSocketFactory());
                    ((HttpsURLConnection)connection).setHostnameVerifier((hostname, session) -> true);  //
                }else{
                    connection = (HttpURLConnection)url.openConnection();
                }
            }catch(MalformedURLException e){
                throw new Exception("AUD   .");
            }catch(IOException e){
                System.out.println("==========IOException");
                throw new Exception("  [api url:"+ apiUrl +"]");
            }

            // HTTP
            connection.setRequestMethod("POST");
            for(Map.Entry<String, String> header : requestHeaders.entrySet()) {
                connection.setRequestProperty(header.getKey(), header.getValue());
            }
            //
            int responseCode = connection.getResponseCode();
            System.out.println("Response Code: " + responseCode);

            if (responseCode == HttpURLConnection.HTTP_OK) {
                apAccessToken = connection.getHeaderField("bimatrix_ap_accessToken");
                if (apAccessToken == null) {
                    throw new Exception("Ap Token   ");
                } else {
                    return apAccessToken;
                }
            } else if (responseCode == HttpURLConnection.HTTP_UNAUTHORIZED) {
                throw new Exception(" Ap token .");
            }
        }catch(Exception e) {
            e.printStackTrace();
            throw e;
        }finally {
            if (connection != null) {
                connection.disconnect();
            }
        }
        return null;
    }

    //
    public static PrivateKey loadPrivateKey(String path) throws Exception {
        //      (    )
        System.out.println("Private Key path: " + path);
        String keyPEM = new String(Files.readAllBytes(Paths.get(path)))
                .replaceAll("-----BEGIN PRIVATE KEY-----", "")
                .replaceAll("-----END PRIVATE KEY-----", "")
```

```
            .replaceAll("\\s", "");  //

    byte[] keyBytes = Base64.getDecoder().decode(keyPEM);
    PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(keyBytes);
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    return keyFactory.generatePrivate(keySpec);
}

//
public static String signMessage(String message, PrivateKey privateKey) throws Exception {
    Signature signature = Signature.getInstance("SHA256withRSA");
    signature.initSign(privateKey);
    signature.update(message.getBytes());

    byte[] signedBytes = signature.sign();
    return Base64.getEncoder().encodeToString(signedBytes);
}

public static String execute(String url, String params, String method, String apAccessToken) throws Exception {
    HttpURLConnection connection = null;

    OutputStream outStream = null;
    BufferedReader bufferReader = null;

    try{
        try{
            // URL
            URL url1 = new URL(AUD_SERVER_URL+url);
            boolean isSecure = AUD_SERVER_URL.toLowerCase().startsWith("https:");
            if(isSecure){
                // https
                connection = (HttpsURLConnection)url1.openConnection();
                SSLContext sslContext = getSslContext();
                ((HttpsURLConnection) connection).setSSLSocketFactory(sslContext.getSocketFactory());
                ((HttpsURLConnection) connection).setHostnameVerifier((hostname, session) -> true);
            }else{
                connection = (HttpURLConnection)url1.openConnection();
            }
        }catch(MalformedURLException e){
            throw new Exception("AUD   .");
        }catch(IOException e){
            throw new Exception(" ");
        }

        // HTTP
        connection.setRequestMethod(method);
        connection.setRequestProperty("bimatrix_ap_accessToken", apAccessToken);
        connection.setRequestProperty("Accept-Charset", "UTF-8");

        connection.setConnectTimeout(10 * 1000);
        connection.setReadTimeout(300 * 1000);
        connection.setDoInput(true);//

        if (params != null && params.length() > 0){
            connection.setDoOutput(true);
            connection.setRequestProperty("Content-Type", "application/json"); // JSON

            outStream = connection.getOutputStream();
            outStream.write(params.getBytes("UTF-8"));
            outStream.flush();
            outStream.close();
        }

        //
        int responseCode = connection.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK){
            bufferReader = new BufferedReader( new InputStreamReader( connection.getInputStream(), "UTF-8" ) );

            String inputText = "";
            StringBuilder sbText = new StringBuilder();
            while ((inputText = bufferReader.readLine()) != null) {
                if(sbText.length() > 0)
                    sbText.append(inputText + "\r\n");
                else
                    sbText.append(inputText);
            }

            return sbText.toString();
        }

    }catch(Exception e){
        e.printStackTrace();
        throw e;
    }finally {
        if (bufferReader != null)
            bufferReader.close();
        if (outStream != null)
            outStream.close();
        if (connection != null)
            connection.disconnect();
    }
    return null;
}

/* SSL   (https) */
private static SSLContext getSslContext() throws Exception{
    SSLContext sslCTX = null;
    // TrustManager
    TrustManager[] trustManagers = null;
    trustManagers = new TrustManager[]{
            new X509TrustManager() {
                @Override
                public void checkClientTrusted(X509Certificate[] x509Certificates, String s) throws CertificateException {
                }

                @Override
                public void checkServerTrusted(X509Certificate[] x509Certificates, String s) throws CertificateException {
                }

                @Override
                public X509Certificate[] getAcceptedIssuers() {
                    //return new X509Certificate[0];
                    return null;
                }
            }
        };
        // SSL context
        sslCTX = SSLContext.getInstance("TLS");
        sslCTX.init(null, trustManagers, new java.security.SecureRandom());
        return sslCTX;
    }
}
```

b. **sh**

i. : ./restore.sh {} {}
ii. restore.sh JAVA_HOME Java

Sample_restore.sh

```
#!/bin/sh

#
#    JAVA
JAVA_HOME="/home/AUDPlatform/apps/openjdk"

rm -rf CmRestoreApi.class
$JAVA_HOME/bin/javac CmRestoreApi.java -encoding UTF-8
chmod -R 755 *

#
# 1.   (  )
# 2.    (: false)
$JAVA_HOME/bin/java CmRestoreApi $1 $2
```

iii. CmRestoreApi.java AUD_SERVER_URL, AUD_SECRET_KEY, AUD_SECRET_KEY_FILE_PATH, AUD_AP_ID

Sample_CmRestoreApi.java

```
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.Signature;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class CmRestoreApi {
    /**
     * AUD_SERVER_URL: AUD
     * AUD_SECRET_KEY: SSH Key
     * AUD_SECRET_KEY_FILE_PATH: private_key.pem
     * AUD_AP_ID: Ap token    (AUD    )
     */
    private static final String AUD_SERVER_URL = "http://192.168.0.59:8087";
    private static final String AUD_SECRET_KEY = "";
    private static final String AUD_SECRET_KEY_FILE_PATH = "";
    private static final String AUD_AP_ID = "";

    public static void main(String[] args) {
        if (args.length == 0){
            System.err.println("restore file name check please....");
            System.exit(1);  //   1
        }

        try {
            // Ap token
            String apAccessToken = getAPToken();
            //     .
            String restoreFileName = args[0];
            String authRestore = "false";
            if (args.length == 2)
                authRestore = args[1];

            String data = "{\"restoreFileName\" : \""+restoreFileName+"\" , \"authRestore\" : \"" + authRestore+"\"}";
            try {
                String returnData = execute ("/api/cm/restore", data, "POST", apAccessToken);
                System.out.println("======restore data : "+returnData);

                //    api
                String returnLogout = execute ("/api/logout", "", "GET", apAccessToken);
                System.out.println("======returnLogout data : "+returnLogout);
            } catch (Exception ex) {
                ex.printStackTrace();
                throw ex;
            }
        } catch(Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);  //   1
        }
    }

    public static String getAPToken() throws Exception {
        HttpURLConnection connection = null;
        String apiUrl = AUD_SERVER_URL + "/api/auth/sign/ap/token";
        String apAccessToken = "";
        try{
            if ((AUD_AP_ID == null || AUD_AP_ID.isEmpty()) || (AUD_SECRET_KEY == null || AUD_SECRET_KEY.isEmpty())){
                System.err.println("AUD  Application   id, pw   .");
                System.exit(1);
            }

            // aud7   secret key ssh private key  .
            PrivateKey privateKey = loadPrivateKey(AUD_SECRET_KEY_FILE_PATH);
            // aud7 secret key
            String signedMessage = signMessage(AUD_SECRET_KEY, privateKey);

            // Header
            Map<String, String> requestHeaders = new HashMap<>();
            requestHeaders.put("X-AUD-AP-Id", AUD_AP_ID);
            requestHeaders.put("X-AUD-AP-Secret-SSH", signedMessage);
            requestHeaders.put("X-AP-UPDATE-ADDR", apiUrl);
            requestHeaders.put("X-AUD-USER", AUD_AP_ID);

            try{
                // URL
                URL url = new URL(apiUrl);
                boolean isSecure = AUD_SERVER_URL.toLowerCase().startsWith("https:");
                            if(isSecure){
                                // https
                                connection = (HttpsURLConnection)url.openConnection();
                                       SSLContext sslContext = getSslContext();
                                ((HttpsURLConnection) connection).setSSLSocketFactory(sslContext.getSocketFactory());
                                ((HttpsURLConnection)connection).setHostnameVerifier((hostname, session) -> true);
                            }else{
                                connection = (HttpURLConnection)url.openConnection();
                            }
            }catch(MalformedURLException e){
                throw new Exception("AUD   .");
            }catch(IOException e){
                System.out.println("==========IOException");
                throw new Exception("  [api url:"+ apiUrl +"]");
            }

            // HTTP
            connection.setRequestMethod("POST");
            for(Map.Entry<String, String> header : requestHeaders.entrySet()) {
```

```
            connection.setRequestProperty(header.getKey(), header.getValue());
        }
        //
        int responseCode = connection.getResponseCode();
        System.out.println("Response Code: " + responseCode);

        if (responseCode == HttpURLConnection.HTTP_OK) {
            apAccessToken = connection.getHeaderField("bimatrix_ap_accessToken");
            if (apAccessToken == null) {
                throw new Exception("Ap Token   ");
            } else {
                return apAccessToken;
            }
        } else if (responseCode == HttpURLConnection.HTTP_UNAUTHORIZED) {
            throw new Exception(" Ap token .");
        }
    }catch(Exception e) {
        e.printStackTrace();
        throw e;
    }finally {
        if (connection != null) {
            connection.disconnect();
        }
    }
    return null;
}

//
public static PrivateKey loadPrivateKey(String path) throws Exception {
    //      (    )
    System.out.println("Private Key path: " + path);
    String keyPEM = new String(Files.readAllBytes(Paths.get(path)))
            .replaceAll("-----BEGIN PRIVATE KEY-----", "")
            .replaceAll("-----END PRIVATE KEY-----", "")
            .replaceAll("\\s", "");  //

    byte[] keyBytes = Base64.getDecoder().decode(keyPEM);
    PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(keyBytes);
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    return keyFactory.generatePrivate(keySpec);
}

//
public static String signMessage(String message, PrivateKey privateKey) throws Exception {
    Signature signature = Signature.getInstance("SHA256withRSA");
    signature.initSign(privateKey);
    signature.update(message.getBytes());

    byte[] signedBytes = signature.sign();
    return Base64.getEncoder().encodeToString(signedBytes);
}

public static String execute(String url, String params, String method, String apAccessToken) throws Exception {
    HttpURLConnection connection = null;

    OutputStream outStream = null;
    BufferedReader bufferReader = null;

    try{
        try{
            // URL
            URL url1 = new URL(AUD_SERVER_URL+url);
                        boolean isSecure = AUD_SERVER_URL.toLowerCase().startsWith("https:");
                        if(isSecure){
                            // https
                            connection = (HttpsURLConnection)url1.openConnection();
                            SSLContext sslContext = getSslContext();
                            ((HttpsURLConnection) connection).setSSLSocketFactory(sslContext.getSocketFactory());
                            ((HttpsURLConnection)connection).setHostnameVerifier((hostname, session) -> true);  //
                        }else{
                            connection = (HttpURLConnection)url1.openConnection();
                        }
    }catch(MalformedURLException e){
        throw new Exception("AUD   .");
    }catch(IOException e){
        throw new Exception(" ");
    }

    // HTTP
    connection.setRequestMethod(method);
    connection.setRequestProperty("bimatrix_ap_accessToken", apAccessToken);
    connection.setRequestProperty("Accept-Charset", "UTF-8");

    connection.setConnectTimeout(10 * 1000);
    connection.setReadTimeout(300 * 1000);
    connection.setDoInput(true);//

    if (params != null && params.length() > 0){
        connection.setDoOutput(true);
        connection.setRequestProperty("Content-Type", "application/json"); // JSON

        outStream = connection.getOutputStream();
        outStream.write(params.getBytes("UTF-8"));
        outStream.flush();
        outStream.close();
    }

    //
    int responseCode = connection.getResponseCode();
    if (responseCode == HttpURLConnection.HTTP_OK){
        bufferReader = new BufferedReader( new InputStreamReader( connection.getInputStream(), "UTF-8" ) );

        String inputText = "";
        StringBuilder sbText = new StringBuilder();
        while ((inputText = bufferReader.readLine()) != null) {
            if(sbText.length() > 0)
                sbText.append(inputText + "\r\n");
            else
                sbText.append(inputText);
            }

            return sbText.toString();
        }
    }catch(Exception e){
        e.printStackTrace();
        throw e;
    }finally {
        if (bufferReader != null)
            bufferReader.close();
        if (outStream != null)
            outStream.close();
        if (connection != null)
            connection.disconnect();
    }
    return null ;
}


/* SSL   (https) */
    private static SSLContext getSslContext() throws Exception{
        SSLContext sslCTX = null;
        // TrustManager
        TrustManager[] trustManagers = null;
        trustManagers = new TrustManager[]{
        new X509TrustManager() {
            @Override
            public void checkClientTrusted(X509Certificate[] x509Certificates, String s) throws CertificateException {
            }
```

```
                @Override
                public void checkServerTrusted(X509Certificate[] x509Certificates, String s) throws CertificateException {
                }

                @Override
                public X509Certificate[] getAcceptedIssuers() {
                    //return new X509Certificate[0];
                    return null;
                }
            }
            };
            // SSL context
            sslCTX = SSLContext.getInstance("TLS");
            sslCTX.init(null, trustManagers, new java.security.SecureRandom());
            return sslCTX;
        }
}
```

(Release No. 7.0.500.20251002-11)

a. **sh**

i. : ./backup.sh {} {}
ii. backup.sh JAVA_HOME Java

Sample_ backup.sh

```
#!/bin/sh

#
#    JAVA
JAVA_HOME="/home/aud7/istream311/AUDPlatform/apps/openjdk"

rm -rf CmBackupApi.class
$JAVA_HOME/bin/javac CmBackupApi.java -encoding UTF-8
chmod -R 755 *

#
# 1.    (  .zip  )
# 2.   json
$JAVA_HOME/bin/java CmBackupApi $1 $2
```

iii. CmBackupApi.java AUD_SERVER_URL, audApId, audApPw

Sample_CmBackupApi.java

```
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.HashMap;
import java.util.Map;

public class CmBackupApi {
    /**
     * AUD
     */
    private static String AUD_SERVER_URL = "http://192.168.0.59:8087";  // AUD
    private static String apAccessToken = null;

    public static void main(String[] args) {
        if (args.length != 2) {
            System.err.println("backup parameter check please....");
            System.exit(1);  //   1
        }

        /**
         * audApId , audApPw  AUD   Application
         *       Application id / pw
         */
        String audApId = "ap.test"; // AUD   Application  id
        String audApPw = "ap.test"; // AUD   Application  pw

        HttpURLConnection connection = null;

        try{
            // header   .
            Map<String, String> requestHeaders = new HashMap<>();
            requestHeaders.put("X-AUD-AP-Id", audApId);
            requestHeaders.put("X-AUD-AP-Secret", audApPw);
            requestHeaders.put("X-AUD-USER", audApId);
            requestHeaders.put("X-AP-UPDATE-ADDR", "127.0.0.1");

            try{
                // URL
                URL url = new URL(AUD_SERVER_URL+"/api/auth/sign/ap/token");
                boolean isSecure = AUD_SERVER_URL.toLowerCase().startsWith("https:");
                            if(isSecure){
                                // https
                                connection = (HttpsURLConnection)url.openConnection();
                                SSLContext sslContext = getSslContext();
                                ((HttpsURLConnection) connection).setSSLSocketFactory(sslContext.getSocketFactory());
                                ((HttpsURLConnection)connection).setHostnameVerifier((hostname, session) -> true);
                            }else{
                                connection = (HttpURLConnection)url.openConnection();
                            }
            }catch(MalformedURLException e){
                throw new Exception("AUD   .");
            }catch(IOException e){
                System.out.println("==========IOException");
                throw new Exception("  [api url:"+ AUD_SERVER_URL +"]");
            }

            // HTTP
            connection.setRequestMethod("POST");
            for(Map.Entry<String, String> header :requestHeaders.entrySet()) {
                connection.setRequestProperty(header.getKey(), header.getValue());
            }
            //
            int responseCode = connection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK){
                apAccessToken = connection.getHeaderField("bimatrix_ap_accessToken");
                if (apAccessToken == null){
                    throw new Exception("ap token   ");
                }
            }

        }catch(Exception e){
            System.err.println(e.getMessage());
            System.exit(1);  //   1
        }finally {
            if (connection != null) {
                connection.disconnect();
            }
```

```
    }

    //     .
    String backupReportCode = args[0];
    String tableInfoFileName = args[1];
    String data = "{\"reportCode\" : \""+backupReportCode+"\" , \"tableInfoFileName\" : \"" + tableInfoFileName+"\"}";
    try {
        String returnData = execute ("/api/cm/backup" , data , "POST");
        System.out.println("======backup data : "+returnData);

        //    api
        String returnLogout = execute ("/api/logout" , "" , "GET");
        System.out.println("======returnLogout data : "+returnLogout);
    } catch (Exception e) {
        e.printStackTrace();
    }

}

public static String execute(String url, String params , String method) throws Exception{
    HttpURLConnection connection = null;

    OutputStream outStream = null;
    BufferedReader bufferReader = null;

    try{
        try{
            // URL
            URL url1 = new URL(AUD_SERVER_URL+url);
                        boolean isSecure = AUD_SERVER_URL.toLowerCase().startsWith("https:");
                        if(isSecure){
                    // https
                            connection = (HttpsURLConnection)url1.openConnection();
                        SSLContext sslContext = getSslContext();
                        ((HttpsURLConnection) connection).setSSLSocketFactory(sslContext.getSocketFactory());
                        ((HttpsURLConnection)connection).setHostnameVerifier((hostname, session) -> true);
                    }else{
                        connection = (HttpURLConnection)url1.openConnection();
                    }
}catch(MalformedURLException e){
    throw new Exception("AUD   .");
}catch(IOException e){
    throw new Exception(" ");
}

// HTTP
connection.setRequestMethod(method);
connection.setRequestProperty("bimatrix_ap_accessToken", apAccessToken);
connection.setRequestProperty("Accept-Charset", "UTF-8");

connection.setConnectTimeout(10 * 1000);
connection.setReadTimeout(300 * 1000);
connection.setDoInput(true);//

if (params != null && params.length() > 0){
    connection.setDoOutput(true);
    connection.setRequestProperty("Content-Type", "application/json"); // JSON

    outStream = connection.getOutputStream();
    outStream.write(params.getBytes("UTF-8"));
    outStream.flush();
    outStream.close();
}

//
int responseCode = connection.getResponseCode();
if (responseCode == HttpURLConnection.HTTP_OK){
    bufferReader = new BufferedReader( new InputStreamReader( connection.getInputStream(), "UTF-8" ) );

    String inputText = "";
            StringBuilder sbText = new StringBuilder();
            while ((inputText = bufferReader.readLine()) != null) {
                if(sbText.length() > 0)
                    sbText.append(inputText + "\r\n");
                else
                    sbText.append(inputText);
            }

            return sbText.toString();
        }

    }catch(Exception e){
        e.printStackTrace();
        System.exit(1);  //   1
    }finally {
        if (bufferReader != null)
            bufferReader.close();
        if (outStream != null)
            outStream.close();
        if (connection != null)
            connection.disconnect();
    }

    return null ;
}
    /* SSL   (https) */
    private static SSLContext getSslContext() throws Exception{
        SSLContext sslCTX = null;
        // TrustManager
        TrustManager[] trustManagers = null;
        trustManagers = new TrustManager[]{
        new X509TrustManager() {
            @Override
            public void checkClientTrusted(X509Certificate[] x509Certificates, String s) throws CertificateException {
            }

            @Override
                public void checkServerTrusted(X509Certificate[] x509Certificates, String s) throws CertificateException {
                }

                @Override
                public X509Certificate[] getAcceptedIssuers() {
                    //return new X509Certificate[0];
                    return null;
                }
            }
            };
            // SSL context
            sslCTX = SSLContext.getInstance("TLS");
            sslCTX.init(null, trustManagers, new java.security.SecureRandom());
            return sslCTX;
        }
}
```

b. **sh**

i. : ./restore.sh {} {}
ii. restore.sh JAVA_HOME Java

Sample_restore.sh

```
#!/bin/sh

#
#    JAVA
JAVA_HOME="/home/aud7/istream311/AUDPlatform/apps/openjdk"

rm -rf CmRestoreApi.class
$JAVA_HOME/bin/javac CmRestoreApi.java -encoding UTF-8
chmod -R 755 *

#
# 1.   (  )
# 2.    (: false)
$JAVA_HOME/bin/java CmRestoreApi $1 $2
```

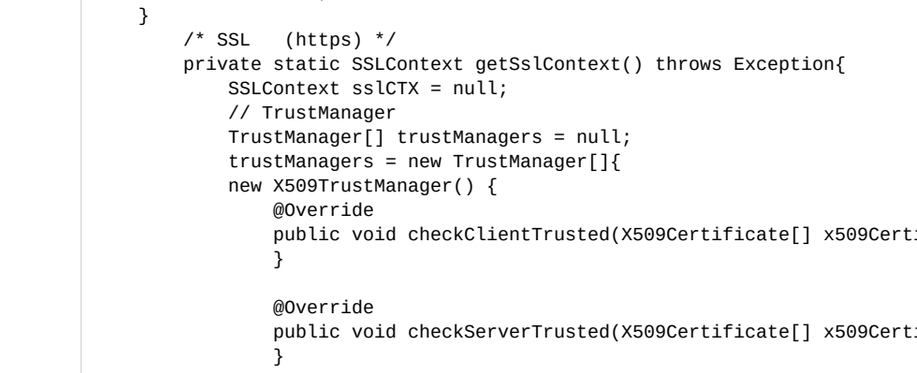
iii. CmRestoreApi.java AUD_SERVER_URL, audApId, audApPw

Sample_CmRestoreApi.java

```
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.HashMap;
import java.util.Map;

public class CmRestoreApi {
    /**
     * AUD
     */
    private static String AUD_SERVER_URL = "http://192.168.0.59:8087";
    private static String apAccessToken = null;

    public static void main(String[] args) {
        if (args.length == 0){
            System.err.println("restore file name check please....");
            System.exit(1);  //   1
        }

        /**
         * audApId , audApPw  AUD   Application
         *       Application id / pw
         */
        String audApId = "ap.test";
        String audApPw = "ap.test";

        HttpURLConnection connection = null;

        try{
            // header   .
            Map<String, String> requestHeaders = new HashMap<>();
            requestHeaders.put("X-AUD-AP-Id", audApId);
            requestHeaders.put("X-AUD-AP-Secret", audApPw);
            requestHeaders.put("X-AUD-USER", audApId);
            requestHeaders.put("X-AP-UPDATE-ADDR", "127.0.0.1");

            try{
                // URL
                URL url = new URL(AUD_SERVER_URL+"/api/auth/sign/ap/token");
                boolean isSecure = AUD_SERVER_URL.toLowerCase().startsWith("https:");
                            if(isSecure){
                                // https
                                connection = (HttpsURLConnection)url.openConnection();
                                SSLContext sslContext = getSslContext();
                                ((HttpsURLConnection) connection).setSSLSocketFactory(sslContext.getSocketFactory());
                                ((HttpsURLConnection)connection).setHostnameVerifier((hostname, session) -> true);
                            }else{
                                connection = (HttpURLConnection)url.openConnection();
                            }
            }catch(MalformedURLException e){
                throw new Exception("AUD   .");
            }catch(IOException e){
                System.out.println("==========IOException");
                throw new Exception("  [api url:"+ AUD_SERVER_URL +"]");
            }

            // HTTP
            connection.setRequestMethod("POST");
            for(Map.Entry<String, String> header :requestHeaders.entrySet()) {
                connection.setRequestProperty(header.getKey(), header.getValue());
            }
            //
            int responseCode = connection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK){
                apAccessToken = connection.getHeaderField("bimatrix_ap_accessToken");
                if (apAccessToken == null){
                    throw new Exception("ap token   ");
                }
            }

        }catch(Exception e){
            System.err.println(e.getMessage());
            System.exit(1);  //   1
        }finally {
            if (connection != null) {
                connection.disconnect();
            }
        }

        //     .
        String restoreFileName = args[0];
        String authRestore = "false";
        if (args.length == 2)
            authRestore = args[1];

        String data = "{\"restoreFileName\" : \""+restoreFileName+"\" , \"authRestore\" : \"" + authRestore+"\"}";
        try {
            String returnData = execute ("/api/cm/restore" , data , "POST");
            System.out.println("======restore data : "+returnData);

            //    api
            String returnLogout = execute ("/api/logout" , "" , "GET");
            System.out.println("======returnLogout data : "+returnLogout);
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    public static String execute(String url, String params , String method) throws Exception{
        HttpURLConnection connection = null;

        OutputStream outStream = null;
        BufferedReader bufferReader = null;
```

```
try{
    try{
        // URL
        URL url1 = new URL(AUD_SERVER_URL+url);
                        boolean isSecure = AUD_SERVER_URL.toLowerCase().startsWith("https:");
                        if(isSecure){
                            // https
                            connection = (HttpsURLConnection)url1.openConnection();
                            SSLContext sslContext = getSslContext();
                            ((HttpsURLConnection) connection).setSSLSocketFactory(sslContext.getSocketFactory());
                            ((HttpsURLConnection)connection).setHostnameVerifier((hostname, session) -> true);
                        }else{
                            connection = (HttpURLConnection)url1.openConnection();
                        }
    }catch(MalformedURLException e){
        throw new Exception("AUD   .");
    }catch(IOException e){
        throw new Exception(" ");
    }

    // HTTP
    connection.setRequestMethod(method);
    connection.setRequestProperty("bimatrix_ap_accessToken", apAccessToken);
    connection.setRequestProperty("Accept-Charset", "UTF-8");

    connection.setConnectTimeout(10 * 1000);
    connection.setReadTimeout(300 * 1000);
    connection.setDoInput(true);//

    if (params != null && params.length() > 0){
        connection.setDoOutput(true);
        connection.setRequestProperty("Content-Type", "application/json"); // JSON

        outStream = connection.getOutputStream();
        outStream.write(params.getBytes("UTF-8"));
        outStream.flush();
        outStream.close();
    }

    //
    int responseCode = connection.getResponseCode();
    if (responseCode == HttpURLConnection.HTTP_OK){
        bufferReader = new BufferedReader( new InputStreamReader( connection.getInputStream(), "UTF-8" ) );

        String inputText = "";
        StringBuilder sbText = new StringBuilder();
        while ((inputText = bufferReader.readLine()) != null) {
            if(sbText.length() > 0)
                sbText.append(inputText + "\r\n");
            else
                sbText.append(inputText);
        }

        return sbText.toString();
    }

}catch(Exception e){
    e.printStackTrace();
    System.exit(1);  //   1
}finally {
    if (bufferReader != null)
        bufferReader.close();
    if (outStream != null)
        outStream.close();
    if (connection != null)
        connection.disconnect();
}

return null ;
    }
        /* SSL   (https) */
        private static SSLContext getSslContext() throws Exception{
            SSLContext sslCTX = null;
            // TrustManager
            TrustManager[] trustManagers = null;
            trustManagers = new TrustManager[]{
            new X509TrustManager() {
                @Override
                public void checkClientTrusted(X509Certificate[] x509Certificates, String s) throws CertificateException {
                }

                @Override
                public void checkServerTrusted(X509Certificate[] x509Certificates, String s) throws CertificateException {
                }

                @Override
                public X509Certificate[] getAcceptedIssuers() {
                    //return new X509Certificate[0];
                    return null;
                }
            }
            };
            // SSL context
            sslCTX = SSLContext.getInstance("TLS");
            sslCTX.init(null, trustManagers, new java.security.SecureRandom());
            return sslCTX;
        }
}
```

6.
    a.



7. **() sh API timeout**
    a. CmBackupApi.java, CmRestoreApi.java timeout
        i. Connect timeout( ): 10
        ii. Read timeout( ): 5
        iii. SocketTimeoutException , timeout .

Sample_Timeout

```
// CmBackupApi.java ->
// CmRestoreApi.java ->

connection.setConnectTimeout(10 * 1000);        // Connect timeout(  ): 10
connection.setReadTimeout(300 * 1000);               // Read timeout(  ): 5
```

	☑ API Open	☐ UI Open	☐ Read Only	☐ Not Use	☐ Hidden	☐ Not Recommend	☐
	• Release No : 7.0.500.20250213-11 • matrix-service : 7.3.500.20250203 / matrix-cm : 7.0.500.10						

- **(Release No. 7.0.500.20251002-11)**

_API sh .zip_20251002-11

_ _SAMPLE.json_20251002-11

- ( )

_API sh .zip

_ _SAMPLE.json