SOPHOS
Security made simple.

Sophos Mobile Control
Network Access Control interface guide

Product version: 6

Document date: December 2015

Contents

1 About Sophos Mobile Control

Sophos Mobile Control is a device management solution for mobile devices like smartphones and tablets. It allows configuration and software distribution as well as security settings and many other device management operations on mobile devices.

With Sophos Mobile Control you can keep corporate data safe by managing apps and security settings. The Sophos Mobile Control client is easily installed and managed with over-the air setup and configuration through the Sophos Mobile Control web console. With the Sophos Mobile Control Self Service Portal for your users, you can reduce IT efforts by allowing them to register their own devices and carry out other tasks without having to contact the helpdesk.

2 About Network Access Control integration

Sophos Mobile Control offers a HTTP-based interface to integrate third-party Network Access Control (NAC) products. The implementation has to be done by the third-party vendor. The interface delivers a list of MAC addresses including the Sophos Mobile Control compliance status. NAC products may use the interface to permit or deny access to network segments depending on the compliance status of a device.

3 Prerequisites

To use the NAC interface, you need a user account for logging on to the Sophos Mobile Control web console. An account consists of the following information:

- Customer
- User
- Password

Note: Sophos Mobile Control is offered in two delivery models: Sophos Mobile Control for on-premise - installation and Sophos Mobile Control as a Service. Depending on the variant you use, NAC is configured by a super administrator (on-premise) or an administrator. For further information on the two variants, refer to the *Sophos Mobile Control administrator help*. For further information on how to configure NAC as a super administrator, refer to the *Sophos Mobile Control super administrator guide*.

The NAC component needs to have a certificate (including private key). This certificate must be used to sign requests sent to the interface. The certificate's subject common name (CN) must start with the prefix "SMCNAC" and can be self-signed.

The public certificate (usually with ".cer" or ".crt" file extension) needs to be uploaded to Sophos Mobile Control:

1. Log in at the Sophos Mobile Control web console
2. In the web console menu bar go to **Settings**, click **System setup** and go to the **Network Access Control** tab.
3. Upload the public certificate and save it.

The interface delivers the MAC addresses of devices of the Sophos Mobile Control customer the certificate has been uploaded to. If you have uploaded the certificate in the super administrator customer, the devices of all customers are used.

After NAC has been configured, log out from the web console and log in again. When you configure compliance rules in the web console, the checkbox **Disallow Network Access** is available. If you select this checkbox, Network Access is denied as soon as compliance criteria have been violated.

4 Protocol

The interface is HTTP-based and uses SSL encryption. It is available at

https://[SMC hostname]/servlets/nac/

You can use a HTTP GET request to verify that the servlet is running.

4.1 List request

The protocol offers just one operation called “list”. This operation lists all known MAC addresses and the compliance statuses of the corresponding devices.

The list command must be sent via HTTP POST. The string “list” must be set in the content. The HTTP request must then be signed using the private key of the certificate whose public part has been uploaded to SMC. The generated signature must be base64-encoded and sent as HTTP header “mdm-signature”. See the appendix for an example of how to sign a request using Java.

4.1.1 Example request

HTTP method: POST

HTTP header: mdm-protocol-version = 1

HTTP header: mdm-signature =
MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAaCAJIAEBGxp
c3QAAAAAAACggDCCAucwggKnAgRRWtDCMAkGByqGSM44BAMwWzELMAkGA1UE[…]

Data:

list

4.2 List response

Sophos Mobile Control Server response lists the MAC addresses of compliant and non-compliant devices, separated by section markers. It also lists configuration values. The only configuration returned is the minimum query interval in seconds. If the NAC component queries too often, a “503 Service unavailable” response is sent. It also includes a “Retry-After header” indicating the time period in seconds before retrying.

4.2.1 Example response

HTTP status code: 200

Data:

---config---

refreshListInterval=120

---compliant---

00:12:34:56:78:9a

00:76:54:32:10:bc

---non-compliant---

00:ab:cd:f0:12:34

4.2.2 Example response if queried too often

HTTP status code: 503

HTTP header: Retry-After = 67

4.2.3 Example response if signature is wrong or unknown

HTTP status code: 403

5 Appendix

5.1 Example using Java

The following snippet shows a simplified Java example on how to sign a request, submit it and how to read the response. Exception and error handling is not included for brevity.

The following example makes use of the BouncyCastle and Apache commons-codec libraries.

```
public void example()
{
   Security.addProvider(new BouncyCastleProvider());
   OutputStream out = null;
   InputStream in = null;
   HttpURLConnection connection = null;
   try
   {
      // prepare the list command
      byte[] plainData = "list".getBytes("UTF-8");
      // sign the data and encode it in base64 format
      String signedB64Data  = signAndEncodeData(plainData);
      // open connection to NAC servlet
      URL url = new URL(("https://SMC-URL/servlets/nac");
      connection = (HttpURLConnection) url.openConnection();
      connection.setRequestMethod("POST");
      connection.setRequestProperty("content-type", "\*/\*");
      connection.setDoInput(true);
      connection.setDoOutput(true);
      connection.setUseCaches(false);
      connection.setRequestProperty("accept", "*/*");
      // add the NAC servlet protocol version header
      connection.addRequestProperty("mdm-protocol-version", "1");
      // add the signed list command base64 encoded
      connection.addRequestProperty("mdm-signature", signedB64Data);
      connection.connect();
      out = connection.getOutputStream();
      // write the list command
      out.write(plainData);
      out.flush();
      out.close();

      // get the response
      int responseCode = connection.getResponseCode();
      if (responseCode == 503)
      {
         // maybe last request is not long enough ago, get next retry in seconds
         String nextRetry = connection.getHeaderField("Retry-After");
         // TODO your error handling here
         return;
      }
      if (responseCode != 200)
      {
         // TODO your error handling here
         return;
      }
```

```
        // get the response content
        in = connection.getInputStream();
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
        List<String> allowed = new ArrayList<String>();
        List<String> denied = new ArrayList<String>();
        List<String> configuration = new ArrayList<String>();

        // read the configuration
        String line = br.readLine();
        if ("---config---".equals(line))
        {
           // read configuration until next block
           while (!"---compliant---".equals((line = br.readLine())))
           {
              if (line.isEmpty())
                 continue;
              configuration.add(line);
           }

           // read mac addresses until next block
           while (!"---non-compliant---".equals((line = br.readLine())))
           {
              if (line.isEmpty())
                 continue;
              allowed.add(line);
           }

           // read mac addresses until end of content
           while ((line = br.readLine()) != null)
           {
              if (line.isEmpty())
                 continue;
              denied.add(line);
           }
        }
        // TODO use the received data
     }
     catch (Exception e)
     {
        // TODO your error handling here
     }
     finally
     {
        // TODO close all streams, cleanup resources
     }
}
```

```
/**
 * Method signs plain data. Signed data will be returned as base64 encoded string.
 * @param plainData byte[]
 * @return String signed data base64 encoded.
 * @throws Exception
 */
private String signAndEncodeData(byte[] plainData) throws Exception
{
   InputStream in = null;
   try
   {
      in = new FileInputStream(pathToKeystore);
      KeyStore keyStore = KeyStore.getInstance("PKCS12");
      keyStore.load(in, keystorePassword);
      PrivateKey key = (PrivateKey)keyStore.getKey(certificateAlias, certificatePwd);
      Certificate[] chain = keyStore.getCertificateChain(certificateAlias);
      CertStore certsAndCRLs = CertStore.getInstance("Collection", new
      CollectionCertStoreParameters(Arrays.asList(chain)), "BC");
      CMSSignedDataGenerator gen = new CMSSignedDataGenerator();
      X509Certificate cert = (X509Certificate) chain[0];
      gen.addSigner(key, cert, CMSSignedGenerator.DIGEST_SHA1);
      gen.addCertificatesAndCRLs(certsAndCRLs);
      CMSProcessable data = new CMSProcessableByteArray(plainData);
      CMSSignedData signed = gen.generate(data, true, "BC");
      return Base64.encodeBase64URLSafeString(signed.getEncoded());
   }
   catch (Exception e)
   {
      // TODO your error handling here
   }
   finally
   {
      // TODO close all streams, cleanup resources
   }
}
```

6 Technical support

You can find technical support for Sophos products in any of these ways:

- Visit the SophosTalk forum at http://community.sophos.com/ and search for other users who are experiencing the same problem.
- Visit the Sophos support knowledgebase at http://www.sophos.com/en-us/support.aspx .
- Download the product documentation at http://www.sophos.com/en-us/support/documentation.aspx .
- Open a ticket with our support team at https://secure2.sophos.com/support/contact-support/support-query.aspx.

7 Legal notices