Windows 8.1 Enterprise Device Management Protocol

Updated: August 2015

Applies to: Windows 8.1

Copyright

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. This document is confidential and proprietary to Microsoft. It is disclosed and can be used only pursuant to a non-disclosure agreement.

© 2015 Microsoft. All rights reserved.

Contents

© 2014 Microsoft. All rights reserved.

© 2014 Microsoft. All rights reserved.

Summary

Windows 8.1 provides IT managers with an enterprise management solution to help manage company security policies and business applications, while helping ensure users have enhanced privacy on their personal devices.

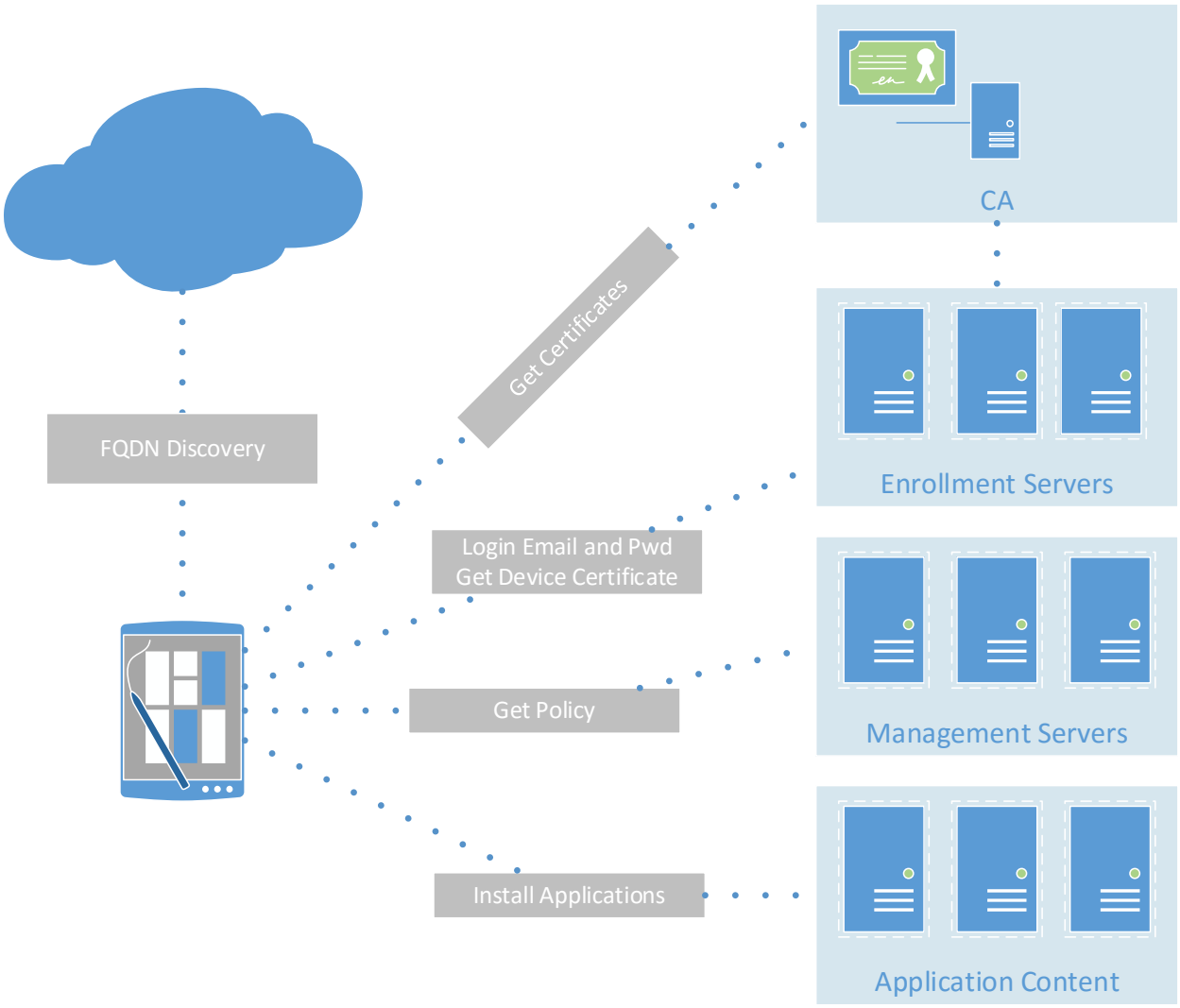
A built-in management component communicates with an enterprise management service. There are two parts to the Windows management component:

- The enrollment agent, which enrolls and configures the device so that it can communicate with an enterprise management service.
- The management agent, which periodically synchronizes with the management service to check for updates and apply the latest policies set by IT.

Third-party mobile device management (MDM) solutions can manage Windows by using the Enterprise Device Management protocol. The built-in management agent can communicate with a third-party service proxy that supports the protocols outlined in this document to perform enterprise management tasks. The third-party service will have the same consistent first-party user experience for enrollment, which also provides simplicity for Windows users. Third-party MDM solutions do not have to create or download a client to manage a Windows-based device.

© 2014 Microsoft. All rights reserved.

The following diagram shows the overall Enterprise Device Management architecture.



Enterprise Device Management architecture

© 2014 Microsoft. All rights reserved.

Connecting to the management infrastructure (enrollment)

The first thing you must do to enable enterprise management is to configure the device to communicate with the MDM server by using security precautions. You do this by using the enrollment process described in this section. The enrollment service verifies that only authenticated and authorized devices can be configured to be managed by their enterprise.

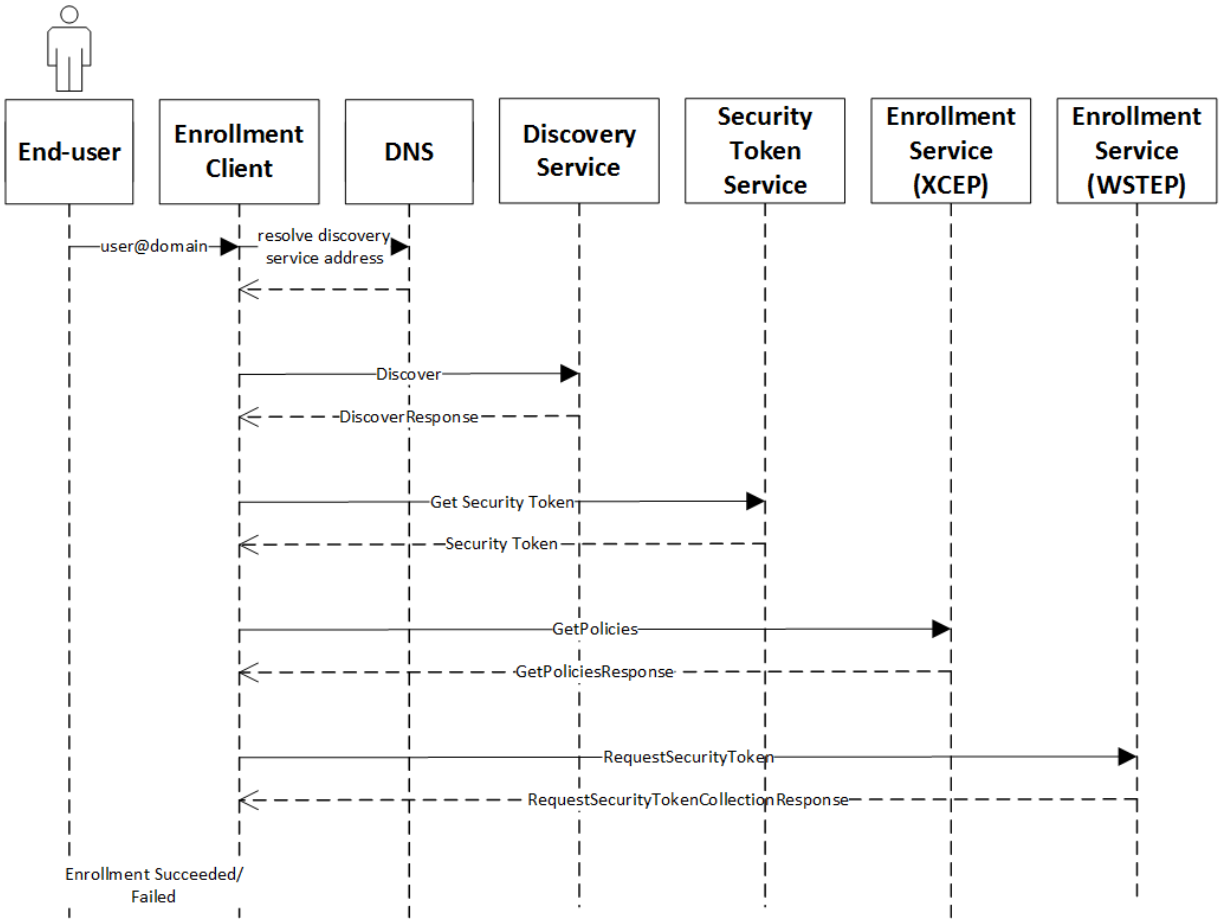
At a high level, the enrollment process consists of three steps:

1. **Discovery of the enrollment endpoint**
   This step provides the enrollment endpoint configuration settings.
2. **Certificate installation**
   This step handles user authentication, certificate generation, and certificate installation. The installed certificates will be used in the future to manage client/server Secure Sockets Layer (SSL) mutual authentication.
3. **DM client provisioning**
   This step configures the Device Management (DM) client to connect to an MDM server after enrollment by using DM SyncML over HTTPS (also known as Open Mobile Alliance Device Management [OMA DM] XML).

© 2014 Microsoft. All rights reserved.

Conceptual flow

The following diagram shows the enrollment flow. The following examples refer to the fictional company Contoso, whose website is contoso.com.

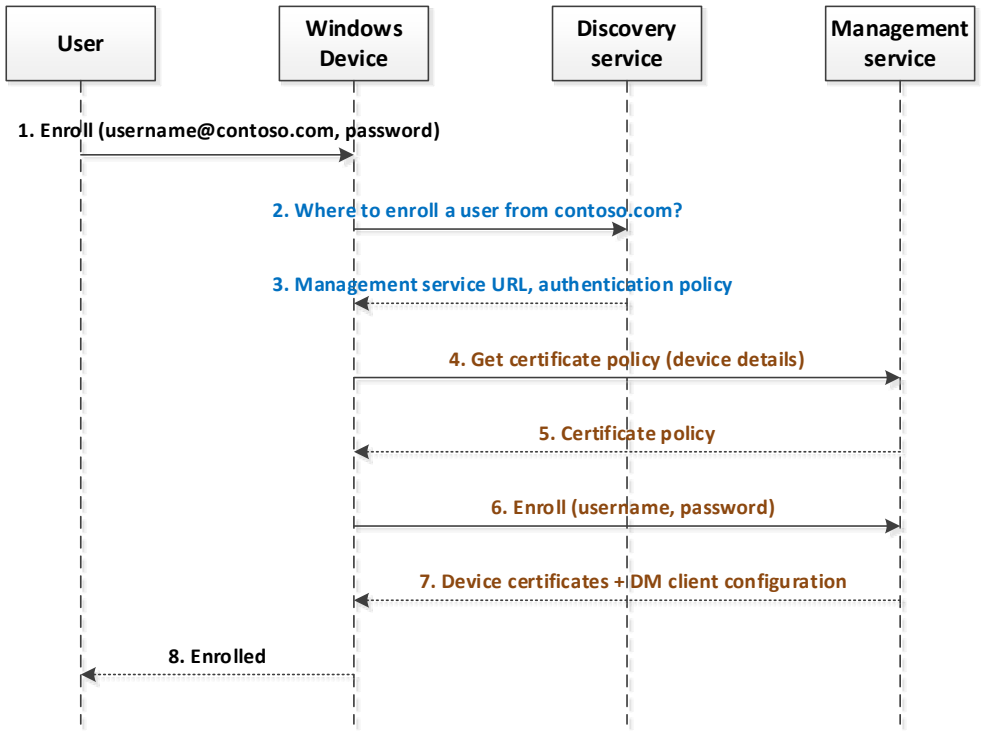


The enrollment flow shown here is discussed in the following steps:

1. The user's email name is entered by using Workplace in modern settings and sent to the enrollment client.
2. The enrollment client extracts the domain suffix from the email address, prepends the domain name with a well- known label (enterpriseenrollment), and resolves the address to the Discovery Service (DS). The administrator configures the network name resolution service (that is, the Domain Name System [DNS]) appropriately.
3. The enrollment client sends an HTTP GET request to the DS to validate that the service endpoint exists.

© 2014 Microsoft. All rights reserved.

4. The enrollment client sends a **Discover** message to the DS. The DS responds with a **DiscoverResponse** message that contains the Uniform Resource Locators (URLs) of service endpoints required for the following steps.
5. The enrollment client communicates with the Security Token Service (STS) to obtain a security token to authenticate with the Enrollment Service (ES).
6. The enrollment client sends a **GetPolicies** message to the enrollment service ES endpoint [MS-XCEP] using the security token received in the previous step. The ES endpoint [MS-XCEP] responds with a **GetPoliciesResponse** message that contains the certificate policies required for the next step. For more information about these messages, see [MS-XCEP] (http://msdn.microsoft.com/en-us/library/dd302869.aspx).
7. The enrollment client sends a **RequestSecurityToken** message to the ES endpoint [MS-WSTEP] using the security token received in step 4. The ES endpoint [MS-WSTEP] responds with **RequestSecurityTokenResponseCollection**, which contains the identity and provisioning information for the device management client [MS-MDM]. For more information about these messages, see [MS-WSTEP] (http://msdn.microsoft.com/en-us/library/dd340609.aspx).



Enrollment flow

© 2014 Microsoft. All rights reserved.

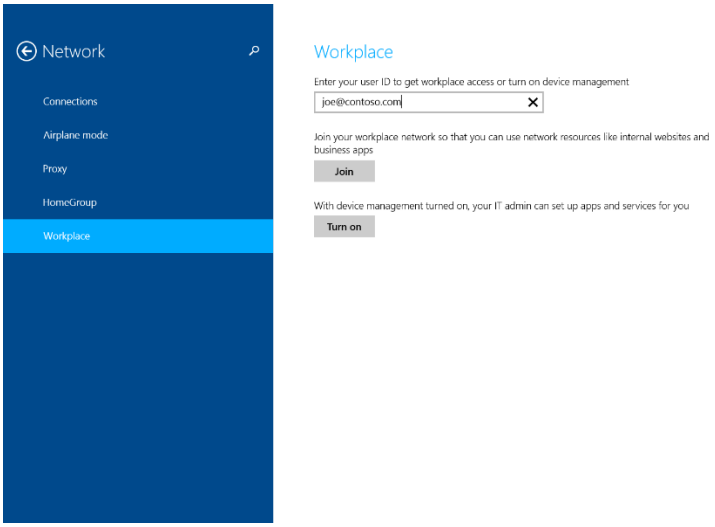
The following table describes the enrollment flow.

Step	Description
1	The user enters their email address, taps the **turn on** management button, and is requested to enter their credentials.
2	The device sends a discover request to the discovery service. The request includes the domain part of the email address.
3	The discovery service returns a response that contains the management service URL and optionally an authentication policy. The authentication policy includes information on the authentication method and required authentication steps.
4, 5	The device contacts the management service and asks for a certificate policy. The management service returns a certificate policy. The returned certificate issuers provide an X.509 v3 security token by using MS-WSTEP.
6, 7	Based on the authentication policy data (step 3) and the certificate policy, the device creates an enrollment request. The management service generates a device certificate and provides DM client settings. The device installs the certificate chain and initiates a DM request to the MDM server.
8	The user receives visual confirmation the device is enrolled.

© 2014 Microsoft. All rights reserved.

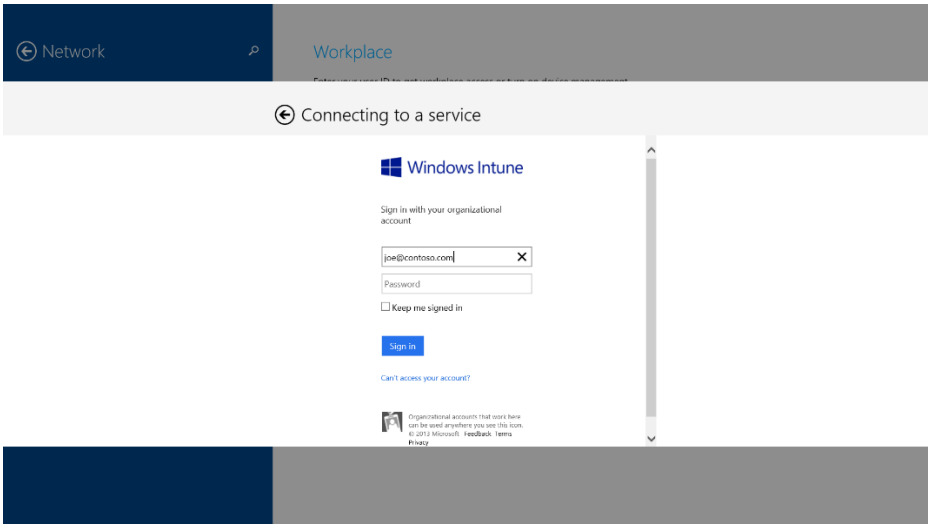
Enrollment UI

The following screen shots show the user experience for enrollment. Notice the user experience is the enrollment client's built-in UI. There is no third-party extensibility. From the modern settings application, the user will select **Network** then select **Workplace**. From the Workplace item, the user enters a corporate email address and then taps the **Turn on** button. The device will then attempt to auto-discover an enrollment endpoint and start the enrollment process.



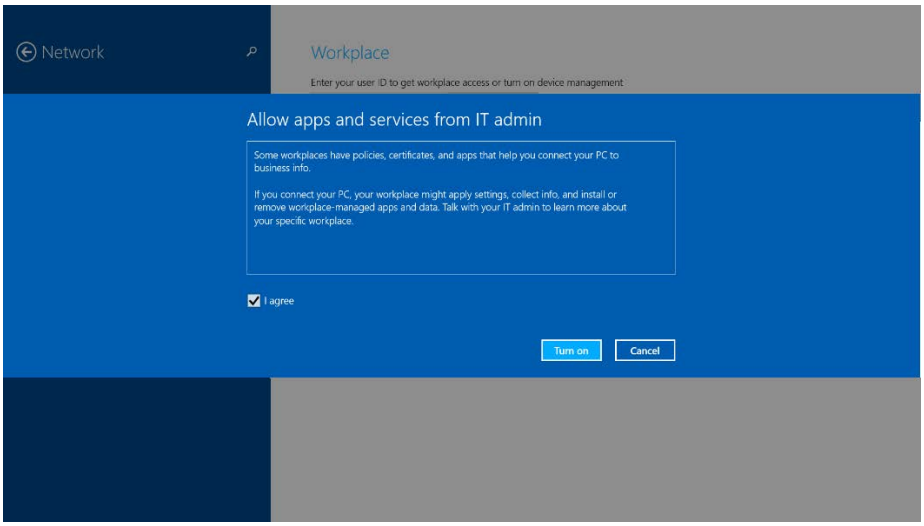
If the enrollment service is auto-discovered, the enrollment service URL and browser control used to log on are returned to the enrollment client. The enrollment client calls the browser control to authenticate the user.

In this example the Windows Intune portal for logging on the user is presented to the end-user.



© 2014 Microsoft. All rights reserved.

When the end-user has successful authenticated, the built-in enrollment UI will present a consent screen which the user must agree to in order to complete the enrollment process.



Supported protocols summary

The following subsections describe the protocols that are used in the enrollment flow.

Discovery request (steps 3–4)

The discovery request is a simple HTTP POST call that returns XML over HTTP. The returned XML includes the authentication URL, the management service URL, and the user credential type. The enrollment agent may also send a simple HTTP GET request to verify that the endpoint exists.

Certificate enrollment policy (step 6)

The certificate enrollment policy configuration is an implementation of the MS-XCEP protocol, which is described in [MS-XCEP]: X.509 Certificate Enrollment Policy Protocol Specification (http://msdn.microsoft.com/en-us/library/dd302869.aspx). Section 4 of the specification provides an example of the policy request and response. The X.509 Certificate Enrollment Policy Protocol is a minimal messaging protocol that includes a single client request message (GetPolicies) with a matching server response message (GetPoliciesResponse).

Certificate enrollment (step 7)

The certificate enrollment is an implementation of the MS-WSTEP protocol.

© 2014 Microsoft. All rights reserved.

Management configuration (step 8)

The server sends provisioning XML that contains a server certificate (for SSL server authentication), a device certificate issued by enterprise CA and OMA-DM agent bootstrap information (for the client to communicate with the management server).

Discovery web service

Description

The discovery web service provides the configuration information that is required for a user to enroll a device with a management service. The service is a RESTful web service over HTTPS (server authentication only).

Prerequisite

The administrator of the discovery service must create a host with the address enterpriseenrollment.domain.com.

Request

The device's automatic discovery workflow uses the domain name of the email address that was submitted to the Workplace settings screen during sign in. The automatic discovery system constructs a URI by appending the subdomain **enterpriseenrollment** to the domain of the email address, and by appending the path /EnrollmentServer/Discovery.svc. For example, if the email address is sample@contoso.com, the resulting URI for the first get request would be: http://enterpriseenrollment.contoso.com/EnrollmentServer/Discovery.svc.

The first request is a standard HTTP GET request.

The following example shows the request send to the discovery server that uses HTTP GET, assuming that the user used the email address user@contoso.com during enrollment.

Request:

```
Request Full Url: http://EnterpriseEnrollment.contoso.com/EnrollmentServer/Discovery.svc
Content Type: unknown
Header Byte Count: 153
Body Byte Count: 0
```

Header:

```
GET /EnrollmentServer/Discovery.svc HTTP/1.1
User-Agent: Windows Enrollment Client
Host: EnterpriseEnrollment.contoso.com
Pragma: no-cache
```

Response:

```
Request Full Url: http://EnterpriseEnrollment.contoso.com/EnrollmentServer/Discovery.svc
Content Type: text/html
```

© 2014 Microsoft. All rights reserved.

```
Header Byte Count: 248
Body Byte Count: 0
```

Header:

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Pragma: no-cache
Cache-Control: no-cache
Content-Type: text/html
Content-Length: 0
```

If the enrollment agent receives an HTTP Status 200 from the GET request, the agent will then send a POST request to enterpriseenrollment.*domain.name*/EnrollmentServer/Discovery.svc. After it gets another response from the server (which should tell the device where the enrollment server is), the next message sent from the agent is to enterpriseenrollment.*domain.name* to the enrollment server.

The following logic is applied: The device first tries HTTPS. If the server cert is not trusted by the device, HTTPS fails and the enrollment process ends with an error.

This example shows an HTTP POST request sent to the discovery web service, assuming that user@contoso.com was the email address entered by the user.

https://EnterpriseEnrollment.Contoso.com/EnrollmentServer/Discovery.svc

Header:

```
POST /EnrollmentServer/Discovery.svc HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
User-Agent: Windows Enrollment Client
Host: EnterpriseEnrollment.Contoso.com
Content-Length: xxx
Cache-Control: no-cache
```

```
<?xml version="1.0"?>
<s:Envelope xmlns:a="http://www.w3.org/2005/08/addressing"
   xmlns:s="http://www.w3.org/2003/05/soap-envelope">
   <s:Header>
      <a:Action s:mustUnderstand="1">
 http://schemas.microsoft.com/windows/management/2012/01/enrollment/IDiscoveryService/Discover
      </a:Action>
      <a:MessageID>urn:uuid: 748132ec-a575-4329-b01b-6171a9cf8478</a:MessageID>
      <a:ReplyTo>
         <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
      </a:ReplyTo>
      <a:To s:mustUnderstand="1">
         https://ENROLLTEST.CONTOSO.COM/EnrollmentServer/Discovery.svc
      </a:To>
   </s:Header>
   <s:Body>
      <Discover xmlns="http://schemas.microsoft.com/windows/management/2012/01/enrollment/">
         <request xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
            <EmailAddress>user@contoso.com</EmailAddress>
            <RequestVersion></RequestVersion>
```

© 2014 Microsoft. All rights reserved.

```
         </request>
      </Discover>
   </s:Body>
</s:Envelope>
```

Response:

The discovery response is in XML format and includes the following elements:

- **Authentication policy (AuthPolicy)**– Indicates what type of authentication is required. The Discovery Service must return "federated". This field is mandatory.
- **Authentication service URL (AuthenticationServiceUrl)** – The URL of the login browser control the enrollment agent will call to authenticate the user. This field is mandatory.
- **Enrollment policy service URL (EnrollmentPolicyServiceUrl)** – Specifies the address of the ES (Enrollment Service) against which the X.509 Certificate Enroll Policy Protocol operations are performed.
- **Enrollment service URL (EnrollmentServiceUrl)** – Specifies the URL of the enrollment endpoint that is exposed by the management service. The enrollment agent will call this URL after the user is authenticated and the login service has posted the SecurityToken back to the enrollment agent. This field is mandatory.

Note: the HTTP server response must not be chunked. It must be sent as one message.

The following example shows a response received from the discovery web service.

Header:

```
HTTP/1.1 200 OK
Content-Length: xxx
Content-Type: application/soap+xml; charset=utf-8
Server: EnterpriseEnrollment.Contoso.com
Date: Tue, 02 Aug 2012 00:32:56 UTC

<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:a="http://www.w3.org/2005/08/addressing">
   <s:Header>
      <a:Action s:mustUnderstand="1">
http://schemas.microsoft.com/windows/management/2012/01/enrollment/IDiscoveryService/DiscoverR
esponse
      </a:Action>
      <ActivityId CorrelationId="48915517-66c6-4ab7-8f77-c8277e45b3cf"
xmlns="http://schemas.microsoft.com/2004/09/ServiceModel/Diagnostics">
        a4067bc9-ce15-446b-a3f7-5ea1006256f5
      </ActivityId>
      <a:RelatesTo>urn:uuid: 748132ec-a575-4329-b01b-6171a9cf8478</a:RelatesTo>
   </s:Header>
   <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <DiscoverResponse
         xmlns="http://schemas.microsoft.com/windows/management/2012/01/enrollment">
         <DiscoverResult>
            <AuthPolicy>Federated</AuthPolicy>
            <AuthenticationServiceUrl>
```

© 2014 Microsoft. All rights reserved.

```
https://portal.manage.contoso.com/LoginRedirect.aspx
            </AuthenticationServiceUrl>
            <EnrollmentPolicyServiceUrl>
               https://enrolltest.contoso.com/ENROLLMENTSERVER/DEVICEENROLLMENTWEBSERVICE.SVC
            </EnrollmentPolicyServiceUrl>
            <EnrollmentServiceUrl>
               https://enrolltest.contoso.com/ENROLLMENTSERVER/DEVICEENROLLMENTWEBSERVICE.SVC
            </EnrollmentServiceUrl>
         </DiscoverResult>
      </DiscoverResponse>
   </s:Body>
</s:Envelope>
```

HTTP errors

The following list specifies the possible faults:

- Redirection 3xx.
- 404 – No CNAME DNS record was registered or a host was not found.
- Server error 5xx.

NOTE: The server SSL certificate must be domain specific. The wildcard SSL certificate (for example, a certificate for *.contoso.com instead of dm.contoso.com) is not accepted by the device.

Web authentication – Security Token Service

After the enrollment agent has successfully received and parsed the DiscoverResponse document from the Discover web server, the next step is to authenticate the user against a security token service (STS). The STS will return a security token representing the authenticated user which the enrollment agent will use in subsequent requests to the enrollment service.

Note: The enrollment agent is agnostic about the protocol flows for authenticating and returning the security token. While the STS might prompt for user credentials directly or enter into a federation protocol with an STS and directory service, the enrollment agent is agnostic to all this. To remain agnostic, all protocol flows about authentication that involve the enrollment client are passive, which means that they are browser-implemented.

Through the Web Authentication Broker (WAB) control, the enrollment agent will ask the WAB to issue an HTTPS request in the following format:

"https://<*AuthenticationServiceUrl*>?appru=<*appid*>&login_hint=<*User Principal Name*>"

- <*AuthenticationServiceUrl*> is the value returned in the AuthenticationServiceUrl element in the DiscoverResponse
- <*appid*> is in the form of: ms-app://<*string*>
- <*User Principal Name*> is the name of the enrolling user, for example, user@contoso.com. The value of this attribute serves as a hint that can be used by the STS as part of the authentication.

© 2014 Microsoft. All rights reserved.

After the authentication is complete, the STS must return an HTML form document that has a POST method action of appid identified in the query string parameter, as shown in the following example.

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Vary: Accept-Encoding
Content-Length: xxx

<!DOCTYPE>
<html>
   <head>
      <title>Working...</title>
      <script>
         function formSubmit() {
            document.forms[0].submit();
         }
           window.onload=formSubmit;
      </script>
   </head>
   <body>
      <form method="post" action="<appid>”>
         <p><input type="hidden" name="wresult" value="<security token value>"/></p>
         <input type="submit"/>
      </form>
   </body>
</html>
```

The security token value is the **base64**-encoded string http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary. This string is opaque to the enrollment client. The client does not interpret the string.

Certificate enrollment policy web service

Description

This web service implements the X.509 Certificate Enrollment Policy Protocol Specification (MS-XCEP) protocol that allows customizing certificate enrollment to match different security needs of enterprises at different times (cryptographic agility). The service processes the **GetPolicies** message from the client, authenticates the client, and returns matching enrollment policies in the **GetPoliciesResponse** message.

Request

The BinarySecurityToken in `GetPolicies` contains the security token returned from the WAB passive authentication.

Header:

```
POST /ENROLLMENTSERVER/DEVICEENROLLMENTWEBSERVICE.SVC HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
User-Agent: Windows Enrollment Client
Host: enrolltest.contoso.com
Content-Length: xxxx
```

© 2014 Microsoft. All rights reserved.

```
Cache-Control: no-cache


<s:Envelope
   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:a="http://www.w3.org/2005/08/addressing"
   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd"
   xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
   xmlns:ac="http://schemas.xmlsoap.org/ws/2006/12/authorization">
   <s:Header>
      <a:Action s:mustUnderstand="1">
         http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy/IPolicy/GetPolicies
      </a:Action>
      <a:MessageID>urn:uuid:72048B64-0F19-448F-8C2E-B4C661860AA0</a:MessageID>
      <a:ReplyTo>
         <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
      </a:ReplyTo>
      <a:To s:mustUnderstand="1">
         https://enrolltest.contoso.com/ENROLLMENTSERVER/DEVICEENROLLMENTWEBSERVICE.SVC
      </a:To>

      <wsse:Security s:mustUnderstand="1">
         <wsse:BinarySecurityToken
            ValueType="http://schemas.microsoft.com/5.0.0.0/
ConfigurationManager/Enrollment/DeviceEnrollmentUserToken"
            EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd#base64binary">
            <!-- Security token removed -->
         </wsse:BinarySecurityToken>
      </wsse:Security>
   </s:Header>
   <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <GetPolicies
         xmlns="http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy">
         <client>
            <lastUpdate xsi:nil="true"/>
            <preferredLanguage xsi:nil="true"/>
         </client>
         <requestFilter xsi:nil="true"/>
      </GetPolicies>
   </s:Body>
</s:Envelope>
```

Response:

After the user is authenticated, the web service retrieves the certificate template the user should enroll with and creates enrollment policies based on the certificate template properties. A sample of the response can be found on MSDN.

MS-XCEP supports very flexible enrollment policies using various Complex Types and Attributes. For Windows, **minimalKeyLength**, **hashAlgorithmOIDReference** policies and the **CryptoProviders** are supported. The **hashAlgorithmOIDReference** has related OID,

© 2014 Microsoft. All rights reserved.

OIDReferenceID and **policySchema** in the **GetPolicesResponse**. The **policySchema** refers to the certificate template version. Certificate template version 3 supports hashing algorithms. For more details, see the Certificate Template Versions topic in the TechNet libaray. (http://technet.microsoft.com/en-us/library/cc725838.aspx)

Be aware that the HTTP server response must not be chunked. It must be sent as one message.

Header:

```
HTTP/1.1 200 OK
Date: Fri, 03 Aug 2012 20:00:00 UTC
Server: <server name here>
Content-Type: application/soap+xml
Content-Length: xxxx
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<s:Envelope
   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"
   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:a="http://www.w3.org/2005/08/addressing">
   <s:Header>
      <a:Action s:mustUnderstand="1">

http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy/IPolicy/GetPoliciesResponse
      </a:Action>
      <ActivityId CorrelationId="08d2997e-e8ac-4c97-a4ce-d263e62186ab"
         xmlns="http://schemas.microsoft.com/2004/09/ServiceModel/Diagnostics">
         d4335d7c-e192-402d-b0e7-f5d550467e3c</ActivityId>
      <a:RelatesTo>urn:uuid: 69960163-adad-4a72-82d2-bb0e5cff5598</a:RelatesTo>
   </s:Header>
   <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <GetPoliciesResponse
         xmlns="http://schemas.microsoft.com/windows/pki/2009/01/enrollmentpolicy">
         <response>
            <policyFriendlyName xsi:nil="true"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
            <nextUpdateHours xsi:nil="true"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
            <policiesNotChanged xsi:nil="true"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
            <policies>
               <policy>
                  <policyOIDReference>0</policyOIDReference>
                  <cAs xsi:nil="true" />
                  <attributes>
                     <policySchema>3</policySchema>
                     <privateKeyAttributes>
                        <minimalKeyLength>2048</minimalKeyLength>
                        <keySpec xsi:nil="true" />
                        <keyUsageProperty xsi:nil="true" />
                        <permissions xsi:nil="true" />
                        <algorithmOIDReference xsi:nil="true" />
                        <cryptoProviders xsi:nil="true" />
                     </privateKeyAttributes>
                     <supersededPolicies xsi:nil="true" />
                     <privateKeyFlags xsi:nil="true" />
```

© 2014 Microsoft. All rights reserved.

```
                    <subjectNameFlags xsi:nil="true" />
                    <enrollmentFlags xsi:nil="true" />
                    <generalFlags xsi:nil="true" />
                    <hashAlgorithmOIDReference>0</hashAlgorithmOIDReference>
                    <rARequirements xsi:nil="true" />
                    <keyArchivalAttributes xsi:nil="true" />
                    <extensions xsi:nil="true" />
                 </attributes>
              </policy>
           </policies>
        </response>
        <cAs xsi:nil="true" />
        <oIDs>
           <oID>
              <value>1.3.14.3.2.29</value>
              <group>1</group>
              <oIDReferenceID>0</oIDReferenceID>
              <defaultName>szOID_OIWSEC_sha1RSASign</defaultName>
           </oID>
        </oIDs>
     </GetPoliciesResponse>
  </s:Body>
</s:Envelope>
```

SOAP faults

If the web service cannot process the request, a SOAP fault is returned with specific fault code and reason, as shown in the following table.

Fault code	Reason
MessageFormatFault	GetPolicies format is invalid
AuthenticationFault	Failed authentication
AuthorizationFault	Failed authorization
InternalServerFault	Internal error such as SQL down
ClientVersionFault	Unsupported version of client

Certificate enrollment web service

Description

This web service implements the MS-WSTEP protocol. It processes the **RequestSecurityToken** (RST) message from the client, authenticates the client, requests the certificate from the CA, and returns it in the **RequestSecurityTokenResponse** (RSTR) to the client. In addition to the issued certificate, the response also contains configuration necessary to provision the OMA-DM agent.

Request

© 2014 Microsoft. All rights reserved.

RequestSecurityToken (RST) must have the authentication token and a certificate request. The UsernameToken in RST is the same as in **GetPolicies**. The BinarySecurityToken in RST contains a Base64-encoded PKCS#10 certificate request, which is generated by the client based on the enrollment policy. The client will request enrollment policy by using MS-XCEP before requesting a certificate using MS-WSTEP. If the PKCS#10 certificate request is accepted by the certification authority (CA) (the key length, hashing algorithm, and so on match the certificate template), the client can enroll successfully.

Be aware that the RequestSecurityToken will use a custom TokenType (http://schemas.microsoft.com/5.0.0.0/ConfigurationManager/Enrollment/DeviceEnrollmentToken), because our enrollment token is more than an X.509 v3 certificate. For more information, see Response, later in this section.

RST may also specify several **AdditionalContext** items, such as **DeviceType** and **Version**. Based on these, the web service can return device-specific and version-specific DM configuration, for example.

Be aware that the policy service and the enrollment service must be on the same server. That is, they must have the same host name.

Here is a sample RST message to show the details.

Header:

```
POST /EnrollmentServer/DeviceEnrollmentWebService.svc HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
User-Agent: Windows Enrollment Client
Host: enrolltest.contoso.com
Content-Length: xxxx
Cache-Control: no-cache
```

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:a="http://www.w3.org/2005/08/addressing"
   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd"
   xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
   xmlns:ac="http://schemas.xmlsoap.org/ws/2006/12/authorization">
   <s:Header>
      <a:Action s:mustUnderstand="1">
         http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RST/wstep
      </a:Action>
      <a:MessageID>urn:uuid:0d5a1441-5891-453b-becf-a2e5f6ea3749</a:MessageID>
      <a:ReplyTo>
         <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
      </a:ReplyTo>
      <a:To s:mustUnderstand="1">
         https://enrolltest.contoso.com:443/ENROLLMENTSERVER/DEVICEENROLLMENTWEBSERVICE.SVC
      </a:To>
      <wsse:Security s:mustUnderstand=”1”>
         <wsse:BinarySecurityToken
```

© 2014 Microsoft. All rights reserved.

```
            ValueType=”http://schemas.microsoft.com/5.0.0.0/
ConfigurationManager/Enrollment/DeviceEnrollmentUserToken”
            EncodingType=”http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd#base64binary”>
            <!-- Token Removed -->
         </wsse:BinarySecurityToken>
      </wsse:Security>
   </s:Header>
   <s:Body>
      <wst:RequestSecurityToken>
         <wst:TokenType>
    http://schemas.microsoft.com/5.0.0.0/ConfigurationManager/Enrollment/DeviceEnrollmentToken
         </wst:TokenType>
         <wst:RequestType>
            http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>
         <wsse:BinarySecurityToken
            ValueType="http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10"
            EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd#base64binary">
            DER format PKCS#10 certificate request in Base64 encoding inserted here
         </wsse:BinarySecurityToken>
         <ac:AdditionalContext xmlns="http://schemas.xmlsoap.org/ws/2006/12/authorization">
            <ac:ContextItem Name="DeviceType">
               <ac:Value>CIMClient_Windows</ac:Value>
            </ac:ContextItem>
            <ac:ContextItem Name="ApplicationVersion">
               <ac:Value>0.0.0.99</ac:Value>
            </ac:ContextItem>
         </ac:AdditionalContext>
      </wst:RequestSecurityToken>
   </s:Body>
</s:Envelope>
```

Response:

After validating the request, the web service looks up the assigned certificate template for the client, forwards the PKCS#10 requests to the CA, processes the response from the CA, constructs an OMA Client Provisioning XML format, and returns it in the **RequestSecurityTokenResponse** (RSTR).

Be aware that the HTTP server response must not be chunked. It must be sent as one message.

Similar to the **TokenType** in the RST, the RSTR will use a custom **ValueType** in the **BinarySecurityToken** (http://schemas.microsoft.com/ConfigurationManager/Enrollment/DeviceEnrollmentProvisionDoc), because the token is more than an X.509 v3 certificate.

The provisioning XML contains:

- (mandatory) The requested certificates
- (mandatory) DM client configuration

The client will install the client certificate, the enterprise root CA certificate, and any intermediate CA certificate(s). The DM configuration includes the name and address of the DM server, which client certificate to use, and schedules when the DM client calls back to the server.

© 2014 Microsoft. All rights reserved.

Here is a sample RSTR message and a sample of OMA client provisioning XML within RSTR. For more information about the configuration service providers (CSPs) used in provisioning XML, see the Enterprise settings, policies, resource access and application management section.

RSTR message:

Header:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: xxxx
Content-Type: application/soap+xml; charset=utf-8
Server: Microsoft-IIS/7.0
Date: Fri, 03 Aug 2012 00:32:59 UTC
```

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:a="http://www.w3.org/2005/08/addressing"
   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
   <s:Header>
      <Action s:mustUnderstand="1" >
         http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RSTRC/wstep
      </Action>
      <a:RelatesTo>urn:uuid:81a5419a-496b-474f-a627-5cdd33eed8ab</a:RelatesTo>
      <o:Security s:mustUnderstand="1" xmlns:o=
         "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <u:Timestamp u:Id="_0">
            <u:Created>2013-08-02T00:32:59.420Z</u:Created>
            <u:Expires>2013-08-02T00:37:59.420Z</u:Expires>
         </u:Timestamp>
      </o:Security>
   </s:Header>
   <s:Body>
      <RequestSecurityTokenResponseCollection
         xmlns="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
         <RequestSecurityTokenResponse>
            <TokenType>
    http://schemas.microsoft.com/5.0.0.0/ConfigurationManager/Enrollment/DeviceEnrollmentToken
            </TokenType>
            <RequestedSecurityToken>
               <BinarySecurityToken
                  ValueType=
"http://schemas.microsoft.com/5.0.0.0/ConfigurationManager/Enrollment/DeviceEnrollmentProvisio
nDoc"
                  EncodingType=
   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd#base64binary"
                  xmlns=
          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
                  B64EncodedBinarySecurityToken
               </BinarySecurityToken>
            </RequestedSecurityToken>
            <RequestID xmlns="http://schemas.microsoft.com/windows/pki/2009/01/enrollment">0
            </RequestID>
         </RequestSecurityTokenResponse>
      </RequestSecurityTokenResponseCollection>
   </s:Body>
</s:Envelope>
```

© 2014 Microsoft. All rights reserved.

```
The following sample shows a wap provisioning document (presented in the previous package as
B64EncodedBinarySecurityToken):
<wap-provisioningdoc version="1.1">
  <!-- This contains information about issued and trusted certificates. -->
  <characteristic type="CertificateStore">
    <!-- This contains trust certificates. -->
    <characteristic type="Root">
      <characteristic type="System">
        <!--The thumbprint of the certificate to be added to the trusted root store -->
        <characteristic type="5BE128213D05DC6CB87A059469130FC6686992EF">
          <!-- Base64 encoding of the trust root certificate -->
          <parm name="EncodedCertificate" value="{encoded root certificate inserted here}" />
        </characteristic>
      </characteristic>
    </characteristic>
      <!-- This contains intermediate certificates. -->
      <characteristic type="CA">
        <characteristic type="System">
        <!—the thumbprint of the intermediate certificate →
        <characteristic type="5DF7DE78255449CFEBD82CD626011982378F40F1">
          <parm name="EncodedCertificate" value=”{encoded intermediate cert inserted here}” />
          </characteristic>
        </characteristic>
      </characteristic>
    <characteristic type="My" >
      <characteristic type="User">
        <!-- Client certificate thumbprint. -->
        <characteristic type="B692158116B7B82EDA4600FF4145414933B0D5AB">
          <!-- Base64 encoding of the client certificate -->
          <parm name="EncodedCertificate" value=”{client certificate inserted here}” />
          <characteristic type="PrivateKeyContainer">
            <parm name="KeySpec" value="2"/>
            <parm name="ContainerName" value="ConfigMgrEnrollment"/>
            <parm name="ProviderType" value="1"/>
          </characteristic>
        </characteristic>
      </characteristic>
    </characteristic>
  </characteristic>

  <!-- Contains information about the management service and configuration for the management
agent -->
  <characteristic type="APPLICATION">
    <parm name="APPID" value="w7"/>
    <!-- Management Service Name. -->
    <parm name="PROVIDER-ID" value="Contoso Management Service"/>
    <parm name="NAME" value="BecMobile"/>
    <!-- Link to an application that the management service may provide eg a Windows Store
application link. The Enrollment Client may show this link in its UX.-->
    <parm name="SSPHyperlink" value="http://go.microsoft.com/fwlink/?LinkId=255310" />
    <!-- Management Service URL. -->
    <parm name="ADDR" value="https://ContosoManagementService.com/MDMHandler"/>
    <parm name="ServerList" value="https://ContosoManagementService.com/MDMHandler" />
    <parm name="ROLE" value="4294967295"/>
    <!-- Discriminator to set whether the client should do Certificate Revocation List
checking. -->
    <parm name="CRLCheck" value="0"/>
    <parm name="CONNRETRYFREQ" value="6" />
    <parm name="INITIALBACKOFFTIME" value="30000" />
```

© 2014 Microsoft. All rights reserved.

```
    <parm name="MAXBACKOFFTIME" value="120000" />
    <parm name="BACKCOMPATRETRYDISABLED" />
    <parm name="DEFAULTENCODING" value="application/vnd.syncml.dm+wbxml" />
    <!-- Search criteria for client to find the client certificate using subject name of the
certificate -->
    <parm name="SSLCLIENTCERTSEARCHCRITERIA" value="Subject=CN%3de4c6b893-07a7-4b24-878e-
9d8602c3d289&amp;Stores=MY%5CUser"/>
    <characteristic type="APPAUTH">
      <parm name="AAUTHLEVEL" value="CLIENT"/>
      <parm name="AAUTHTYPE" value="DIGEST"/>
      <parm name="AAUTHSECRET" value="dummy"/>
      <parm name="AAUTHDATA" value="nonce"/>
    </characteristic>
    <characteristic type="APPAUTH">
      <parm name="AAUTHLEVEL" value="APPSRV"/>
      <parm name="AAUTHTYPE" value="DIGEST"/>
      <parm name="AAUTHNAME" value="dummy"/>
      <parm name="AAUTHSECRET" value="dummy"/>
      <parm name="AAUTHDATA" value="nonce"/>
    </characteristic>
  </characteristic>
  <!-- Extra Information to seed the management agent’s behavior . -->
  <characteristic type="Registry">
    <characteristic type="HKLM\Security\MachineEnrollment">
      <parm name="RenewalPeriod" value="363" datatype="integer" />
    </characteristic>
    <characteristic type="HKLM\Security\MachineEnrollment\OmaDmRetry">
      <!-- Number of retries if client fails to connect to the management service. -->
      <parm name="NumRetries" value="8" datatype="integer" />
      <!--Interval in minutes between retries. -->
      <parm name="RetryInterval" value="15" datatype="integer" />
      <parm name="AuxNumRetries" value="5" datatype="integer" />
      <parm name="AuxRetryInterval" value="3" datatype="integer" />
      <parm name="Aux2NumRetries" value="0" datatype="integer" />
      <parm name="Aux2RetryInterval" value="480" datatype="integer" />
    </characteristic>
  </characteristic>
  <!-- Extra Information about where to find device identity information.  This is redundant
in that it is duplicative to what is here, but it is required in the current version of the
protocol. -->
  <characteristic type="Registry">
    <characteristic type="HKLM\Software\Windows\CurrentVersion\MDM\MachineEnrollment">
      <parm name="DeviceName" value="" datatype="string" />
    </characteristic>
  </characteristic>
  <characteristic type="Registry">
    <characteristic type="HKLM\SOFTWARE\Windows\CurrentVersion\MDM\MachineEnrollment">
      <!--Thumbprint of root certificate. -->
      <parm name="SslServerRootCertHash" value="5BE128213D05DC6CB87A059469130FC6686992EF"
datatype="string" />
      <!-- Store for device certificate. -->
      <parm name="SslClientCertStore" value="MY%5CSystem" datatype="string" />
      <!--  Common name of issued certificate. -->
      <parm name="SslClientCertSubjectName" value="CN%3de4c6b893-07a7-4b24-878e-9d8602c3d289"
datatype="string" />
      <!--Thumbprint of issued certificate. -->
      <parm name="SslClientCertHash" value="B692158116B7B82EDA4600FF4145414933B0D5AB"
datatype="string" />
    </characteristic>
    <characteristic
type="HKLM\Security\Provisioning\OMADM\Accounts\037B1F0D3842015588E753CDE76EC724">
```

© 2014 Microsoft. All rights reserved.

```
      <parm name="SslClientCertReference"
value="My;System;B692158116B7B82EDA4600FF4145414933B0D5AB" datatype="string" />
    </characteristic>
  </characteristic>
</wap-provisioningdoc>
```

Note 1: All characteristics in the example wap-provisioningdoc must exist in the document returned by the enrollment service even though some characteristics are ignored by the OMA-DM agent.

CertificateStore Characteristic

The characteristic type CertificateStore is the element provided for the enrollment service to return a root certificate and client certificate for the management agent to use for authenticating to the management service. Certificate TLS authentication is the only authentication method supported by the maangement agent. If intermediate certificates (certificate chain) exist between the device certfiicate and the root certificae, they must be included in the provisioning document as well. The sample here includes an intermediate certificate for completeness.

APPLICATION Characteristic

The characteristic type APPLICATION is the element provided to bootstrap and configure the management (OMA-DM) agent.

The following table describes the parameters that must exist beneath the APPLICATION characteristic.

APPLICATION Parameter	Description
APPID	Must be set to “w7”
PROVIDER-ID	A unique ID representing the management service.
NAME	A unique NAME representing the displayable name of the management service.
SSPHyperlink	Unused but must be present.
ADDR	Represents the primary management endpoint. After enrollment is complete, the OMA-DM agent will immediately attempt to initiate a mangement session with this endpoint.
ServerList	This parameter is useful if the mangement service exposes more than one management endpoint address. At minimum one management endpoint value must exist and typically is the same as the ADDR parameter here. When more than one endpoint is returned, they are separated by “;” (semicolons).

© 2014 Microsoft. All rights reserved.

ROLE	Unused but must be present
CRLCheck	Set to **1** if the OMA-DM agent should perform a certificate revocation list check before attempting TLS authentication. Set to **0** to skip the check.
CONNRETRYFREQ	Unused but must be present.
INITIALBACKOFFTIME	Unused but must be present.
MAXBACKOFFTIME	Unused but must be present.
BACKCOMPATRETRYDISABLED	Unused. An empty parm element that has this attribute name must exist.
DEFAULTENCODING	Must be set to "application/vnd.syncml.dm+wbxml" Note:vnd.syncml.dm+wbxml is not supported. The service must return all SyncML messages that have the Content-Type set to "application/vnd.syncml.dm+xml"
SSLCLIENTCERTSEARCHCRITERIA	Used for selecting the certificate to be used for client authentication. The search is based on the subject attribute of the signed user certificate

HKLM\Security\MachineEnrollment Characteristic

This characteristic must be exist and is used to configure the client certificate renewal period.

The following table describes the parameter that must exist beneath the MachineEnrollment characteristic.

MachineEnrollment Parameter	Description
RenewalPeriod	The number of days before the certificate expires when the OMA-DM agent will attempt to renew the certificate. For example, if the certificate expires in 30 days and you want the renewal to be attempted 5 days before it expires, then on the 25th day, you would specify this value as 5

HKLM\Security\MachineEnrollment\OmaDmRetry Characteristic

This characteristic must be exist and is used to configure the OMA-DM agent's retry behavior.

© 2014 Microsoft. All rights reserved.

The following table describes the parameters that must exist beneath the OmaDmRetry characteristic.

OmaDmRetry Parameter	Description
NumRetries	After enrollment successfully completes, the OMA-DM agent will contact the management service endpoint. *NumRetries* specifies the number of sessions the OMA-DM agent will establish with the service. The label is misleading and should read something like *NumSessions*. Note: the current version of the agent is reading the AuxNumRetries. This value must be present and should be set to the same value as *AuxNumRetries*.
RetryInterval	The time in minutes the OMA-DM agent will wait for establishing a new session with the management service. Note: the current version of the agent is only reading the *AuxRetryInterval*. This value must be present and should be set to the same value as *AuxRetryInterval*.
AuxNumRetries	After enrollment successfully completes, the OMA-DM agent will contact the management service endpoint. *AuxNumRetries* specifies the number of sessions the OMA-DM agent will establish with the service.
AuxRetryInterval	The time in minutes the OMA-DM agent will wait for establishing a new session with the management service.
Aux2NumRetries	Unused but must be present
Aux2RetryInterval	Unused but must be present

Note 1: The APPAUTH characteristic is ignored by the OMA-DM agent but must exist in the current version of the protocol.

Note 2: The HKLM\SOFTWARE\Windows\CurrentVersion\MDM\MachineEnrollment characteristic information is duplicative of other characteristics in the wap provisioning document but must exist in the current version of the document.

Certificate enroll on behalf of web service

Description

This web service is a modification to the Certificate enrollment web service detailed in the previous section. The purpose of this modification is to enable an IT manager to enroll a device on behalf of an existing low rights user on the device.

© 2014 Microsoft. All rights reserved.

In addition to the modification of the enrollment protocol, several registry keys must be populated to aid the enrollment agent in the enroll on behalf of process.

- MachineMDMEnrollment is set to 1.
- MachineMDMEnrollmentUserUPN is set to a value which the server can identify and associate with the locally-managed user.
- MachineMDMEnrollmentUserSID is set to the SID of the local user for which the administrator is enrolling on behalf.

For example, the registry keys are defined as follows:

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\CurrentVersion\MDM]

"MachineMDMEnrollment"=dword:00000001

"MachineMDMEnrollmentUserUPN"="joe@contoso.com"

"MachineMDMEnrollmentUserSID"="S-1-5-21-425223123-4157917690-3751521321-1002"

The enroll on behalf of web service implements the MS-WSTEP protocol. It processes the **RequestSecurityTokenOnBehalfOf** from the enrollment agent, authenticates the agent, requests the certificate from the CA, and then returns the certificate in the **RequestSecurityTokenResponse** (RSTR) to the agent. In addition to the issued certificate, the response also contains configuration necessary to provision the OMA-DM agent.

Request

The **RequestSecurityTokenOnBehalfOf** message is the same as a **RequestSecurityToken** message except for adding an additional <ContextItem> to the message.

```
<ac:ContextItem Name="EnrollmentOnBehalfOfUser">
   <ac:Value>{UPN}</ac:Value>
</ac:ContextItem>
```

UPN – The UPN of the user for which the principal is enrolling on behalf of.

Here is a sample RequestSecurityTokenOnBehalfOf message to show the details.

Header:

```
POST /EnrollmentServer/DeviceEnrollmentWebService.svc HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
User-Agent: Windows Enrollment Client
Host: enrolltest.contoso.com
Content-Length: xxxx
Cache-Control: no-cache
```

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:a="http://www.w3.org/2005/08/addressing"
```

© 2014 Microsoft. All rights reserved.

```
   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd"
   xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
   xmlns:ac="http://schemas.xmlsoap.org/ws/2006/12/authorization">
   <s:Header>
      <a:Action s:mustUnderstand="1">
         http://schemas.microsoft.com/windows/pki/2009/01/enrollment/RST/wstep
      </a:Action>
      <a:MessageID>urn:uuid:0d5a1441-5891-453b-becf-a2e5f6ea3749</a:MessageID>
      <a:ReplyTo>
         <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
      </a:ReplyTo>
      <a:To s:mustUnderstand="1">
         https://enrolltest.contoso.com:443/ENROLLMENTSERVER/DEVICEENROLLMENTWEBSERVICE.SVC
      </a:To>
      <wsse:Security s:mustUnderstand=”1”>
         <wsse:BinarySecurityToken
            ValueType=”http://schemas.microsoft.com/5.0.0.0/
ConfigurationManager/Enrollment/DeviceEnrollmentUserToken”
            EncodingType=”http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd#base64binary”>
            <!-- Token Removed -->
         </wsse:BinarySecurityToken>
      </wsse:Security>
   </s:Header>
   <s:Body>
      <wst:RequestSecurityTokenOnBehalfOf>
         <wst:TokenType>
    http://schemas.microsoft.com/5.0.0.0/ConfigurationManager/Enrollment/DeviceEnrollmentToken
         </wst:TokenType>
         <wst:RequestType>
            http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>
         <wsse:BinarySecurityToken
            ValueType="http://schemas.microsoft.com/windows/pki/2009/01/enrollment#PKCS10"
            EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd#base64binary">
            DER format PKCS#10 certificate request in Base64 encoding inserted here
         </wsse:BinarySecurityToken>
         <ac:AdditionalContext xmlns="http://schemas.xmlsoap.org/ws/2006/12/authorization">
            <ac:ContextItem Name="DeviceType">
               <ac:Value>CIMClient_Windows</ac:Value>
            </ac:ContextItem>
            <ac:ContextItem Name="ApplicationVersion">
               <ac:Value>0.0.0.99</ac:Value>
            </ac:ContextItem>
            <ac:ContextItem Name="EnrollmentOnBehalfOfUser">
               <ac:Value>
                 {the UPN of the user for which the principal is enrollong on behalf}
               </ac:Value>
            </ac:ContextItem>
         </ac:AdditionalContext>
      </wst:RequestSecurityTokenOnBehalfOf>
   </s:Body>
</s:Envelope>
```

© 2014 Microsoft. All rights reserved.

Certificate renewal

The OMA-DM agent always initiates the certificate renewal process with the Enrollment Service. It's typically referred to as "renew on behalf of" or ROBO. The OMA-DM agent renews the device certificate on behalf of the user. Windows 8.1 does not support any workflow where the user can initiate the renewal of the device certificate.

For the certificate renewal process, the **RequestSecurityToken** message is modified. The renewal request specifies a different RequestType from the initial enrollment request (Renew instead of Issue). It also uses a different BinarySecurityToken ValueType (PKCS#7 instead of PKCS#10).

wst:RequestType: The <wst:RequestType> element must be: http://docs.oasis-open.org/ws-sx/ws-trust/200512/Renew. For example:

```
<wst:RequestType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/Renew</wst:RequestType>
```

For more information, see [WSTrust1.3] section 3.1 (http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html)

wsse:BinarySecurityToken/attributes/ValueType: The <wsse:BinarySecurityToken> ValueType attribute must be: http://schemas.microsoft.com/windows/pki/2009/01/ enrollment#PKCS7.

Because the enrollment client uses the existing client certificate to perform client Transport Layer Security (TLS), the security token is not populated in the SOAP header. Therefore, the Enrollment Service (ES) is required to support client TLS.

Response for certificate renewal

When **RequestType** is Renew, the web service verifies the following (in additional to initial enrollment):

- The signature of the PKCS#7 BinarySecurityToken is correct
- The client's certificate is in the renewal period
- The certificate was issued by the enrollment service
- The requester is the same as the requester for initial enrollment
- For standard client's request, the client hasn't been blocked

After validations, the web service retrieves the PKCS#10 content from the PKCS#7 BinarySecurityToken. The rest is the same as initial enrollment, except that the Provisioning XML only requires the new certificate issued by the CA.

Be aware that the HTTP server response must not be chunked. It must be sent as one message.

```
<wap-provisioningdoc version="1.1">
  <!-- This contains information about issued certificate. -->
  <characteristic type="CertificateStore">
```

© 2014 Microsoft. All rights reserved.

```
      <characteristic type="My" >
        <characteristic type="User">
          <!-- Certificate thumbprint. -->
          <characteristic type="{certificate thumbprint inserted here}"/>
            <!-- Base64 encoding of issued certificate. -->
            <parm name="EncodedCertificate" value="{encoded certificate inserted here}" />
          </characteristic>
        </characteristic>
      </characteristic>
    </characteristic>
  </wap-provisioningdoc>
```

SOAP faults

If the web service cannot process the request, a SOAP fault is returned with specific fault code and reason.

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:a="http://www.w3.org/2005/08/addressing" xmlns:u=
   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   <s:Header>
      <a:Action s:mustUnderstand="1" xmlns:a="http://www.w3.org/2005/08/addressing"
         xmlns:s="http://www.w3.org/2003/05/soap-envelope">
         http://www.w3.org/2005/08/addressing/soap/fault
      </a:Action>
      <a:RelatesTo xmlns:a="http://www.w3.org/2005/08/addressing">
         urn:uuid:2d37bdb7-e4ac-4bb8-bca3-29cc9f5cf6b4
      </a:RelatesTo>
      <o:Security s:mustUnderstand="1" xmlns:o=
         "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <u:Timestamp u:Id="_0">
            <u:Created>2012-9-27T04:20:03.408Z</u:Created>
            <u:Expires>2012-9-27T04:25:03.408Z</u:Expires>
         </u:Timestamp>
      </o:Security>
   </s:Header>
   <s:Body>
      <s:Fault>
         <s:Code>
            <s:Value>s:Receiver</s:Value>
            <s:Subcode>
               <s:Value>s:MessageFormat</s:Value>
            </s:Subcode>
         </s:Code>
         <s:Reason>
            <s:Text xml:lang="en-US">Invalid TokenType in request</s:Text>
         </s:Reason>
      </s:Fault>
   </s:Body>
</s:Envelope>
```

This table lists the SOAP fault codes and reasons.

Fault code	Reason
MessageFormatFault	RST format is invalid
AuthenticationFault	Failed authentication

© 2014 Microsoft. All rights reserved.

Fault code	Reason
AuthorizationFault	Failed authorization
CertificateRequestFault	CA denied certificate request
InternalServerFault	Internal error such as SQL down
InvalidRenewalRequesterFault	Request for renewal differs from initial enrollment requester
RenewalWindowFault	Certificate is not in renewal window
ClientVersionFault	Unsupported version of client
NotReachedRenewalWindow	Renewal request isn't within the renewal window. Requester: user thumbprint: <*thumbprint inserted here*>

Best practice tips

General notes

All POSTs should have HTTP Content-Type **application/soap+xml; charset=utf-8** for the discovery and enrollment phase. For SyncML, the correct content type is **application/vnd.syncml.dm+xml**. The type **application/vnd.syncml.dm+wbxml** is not supported.

HTTP 1.1 Content-Encoding using Chunked is not supported. The server should explicitly specify the Content-Length header accordingly in all responses, which turns off the chunked response generation in most frameworks.

Certificates

Web server SSL certificate

The SSL certificate on the discovery and enrollment server must be subdomain specific, in other words, wildcard certificates can't be used. A wildcard certificate will prevent the discovery phase from completing, and the device won't follow up with the enrollment call.

For example, if you're hosting your device management discovery system at discovery.contoso.com, the certificate must be issued for that particular subdomain. A wildcard certificate for *.contoso.com can't be used.

The web server certificate must also have certain X.509 v3 extensions before the device accepts it. The discovery and enrollment phases could have more relaxed criteria for the certificate, but the SyncML POST must have an exact set of extensions present. The following table shows the required extensions.

© 2014 Microsoft. All rights reserved.

Extension	Description	Value
Key Usage	Lists the permitted uses of this particular certificate.	Digital Signature, Key Encipherment
Extended Key Usage	Extensions for the key usage.	TLS Web Server Authentication
Subject Alternative Name	Alternative subjects for this certificate. Used for listing the alternate domains for which the certificate can be used.	DNS entries for each subdomain you're using this certificate for. For example, "DNS:secure.mydmpoc.net, DNS:enterpriseenrollment.mydiscovery.net"

To avoid the need to manually install server certificates, it is best if the certificates for the web server, MDM server, and client chain to the same root CA as those that are installed during enrollment or to another root CA already trusted by the device.

Signed client certificate

The client certificate that is provisioned to the client in the provisioning XML should also have certain X.509 v3 extensions. The following extensions are required.

Extension	Description	Value
Key Usage	Lists the permitted uses of this particular certificate.	Digital Signature
Extended Key Usage	Extensions for the key usage.	TLS Web Client Authentication
Subject Key Identifier	Provides a means for identifying certificates that contain a particular public key.	

Be aware that you should also specify the subject of the certificate in such a way that you can reference the certificate in the provisioning XML's **SSLCLIENTCERTSEARCHCRITERIA** parameter.

Be aware that when creating a certificate, the server should set meaningful/distinguished common name instead of some well known GUIDs.

For more information about how to embed the certificate to the provisioning XML, see the Response section of the Certificate enrollment web service section, earlier in this document.

© 2014 Microsoft. All rights reserved.

Disconnecting from the management infrastructure (unenrollment)

Disconnecting may be initiated either locally by the user from the device or remotely by the IT admin by using management server. User-initiated disconnection is performed much like the initial connection, and it is initiated from the same location in the modern settings application (PC Settings->Network->Workplace). Users may choose to disconnect for any number of reasons, including leaving the company or getting a new device and no longer needing access to their LOB apps on the old device. When an admin initiates a disconnection, a push notification is sent to the device which wakes up the OMA-DM agent. The OMA-DM agent will immediately initiate an OMA-DM session with the management service. Admins may choose to disconnect a user's device after they've left the company or because the device is regularly failing to comply with the organization's security settings policy.

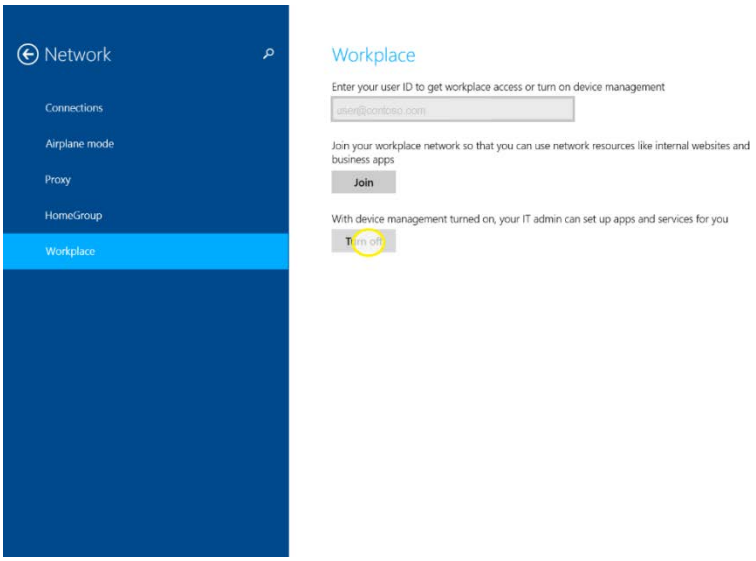
During disconnection, the client does the following:

- Removes the enterprise side-loading key that enabled installing and running LOB apps.
- Removes all LOB applications that were installed by the DM service.
- Removes certificates that were installed by the DM service.
- Removes Wi-Fi and VPN profiles configured by the DM service.
- Removes the device management client configuration and other setting configuration added by DM service, including the scheduled maintenance task. The agent remains dormant unless the user re-enrolls the device to the management service.
- Removes encryption keys associated to the enterprise domain(s).
- Reports successful initiated disassociation to the management infrastructure if the admin initiated the process.

© 2014 Microsoft. All rights reserved.

User-initiated disconnection

The following screen shot shows shows the user experience of disconnection from enterprise management.



Note: The OMA-DM agent does not notify the management agent when an end-user removes the device from management. If the managed user was enrolled into management by using the enroll on behalf of method, they do not have the option to remove the device from management. Only managed users who are Administrators on the device can remove the device from management.

IT admin–requested disconnection

The OMA-DM agent supports the IT admin initiated remote unenroll scenario. The remote unenroll scenario typically consists of the following workflow.

- From the DM service management console, the IT admin locates the device and selects a remote un-enroll
- The DM Service creates a policy action for un-enrolling the targeted device.
- The DM Service then issues a push notification targeted at the device to wake it up.
- The OMA-DM agent wakes up and initiates a OMA-DM session with the DM service.
- The DM Service sends a un-enroll command to the agent
- The OMA-DM agent processes the un-enroll effectively removing the device from management.

The actual SyncML command to un-enroll a device is discussed later in the document.

© 2014 Microsoft. All rights reserved.

Note: the user of the device is not notified when the device is removed from management.

© 2014 Microsoft. All rights reserved.

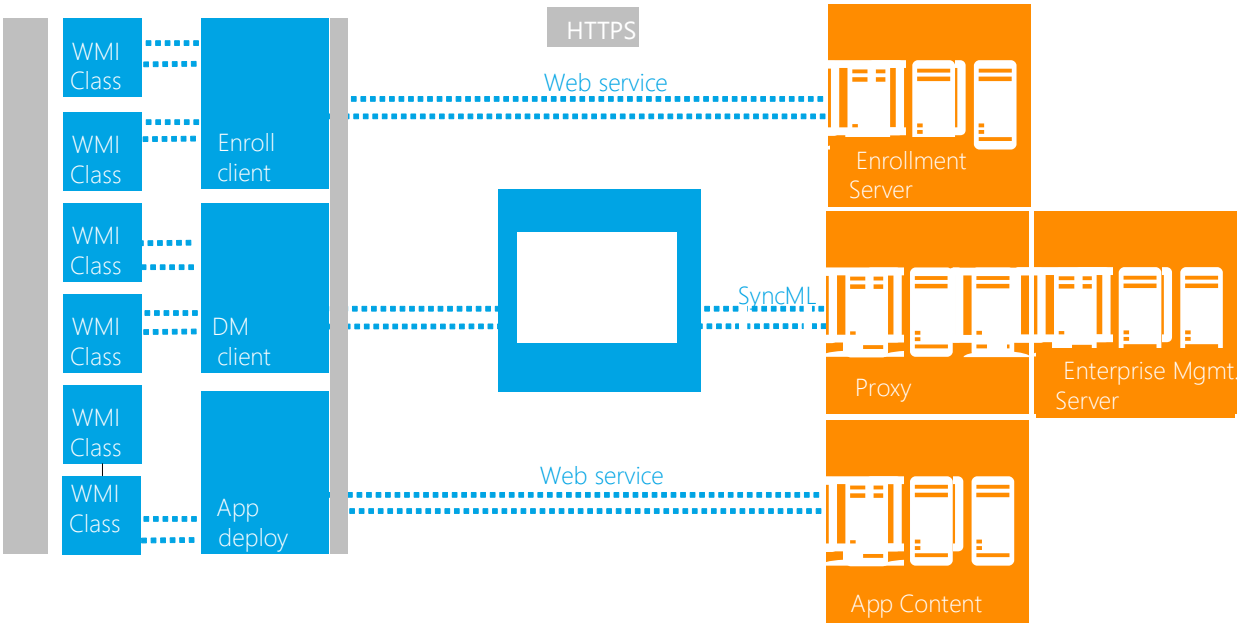
Enterprise settings, policies, resource access and application management

The actual management interaction between the device and server is performed by using the DM agent. The DM agent communicates with the enterprise management server by using DM v1.2 SyncML syntax. The OMA website (http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases) provides the protocol details.

Windows 8.1 currently supports enrolling the device in one MDM service. The DM client that is configured by using the enrollment process is granted access to enterprise related settings. Enterprise MDM settings are exposed by using an approved list of remote WMI classes. Those classes are described in this section.

The DM agent is also configured during the enrollment process to be invoked by the task scheduler to periodically poll the MDM server. Additionally, the enterprise management server/service can send a push notification to the device, through Windows Notification Service (WNS), which forces the DM agent to connect to the server immediately.

The following diagram shows the work flow between server and client.



Management work flow

This protocol defines an HTTPS-based client/server communication with SyncML as the package payload that carries management requests and execution results. The configuration request is addressed by using a remote WMI class on the device. The DM client maintains an approved list

© 2014 Microsoft. All rights reserved.

of WMI classes on the device that are available for management. There is an approved list for read access classes and also an approved list for write access classes.

To facilitate security-enhanced communication with the remote server for enterprise management, Windows 8.x supports certificate-based mutual authentication over an encrypted SSL HTTP channel between the DM agent and management service. The server and client certificates are provisioned during the enrollment process.

DM client configuration, company policy enforcement, business application management, and device inventory are all exposed or expressed by using remote WMI classes on the device. The DM client communicates with the server and sends configuration request to WMI classes. The server must understand which WMI classes are exposed for MDM and also how to address the properties in each class.

As a summary, the following are the DM tasks that an organization's management server could support:

- Device Inventory: The MDM service can query the device for hardware inventory. Examples of hardware that can be queried are as follows: Processor, Bios, Battery, Physical Memory and Network Adapter.
- Enterprise Application Management: The MDM service can use this service to install line of business applications on the device and query for line of business applications that are installed on the device through MDM. The MDM service can also create web links and remote application links on the secondary Start Screen.
- Wi-Fi and VPN Profile Management.
- Certificate Management: The MDM service can install new ROOT and CA certificates through the MDM channel. The MDM service can also set up a certificate provisioning request through SCEP.
- Password policy.
- Device restrictions.
- Data protection: Remote device lock and remote device un-enroll.

© 2014 Microsoft. All rights reserved.

DM SyncML functionality support

This section describes the OMA DM functionalities that the DM client supports. Be aware that for enterprise device management, not all OMA DM client functions are needed. The full description of the OMA DM protocol v1.2 can be found at the OMA website (http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases).

OMA DM standards

The following table shows the OMA DM standards that Windows 8.1 uses.

General area	OMA DM standard that is supported
Data transport and session	Client-initiated remote HTTPS DM session over SSL.
Bootstrap XML	OMA Client Provisioning profile.

© 2014 Microsoft. All rights reserved.

General area	OMA DM standard that is supported
DM protocol commands	The following list shows the commands that are supported by a Windows 8.1-based device. For more information about the OMA DM command elements, see SyncML Representation Protocol Device Management Usage (OMA-SyncML-DMRepPro-V1_1_2-20030613-A) available from the OMA website (http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases). • Add (Implicit Add supported). • Alert (DM alert): server-initiated management alert (1200) (not used by enterprise management), session abort (1223), UI Alerts (1100, 1101, 1102, 1103, 1104) (not used by enterprise management), generic alert (1226) (only used by enterprise management client when the user triggers a MDM unenrollment action from the device). The current agent in Windows 8.1 will also send an Alert (0) when initiating a DM session. • Atomic: Be aware that performing an Add command followed by Replace on the same node within an Atomic element is not supported. Nested Atomic and Get commands are not allowed and will generate error code 500. • Delete: Removes a node from the DM tree, and the entire subtree beneath that node if one exists. • Exec: Invokes a static method on a WMI class. • Get: Retrieves data from the device; for interior nodes, the child node names in the Data element are returned in URI-encoded format. • Replace: Overwrites data on the device. • Result: Returns the data results of a Get command to the DM server. • Status: Indicates the completion status (success or failure) of an operation. If an XML element that is not a valid OMA DM command is under one of the following elements, the status code 400 is returned for that element: • SyncBody. • Atomic. If no CmdID is provided in the DM command, the client returns blank in the status element and the status code 400. If Atomic elements are nested, the following status codes are returned: • The nested Atomic command returns 500. • The parent Atomic command returns 507. Note: Performing an Add command followed by Replace on the same node within an Atomic element is not supported. LocURI cannot start with "/". Meta XML tag in SyncHdr is ignored.

© 2014 Microsoft. All rights reserved.

General area	OMA DM standard that is supported
OMA DM standard objects	The standard way to address device resources is through a remote WMI class but the Windows 8.1 DM client does support the following OMA-DM objects. • DevInfo • DevDetail
Security	• vSSL level certificate-based client/server authentication, encryption, and data integrity check
Nodes	For the OMA DM standard objects supported by the DM client, the following rules apply for the node name: • "." can be part of the node name. • The node name cannot be empty. • The node name cannot be only the asterisk (*) character.
Provisioning files	Provisioning XML must be well formed and follow the definition in SyncML Representation Protocol specification. If an XML element that is not a valid OMA DM command is under SyncBody, the status code 400 is returned for that element. Note: To represent a Unicode string as a URI, first encode the string as UTF-8. Then encode each of the UTF-8 bytes using URI encoding.

OMA DM protocol common elements

Common elements are used by other OMA DM element types. The following table lists the OMA DM common elements that are used to configure Windows-based devices. For more information about OMA DM common elements, see "SyncML Representation Protocol Device Management Usage" on the OMA website (http://technical.openmobilealliance.org/Technical/release_program/docs/DM/V1_1_2-20040113-A/OMA-SyncML-DMRepPro-V1_1_2-20030613-A.pdf).

Element	Description
Cmd	Specifies the name of an OMA DM command referenced in a Status element.
CmdID	Specifies the unique identifier for an OMA DM command.
CmdRef	Specifies the ID of the command for which status or results information is being returned. This element takes the value of the CmdID element of the corresponding request message.

© 2014 Microsoft. All rights reserved.

Element	Description
Final	Indicates that the current message is the last message in the package.
LocURI	Specifies the address of the target or source location.
MsgID	Specifies a unique identifier for an OMA DM session message.
MsgRef	Specifies the ID of the corresponding request message. This element takes the value of the request message MsgID element.
RespURI	Specifies the URI that the recipient must use when sending a response to this message.
SessionID	Specifies the identifier of the OMA DM session associated with the containing message.
Source	Specifies the message source address.
SourceRef	Specifies the source of the corresponding request message. This element takes the value of the request message Source element and is returned in the Status or Results element.
Target	Specifies the address of the node, in the DM Tree, that is the target of the OMA DM command.
TargetRef	Specifies the target address in the corresponding request message. This element takes the value of the request message Target element and is returned in the Status or Results element.
VerDTD	Specifies the major and minor version identifier of the OMA DM representation protocol specification used to represent the message.
VerProto	Specifies the major and minor version identifier of the OMA DM protocol specification used with the message.

Device Management session

A DM session consists of a series of commands exchanged between a DM server and a Windows-based device. The server sends commands indicating operations to be performed on the device. The device responds by sending commands that contain the results and any requested status information.

An example of a short DM session would be the following:

A server sends a Get command to a device to retrieve the contents of a property in a WMI class. The device performs the operation and responds with a Result command that contains the requested contents.

A DM session can be divided into two phases:

© 2014 Microsoft. All rights reserved.

- Setup phase: In response to a trigger event, a device sends an initiating message to a DM server. The device and server exchange needed authentication and device information. This phase is represented by steps 1, 2, and 3 in the following table.
- Management phase: The DM server is in control. It sends management commands to the device, and the device responds. Phase two ends when the DM server stops sending commands and terminates the session. This phase is represented by steps 3, 4, and 5 in the following table.

The following table shows the sequence of events during a typical DM session.

Note: The step numbers in the table do not represent message identification numbers (MsgID). All messages from the server must have a MsgID that is unique within the session, starting at 1 for the first message and increasing by an increment of 1 for each additional message. For more information about MsgID and OMA SyncML protocol, see "OMA Device Management Representation Protocol" on the OMA website (http://technical.openmobilealliance.org/Technical/technical-information/release-program/release-program-copyright-notice?rp=1374&r_type=technical&fp=Technical/Release_Program/docs/DM/V1_2_1-20080617-A/OMA-TS-DM_RepPro-V1_2_1-20080617-A.pdf).

Step	Action	Description
1	The maintenance task schedule invokes the DM client.	At the scheduled time, the DM client is invoked periodically to call back to the enterprise management server over HTTPS.
2	The device sends a message, over an IP connection (HTTPS), to initiate the session.	This message includes the device certificate and device information. The client and server do certificate-based authentication over an SSL channel.
3	The DM server responds, over an IP connection (HTTPS).	The server sends initial device management commands, if any.
4	The device responds to server management commands.	This message includes the results of performing the specified device management operations.
5	The DM server terminates the session or sends an additional command or commands.	The DM session ends, or step 4 is repeated.

© 2014 Microsoft. All rights reserved.

OMA DM provisioning files

OMA DM commands are transmitted between the server and the device in messages. A message can contain one or more commands. For a list of commands supported in Windows 8.1, see the table in "OMA DM standards" earlier in this section.

A DM message is an XML document. The structure and content of the document is defined in the OMA DM Representation Protocol (OMA-SyncML-DM_RepPro-V1_2_1-20080617-A.pdf) available on the OMA website (http://technical.openmobilealliance.org/Technical/technical-information/release-program/release-program-copyright-notice?rp=1374&r_type=technical&fp=Technical/Release_Program/docs/DM/V1_2_1-20080617-A/OMA-TS-DM_RepPro-V1_2_1-20080617-A.pdf).

Each message is composed of a header, specified by the **SyncHdr** element, and a message body, specified by the **SyncBody** element.

The following table shows the OMA DM versions that are supported in Windows 8.1.

Version	Format
OMA DM version 1.2	<SyncML xmlns='SYNCML:SYNCML1.2'> </SyncML>

File format

The following example shows the general structure of the XML document sent by the server, using OMA DM version 1.2 for demonstration purposes only. The initial XML packages exchanged between client and server could contain additional XML tags. For a detailed description and samples for those packages, see the OMA Device Management Protocol 1.2.1 specification (http://technical.openmobilealliance.org/Technical/technical-information/release-program/release-program-copyright-notice?rp=1374&r_type=technical&fp=Technical/Release_Program/docs/DM/V1_2_1-20080617-A/OMA-TS-DM_Protocol-V1_2_1-20080617-A.pdf).

```
<SyncML xmlns='SYNCML:SYNCML1.2'>
   <SyncHdr>
      <VerDTD>1.2</VerDTD>
      <VerProto>DM/1.2</VerProto>
      <SessionID>1</SessionID>
      <MsgID>1</MsgID>
      <Target>
         <LocURI>{unique device ID}</LocURI>
      </Target>
      <Source>
         <LocURI>https://www.thephone-company.com/mgmt-server</LocURI>
      </Source>
   </SyncHdr>
   <SyncBody>
      <!-- query a device OS system version -->
      <Get>
         <CmdID>2</CmdID>
         <Item>
```

© 2014 Microsoft. All rights reserved.

```
            <Target>
               <LocURI>./DevDetail/SwV</LocURI>
            </Target>
         </Item>
      </Get>
      <!-- Update device policy -->

      <Final />
   </SyncBody>
</SyncML>
```

For more information about the header and body, see **SyncHdr** and **SyncBody** on MSDN.

SyncHdr element

SyncHdr includes the following information:

- Document Type Definition (DTD) and protocol version numbers
- Session and message identifiers. Be aware that each message in the same DM session must have a different MsgID.
- Message source and destination Uniform Resource Identifiers (URIs)

This information is used to by the device to appropriately manage the DM session.

Code example

The following example shows the header component of a DM message. In this case, OMA DM version 1.2 is used as an example only.

> Note: The <LocURI> node value for the <Source> element in the SyncHdr of the device-generated DM package should be the same as the value of ./DevInfo/DevID. For more information about DevID, see DevInfo configuration service provider.

```
<SyncHdr>
   <VerDTD>1.2</VerDTD>
   <VerProto>DM/1.2</VerProto>
   <SessionID>1</SessionID>
   <MsgID>1</MsgID>
   <Target>
      <LocURI>{unique device ID}</LocURI>
   </Target>
   <Source>
      <LocURI>https://www.thephone-company.com/mgmt-server</LocURI>
   </Source>
</SyncHdr>
```

SyncBody element

SyncBody contains one or more DM commands.

Be aware that **SyncBody** can contain multiple DM commands; each command must have a different CmdID value.

The following example shows the body component of a DM message. In this example, **SyncBody** contains only one command, Get. This is indicated by the <Final /> tag that occurs immediately after the terminating tag for the Get command.

© 2014 Microsoft. All rights reserved.

```
<SyncBody>
   <!-- query device OS software version -->
   <Get>
      <CmdID>2</CmdID>
      <Item>
         <Target>
            <LocURI>./DevDetail/SwV</LocURI>
         </Target>
      </Item>
   </Get>
   <Final />
</SyncBody>
```

Note: When using SyncML for OMA DM provisioning, a LocURI in **SyncBody** can have a "." as a valid segment name only in the first segment. However, a "." is not a valid segment name for the other segments. For example, the following LocURI is not valid because the segment name of the seventh segment is a ".".

```
<LocURI>./cimv2/MDM_ConfigSetting.SettingName</LocURI>
```

Update device settings example

The **Replace** command is used to update a device setting.

Code example

The following example shows how to use the **Replace** command to update a device setting.

```
<SyncHdr>
   <VerDTD>1.2</VerDTD>
   <VerProto>DM/1.2</VerProto>
   <SessionID>1</SessionID>
   <MsgID>1</MsgID>
   <Target>
      <LocURI>{unique device ID}</LocURI>
   </Target>
   <Source>
      <LocURI>https://www.thephone-company.com/mgmt-server</LocURI>
   </Source>
</SyncHdr>
<SyncBody>
   <!-- update device setting -->
   <Replace>
      <CmdID>2</CmdID>
      <Item>
         <Target>
            <LocURI>./cimv2/MDM_ConfigSetting/MDM_ConfigSetting.
              SettingName=%22UnEnrollOnCertExpiry%22/SettingValue</LocURI>
         </Target>
         <Meta>
            <Type xmlns="syncml:metinf">text/plain</Type>
            <Format xmlns="syncml:metinf">bool</Format>
         </Meta>
         <Data>true</Data>
      </Item>
   </Replace>
   <Final />
</SyncBody>
```

© 2014 Microsoft. All rights reserved.

Server requirements for OMA DM

The following are the general server requirements for using OMA DM to manage Windows-based devices:

- The OMA DM server must support the OMA DM v1.2 protocol.
- Secure Sockets Layer (SSL) must be on the OMA DM server, and it must provide server certificate-based authentication, data integrity checking, and data encryption. If the certificate is not issued by a commercial certification authority whose root certificate is preinstalled in the device, you must provision the company's root certificate in the device's ROOT store.
- To authenticate the client, you must use client certificate-based authentication.

Enterprise OMA DM supported configuration service providers

The following configuration service providers are supported by using OMA DM:

- DevInfo configuration service provider

Enterprise OMA DM supported remote WMI classes

The following remote WMI classes are supported by using OMA DM:

- Browser Management
  - MDM_BrowserSecurityZones
  - MDM_BrowserSettings
- Device Configuration
  - MDM_ConfigSetting
  - MDM_MgmtAuthority
  - MDM_DeviceRegistrationInfo
  - MDM_SecurityStatus
- Device Inventory
  - Win32_Battery
  - Win32_BIOS
  - Win32_ComputerSystem
  - Win32_ComputerSystemProduct
  - Win32_CurrentTime
  - Win32_DesktopMonitor
  - Win32_DiskDrive
  - Win32_DisplayConfiguration
  - Win32_InfraredDevice
  - Win32_LocalTime
  - Win32_LogicalDisk
  - Win32_NetworkAdapter
  - Win32_NetworkAdapterConfiguration

© 2014 Microsoft. All rights reserved.

    - Win32_OperatingSystem
    - Win32_PhysicalMemory
    - Win32_PnPDevice
    - Win32_PortableBattery
    - Win32_Processor
    - Win32_QuickFixEngineering
    - Win32_Service
    - Win32_Share
    - Win32_SystemBIOS
    - Win32_SystemEnclosure
    - Win32_TimeZone
    - Win32_UTCTime
    - Win32_WindowsUpdateAgentVersion

- Device Restrictions
    - MDM_Restrictions
    - MDM_RestrictionsUser
- Data Protection
    - MDM_Client
- Password Policy
    - MDM_EASPolicy
- Application Management
    - MDM_Application
    - MDM_ApplicationFramework
    - MDM_AppInstallJob
    - MDM_ApplicationSetting
    - MDM_Sideloader
    - MDM_RemoteApplication
    - MDM_RemoteAppUserCookie
    - MDM_WebApplication
- Resource Access
    - MDM_Certificate
    - MDM_CertificateEnrollment
    - MDM_WirelessProfile
    - MDM_WirelessProfileXml
    - MSFT_VpnConnection
    - MDM_VpnApplicationTrigger
- Push Notification
    - MDM_WNSConfiguration
    - MDM_WNSChannel

© 2014 Microsoft. All rights reserved.

OMA DM Mode

The OMA DM agent can execute under different contexts on the device, as well as varying conditions (both internal and external) that caused the agent to launch. The OMA DM agent forwards the context and condition to the DM service by using the "mode" parameter in the request URL. The mode parameter will contain one of the following three values:

- Maintenance
- Apps
- Machine

If the mode parameter is set to **Maintenance** the OMA-DM agent was launched in the maintenance window by the Windows Task Scheduler. The DM Service can issue ALL commands to the device in this mode.

If the mode parameter is set to **Apps** the OMA-DM agent was launched because the DM Service sent a push notification to the device. The DM Service can issue ALL commands to the device in this mode but may want to process real time actions first such as **remote un-enroll** or **remote device lock**.

If the mode parameter is set to **Machine**, the OMA-DM agent was launched in the System context and will not have access to managed user's profile. The DM Service should process the OMA-DM checkin but must not send any commands to the device except for device inventory, ResetPassword or SendUnEnrollRequest while in this mode

© 2014 Microsoft. All rights reserved.

DM session initialization

After the device has successfully enrolled into management, the OMA-DM agent process is immediately executed on the device and will send an OMA-DM SyncML session init message to the configured management endpoint. The following is an example of the SyncML message.

```
<SyncML xmlns="SYNCML:SYNCML1.2">
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.2</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>http://localhost:8000/handler.ashx</LocURI>
    </Target>
    <Source>
      <LocURI>e49e0231-67bf-4161-b69f-cb5928f63bff</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
    <Alert>
      <CmdID>2</CmdID>
      <Data>0</Data>
    </Alert>
    <Replace>
      <CmdID>3</CmdID>
      <Item>
        <Source>
          <LocURI>./DevInfo/DevId</LocURI>
        </Source>
        <Data>37565048</Data>
      </Item>
      <Item>
        <Source>
          <LocURI>./DevInfo/Man</LocURI>
        </Source>
        <Data>Microsoft Corporation</Data>
      </Item>
      <Item>
        <Source>
          <LocURI>./DevInfo/Mod</LocURI>
        </Source>
        <Data>Microsoft Windows NT Workstation 6.3</Data>
      </Item>
      <Item>
        <Source>
          <LocURI>./DevInfo/DmV</LocURI>
        </Source>
        <Data>1.2</Data>
      </Item>
      <Item>
        <Source>
          <LocURI>./DevInfo/Lang</LocURI>
        </Source>
        <Data>en-US</Data>
      </Item>
    </Replace>
    <Final/>
  </SyncBody>
</SyncML>
```

© 2014 Microsoft. All rights reserved.

The server can determine a client is attempting to establish a session by keying off the **Alert Command** in the **SyncBody**. The **Data** element of the **Alert Command** will contain a 0. This value is not compliant with a standard OMA-DM Alert Command where valid values for client initiated sessions are: 1201 and 1202. The session init message will also contain five **Replace Commands** with **DevInfo** properties. The properties are:

- ./DevInfo/DevId

  The DevId is unknown or invalid until the server sends the DevId to the client using a **Replace** command. The server should always identify the client by the subject name in the device identity certificate which is included in all HTTPS POST commands from the client to the server.

- ./DevInfo/Man

  The Man property is the manufacturer of the device. The OMA-DM agent will always set this property to **Microsoft Corporation**.

- ./DevInfo/Mod

  The Mod property is the model of the device. The OMA-DM agent will set this property to the Operating System version that is running on the device. In this example Mod is set to "`Microsoft Windows NT Workstation 6.2`".

- ./DevInfo/DmV

  The DmV property is the device management protocol version the agent supports. In this example DmV is set to **1.2**. The OMA-DM agent will always send a protocol version with a minimum value of **1.2**.

- ./DevInfo/Lang

  The Lang property is the default language on the device. In this example Lang is set to "en-us" – United States English.

© 2014 Microsoft. All rights reserved.

The Server must respond acknowledging the session init message. The following is an example of such a response:

```
<SyncML xmlns="SYNCML:SYNCML1.2">
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.2</VerProto>
    <SessionID>1</SessionID>
    <MsgID>1</MsgID>
    <Target>
      <LocURI>e49e0231-67bf-4161-b69f-cb5928f63bff</LocURI>
    </Target>
    <Source>
      <LocURI>http://localhost:8000/handler.ashx</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
    <Status>
      <CmdID>1</CmdID>
      <MsgRef>1</MsgRef>
      <CmdRef>0</CmdRef>
      <Cmd>SyncHdr</Cmd>
      <Data>200</Data>
    </Status>
    <Status>
      <CmdID>2</CmdID>
      <MsgRef>1</MsgRef>
      <CmdRef>2</CmdRef>
      <Cmd>Alert</Cmd>
      <Data>200</Data>
    </Status>
    <Status>
      <CmdID>3</CmdID>
      <MsgRef>1</MsgRef>
      <CmdRef>3</CmdRef>
      <Cmd>Replace</Cmd>
      <Data>200</Data>
    </Status>
    <Get>
      <CmdID>6</CmdID>
      <Item>
        <Target>
          <LocURI>./cimv2/MDM_Client</LocURI>
        </Target>
      </Item>
    </Get>
    <Final />
  </SyncBody>
</SyncML>
```

The server sent back **Status Commands** for the **SyncHdr** and for the **Alert** and **Replace Commands**. The server could shorten the message by only sending a **Status Command** back for the **SyncHdr**. In this example the server is also sending a Get Command to the client asking for an addressable instance of the MDM_Client class. If the server does not include any Commands for the client in the message, the session is terminated.

© 2014 Microsoft. All rights reserved.

The client will respond to the Get Command on the MDM_Client class with **Status** and **Results** commands similar to the following.

```
<Status>
  <CmdID>2/CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>6</CmdRef>
  <Cmd>Get</Cmd>
  <Data>200</Data>
</Status>
<Results>
  <CmdID>3</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>6</CmdRef>
  <Item>
    <Source>
      <LocURI>./cimv2/MDM_Client</LocURI>
    </Source>
    <Meta>
      <Format xmlns="syncml:metinf">node</Format>
    </Meta>
    <Data>MDM_Client.DeviceID="e49e0231-67bf-4161-b69f-cb5928f63bff"</Data>
  </Item>
</Results>
```

© 2014 Microsoft. All rights reserved.

DM device inventory

The DM service can query the device for inventory during any DM session, but the recommended workflow is for the service to query the device for full inventory the first time that the device checks in after enrolling. When the service has persisted full inventory for the device, inventory should only be queried occasionally for possible updates on the device, such as additional memory through an SD card.

The OMA-DM agent has an approved list of Windows remote WMI classes for inventory. This approved list can only be targeted in SyncML Get commands. Put, Replace and Exec are not supported on these classes. The following is the exhaustive list of available classes together with their MSDN reference:

[MSDN-Win32_Battery] Microsoft Corporation, "Win32_Battery class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394074(v=vs.85).aspx

[MSDN-Win32_BIOS] Microsoft Corporation, "Win32_BIOS class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394077(v=vs.85).aspx

[MSDN-Win32_CmptrSys] Microsoft Corporation, "Win32_ComputerSystem class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394102(v=vs.85).aspx

[MSDN-Win32_CmptrSysProd] Microsoft Corporation, "Win32_ComputerSystemProduct class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394105(v=vs.85).aspx

[MSDN-Win32_CurrentTime] Microsoft Corporation, "Win32_CurrentTime class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394114(v=vs.85).aspx

[MSDN-Win32_DiskDrive] Microsoft Corporation, "Win32_DiskDrive class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394132(v=vs.85).aspx

[MSDN-Win32_DisplayCnfg] Microsoft Corporation, "Win32_DisplayConfiguration class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394137(v=vs.85).aspx

[MSDN-Win32_DsktpMntr] Microsoft Corporation, "Win32_DesktopMonitor class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394122(v=vs.85).aspx

[MSDN-Win32_InfrdDevice] Microsoft Corporation, "Win32_InfraredDevice class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394158(v=vs.85).aspx

[MSDN-Win32_LocalTime] Microsoft Corporation, "Win32_LocalTime class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394171(v=vs.85).aspx

[MSDN-Win32_LogicalDisk] Microsoft Corporation, "Win32_LogicalDisk class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394173(v=vs.85).aspx

[MSDN-Win32_NtwkAdptr] Microsoft Corporation, "Win32_NetworkAdapter class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394216(v=vs.85).aspx

© 2014 Microsoft. All rights reserved.

[MSDN-Win32_NtwkAdptrCnfg] Microsoft Corporation, "Win32_NetworkAdapterConfiguration class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394217(v=vs.85).aspx

[MSDN-Win32_OpSys] Microsoft Corporation, "Win32_OperatingSystem class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394239(v=vs.85).aspx

[MSDN-Win32_PhysMemory] Microsoft Corporation, "Win32_PhysicalMemory class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394347(v=vs.85).aspx

[MSDN-Win32_PnPDevice] Microsoft Corporation, "Win32_PnPDevice class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394352(v=vs.85).aspx

[MSDN-Win32_PortBattery] Microsoft Corporation, "Win32_PortableBattery class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394357(v=vs.85).aspx

[MSDN-Win32_Processor] Microsoft Corporation, "Win32_Processor class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394373(v=vs.85).aspx

[MSDN-Win32_QFE] Microsoft Corporation, "Win32_QuickFixEngineering class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394391(v=vs.85).aspx

[MSDN-Win32_Service] Microsoft Corporation, "Win32_Service class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394418(v=vs.85).aspx

[MSDN-Win32_Share] Microsoft Corporation, "Win32_Share class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394435(v=vs.85).aspx

[MSDN-Win32_SysBIOS] Microsoft Corporation, "Win32_SystemBIOS class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394467(v=vs.85).aspx

[MSDN-Win32_SysEnclose] Microsoft Corporation, "Win32_SystemEnclosure class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394474(v=vs.85).aspx

[MSDN-Win32_TimeZone] Microsoft Corporation, "Win32_TimeZone class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394498(v=vs.85).aspx

MSDN-Win32_UTCTime] Microsoft Corporation, "Win32_UTCTime class," http://msdn.microsoft.com/en-us/library/windows/desktop/aa394510(v=vs.85).aspx

© 2014 Microsoft. All rights reserved.

In the following example, the Server is sending a **Get** Command asking the client for the default processor on the device.

```
<SyncML xmlns="SYNCML:SYNCML1.2">
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.2</VerProto>
    <SessionID>1</SessionID>
    <MsgID>3</MsgID>
    <Target>
      <LocURI>e49e0231-67bf-4161-b69f-cb5928f63bff</LocURI>
    </Target>
    <Source>
      <LocURI>http://localhost:8000/handler.ashx</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
    <Status>
      <CmdID>1</CmdID>
      <MsgRef>1</MsgRef>
      <CmdRef>0</CmdRef>
      <Cmd>SyncHdr</Cmd>
      <Data>200</Data>
    </Status>
    <Get>
      <CmdID>2</CmdID>
      <Item>
        <Target>
          <LocURI>./cimv2/Win32_Processor</LocURI>
        </Target>
      </Item>
    </Get>
    <Final />
  </SyncBody>
</SyncML>
```

© 2014 Microsoft. All rights reserved.

The client responds with the following SyncML.

```
<SyncML xmlns="SYNCML:SYNCML1.2">
  <SyncHdr>
    <VerDTD>1.2</VerDTD>
    <VerProto>DM/1.2</VerProto>
    <SessionID>1</SessionID>
    <MsgID>3</MsgID>
    <Target>
      <LocURI>http://localhost:8000/handler.ashx</LocURI>
    </Target>
    <Source>
      <LocURI>e49e0231-67bf-4161-b69f-cb5928f63bff</LocURI>
    </Source>
  </SyncHdr>
  <SyncBody>
    <Status>
      <CmdID>1</CmdID>
      <MsgRef>3</MsgRef>
      <CmdRef>1</CmdRef>
      <Cmd>Get</Cmd>
      <Data>200</Data>
    </Status>
    <Results>
      <CmdID>2</CmdID>
      <MsgRef>3</MsgRef>
      <CmdRef>1</CmdRef>
      <Item>
        <Source>
          <LocURI>./cimv2/Win32_Processor</LocURI>
        </Source>
        <Meta>
          <Format xmlns="syncml:metinf">node</Format>
        </Meta>
        <Data>Win32_Processor.DeviceID="CPU0"</Data>
      </Item>
    </Results>
    <Final />
  </SyncBody>
</SyncML>
```

The **DeviceID** property in the Win32_Processor class is a Key qualifier which can then be used in subsequent **Get** Commands to retrieve properties of the Win32_Processor class.

© 2014 Microsoft. All rights reserved.

For example the following **Get** Commands would retrieve the data width and clock speed of the processor.

```
<Get>
  <CmdID>2</CmdID>
  <Item>
    <Target>
      <LocURI>./cimv2/Win32_Processor/Win32_Processor.DeviceID=%22CPU0%22/DataWidth</LocURI>
    </Target>
  </Item>
</Get>
<Get>
  <CmdID>2</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/Win32_Processor/Win32_Processor.DeviceID=%22CPU0%22/CurrentClockSpeed
      </LocURI>
    </Target>
  </Item>
</Get>
```

© 2014 Microsoft. All rights reserved.

DM agent configuration

When a device is enrolled into management, OMA-DM agent bootstrap and configuration properties are returned in the wap-provisioning document. The OMA-DM agent provides capabilities to read and modify some of those properties through the management protocol. When the DM service modifies a property, the change doesn't take affect until the next time that the OMA-DM agent runs either from a push notification or during the maintenance window.

Managing the configuration settings is performed through the remote WMI class MDM_ConfigSettings.

```
[Description("This class provides capability to get and set configuration settings for the
oma-dm agent")]
class MDM_ConfigSetting
{
  [Key,Description("The key to identify a setting by name")]
  string SettingName;

  [Description("This property contains a setting value associated to a SettingName")]
  string SettingValue;
};
```

To query a setting to get its current value the DM service will issue a SyncML Get command targeting the setting's name. The following partial SyncML command demonstrates **retrieving the unenroll the agent on certificate expiration** setting.

```
<Get>
  <CmdID>18</CmdID>
  <Item>
    <Target>
      <LocURI>
         ./cimv2/MDM_ConfigSetting/MDM_ConfigSetting.
SettingName=%22UnEnrollOnCertExpiry%22/SettingValue
      </LocURI>
    </Target>
  </Item>
</Get>
```

The OMA-DM agent will return a SyncML message that contains **Status** and **Results** for the previous **Get** command which would resemble the following partial SyncML.

```
<Status>
  <CmdID>2</CmdID>
  <CmdRef>18</CmdRef>
  <Cmd>Get</Cmd>
  <Data>200</Data>
</Status>
<Results>
  <CmdID>3</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>18</CmdRef>
  <Item>
    <Source>
      <LocURI>
./cimv2/MDM_ConfigSetting/MDM_ConfigSetting.
SettingName=%22UnEnrollOnCertExpiry%22/SettingValue
```

© 2014 Microsoft. All rights reserved.

```
      </LocURI>
    </Source>
    <Data>true</Data>
  </Item>
</Results>
```

The DM service can update existing agent settings by issuing a SyncML **Replace** command again targeting the setting's name. The following partial SyncML command demonstrates setting the **unenroll the agent on certificate** expiration to **true**.

```
<Replace>
  <CmdID>19</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_ConfigSetting/MDM_ConfigSetting.
SettingName=%22UnEnrollOnCertExpiry%22/SettingValue
      </LocURI>
    </Target>
    <Data>true</Data>
  </Item>
</Replace>
```

The following table lists the agent settings that can be queried and modified.

Setting Name	Commands	Data Type	Description
UnEnrollOnCertExpiry	Get/Replace	true/false	If true, the agent will un-enroll itself from management when the device certificate has expired.
EmailAddress	Get	string	The UPN of the user who enrolled the device. Examples: joe@contoso.com; CONTOSO\joe
RoboEnabled	Get/Replace	true/false	If true, the agent will support a ROBO request for the device certificate.
BitsRetryDelay	Get/Replace	int	The number of seconds the agent will wait before retrying a failed Bits request. **Note:** The Bits protocol is used to download applications from the DM service. Supporting Bits is not a requirement. The agent will fall back to standard HTTP.
BitsNoProgressTimeout	Get/Replace	int	The total number of seconds the agent will attempt a Bits request before failing.
AuthorityName	Get		The name of the management service the device enrolled into.

© 2014 Microsoft. All rights reserved.

DeviceClientId	Get/Replace	string	The device client ID assigned by the management service.
EnrolledUserSID	Get	string	The local SID for the user who enrolled the device.
Setting Name	Commands	Data Type	Description
BootstrapMP	Get	string	The URL of the primary management endpoint configured during enrollment.
EnrollmentServer	Get	string	The enrollment server host the device enrolled against.
EnterpriseIDs	Get/Replace	string	A "|" delimited list of enterprise domains the OMA-DM will call during unenrollment to delete associated encryption keys. EFS enlightened application such as the built-in mail client create storage containers encrypted with a known domain id. The IT manager is responsible for understanding the domains used in their company. The DM service must provide a mechanism in the console for the IT manager to add the appropriate domains which the DM service will then send to the OMA-DM agent to help with "Selective Wipe".
MachineMDMMode	Get	int	The return value will be 1 if the device was EOBO (enroll on behalf of) enrolled.

© 2014 Microsoft. All rights reserved.

DM push notification

After the device is successfully enrolled, the OMA-DM agent is configured to run during the Windows maintenance task which is typically configured to run one time per day. To force the OMA-DM agent to run on demand, the DM service can send a push notification to the device. Along with adding the OMA-DM agent to the Windows maintenance task, the enrollment client also creates another Windows task called "MDMAppInstallationTask". This task is created by using a custom trigger and does not run on a preconfigured schedule. The "MDMAppInstallationTask" executes the OMA-DM agent with the "-App" command line parameter. The name of the task and the command line parameter "-App" are artifacts from Windows 8 RT when the only action that required push notification was user initiated application installs. When the OMA-DM agent calls the client WNS framework to register for push notifications, it also passes the "MDMAppInstallationTask" as the task it wants run when the device receives a push notification targeted to the OMA-DM agent.

The OMA-DM agent takes advantage of the Windows Notification Service for push notification support.

The Windows Notification Service (WNS) requires three pieces of information to send push notifications:

- Application ID
- Package Family Name (PFN)
- Client Secret

The Application ID is a unique name given to the application by the developer. The Package Family Name or PFN is derived by appending the Application ID, application version, application architecture and a hash of the publisher's CN. The Client Secret is generated by the Windows Store when an application is created in the Live Connect Developer Center.

Because push notification on Windows 8x was initially designed for modern Windows-based applications only, MDM vendors must follow an ad hoc process to generate the ApplicationID, Package Family Name and Client Secret by creating a sample, or placeholder modern application that will never be submitted to the store.

The steps to generate the push notification credentials from a sample, or placeholder application are:

1. In Visual Studio, use the Blank App template to create a Windows Store application.
2. Give the application a name associated to your MDM Service, such as awesomemdm.contoso.com.
3. Right-click the Package.appxmanifest file in your solution, select packaging. Notice that a PFN has been generated for the application as well as a Publisher that is associated to your developer account.
4. From a browser window, enter the URL appdev.microsoft.com.

© 2014 Microsoft. All rights reserved.

5. From the Dashboard, select Submit your application.
6. The next step is to give your application a name. Use the same application name as you used in Visual Studio.
7. From the application page, select Services.
8. Select Live Services Site. Notice that a Client Secret, Package SID, and Application identity have been created for your application.
9. Now go back to your solution in Visual Studio and open the Package.appxmanifest file in a standard XML or Text Editor. Change the Identity attributes **Name** and **Publisher** to the values created in the Live Services Site. Save and close the file.
10. When you go back to the Visual Studio solution and double-click the Package.appxmanifiest file, then select Packaging you will notice a new PFN has been generated for the application.

From this sample or placeholder modern application you now have three pieces of information required by WNS to send push notifications. You MUST never submit the sample or placeholder modern application to the Windows store or delete the application from the developer portal.

The DM Service must send the AppID and PFN to the OMA-DM agent which the OMA-DM agent in turn registers for push notification with WNS. The result of the internal WNS registration is a unique channel URI which the DM Service must retrieve from the OMA-DM agent. The unique channel URI and the client secret are used to POST or send notifications to the Windows Notification Service (WNS). To be specific, the client secret is necessary to set up an authenticated channel between the DM service and the Windows Notification Service.

Managing the WNS channel is performed through two remote WMI classes called MDM_WNSConfiguration and MDM_WNSChannel respectively. MDM_WNSConfiguration is used to configure the AppId and PFN in the agent. When the agent is configured with the AppId and PFN, the agent will attempt to request a channel with WNS. MDM_WNSChannel is used by the DM service to retrieve channel URI information.

```
[Description("This class provides information on the WNS channel used for notifications to the
MDM agent")]
class MDM_WNSChannel
{
  [key,
   Description("The identity of the MDM application")]
  string AppId;
  [Description("The SID of the enrolled user")]
  string UserSID;
  [Description("This is the channel created for WNS notifications for the user on this
device")]
  string Channel;
  [Description("The expiration time for the WNS channel")]
  datetime ExpirationTime;
  [Description("WNS Channel Creation Status"),ValueMap{"0", "1"}]
  uint32 ChannelStatus;
  [Description("This is the error code in the event that WNS channel creation fails")]
  uint32 LastError;
};
```

© 2014 Microsoft. All rights reserved.

```
[Description("This class provides the capability to configure WNS notifications for the MDM
agent")]
class MDM_WNSConfiguration
{
  [key,
  Description("The identity of the MDM application")]
 string AppId;
  [Description("ConfigurationStatus is 0 if not configured and 1 if configured")]
  uint32 ConfigurationStatus;
  [static, Description("Updates WNS Configuration")]
  uint32 UpdateConfiguration([In,Description("ConfigString is in the format <AppId>;<WNS
Package Family Name>")] string ConfigString);
}
```

The DM service should first query the OMA-DM agent to see if an existing unique channel URI has already been established on the device. The following partial SyncML message demonstrates how that is done.

```
<Get>
  <CmdID>2</CmdID>
  <Item>
    <Target>
      <LocURI>
     ./cimv2/MDM_WNSConfiguration/MDM_WNSConfiguration.AppId=%22{AppId}%22/ConfigurationStatus
      </LocURI>
    </Target>
  </Item>
</Get>
<Get>
  <CmdID>3</CmdID>
  <Item>
    <Target>
```

© 2014 Microsoft. All rights reserved.

```
      <LocURI>
        ./cimv2/MDM_WNSChannel/MDM_WNSChannel.AppId=%22{AppId}%22/Channel
      </LocURI>
    </Target>
  </Item>
</Get>
<Get>
  <CmdID>4</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_WNSChannel/MDM_WNSChannel.AppId=%22{AppId}%22/ChannelStatus
      </LocURI>
    </Target>
  </Item>
</Get>
<Get>
  <CmdID>5</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_WNSChannel/MDM_WNSChannel.AppId=%22{AppId}%22/ExpirationTime
      </LocURI>
    </Target>
  </Item>
</Get>
```

If the MDM_WNSConfiguration.ConfigurationStatus equals **1** and MDM_WNSChannel.ChannelStatus equals 1 and MDM_WNSChannel.ExpirationTime has not expired, the DM service can use MDM_WNSChannel.Channel for the unique channel URI and send notifications to the device. However if the previous is not true, the DM service should update the WNS configuration.

If the DM service updates MDM_WNSConfiguration, it invalidates MDM_WNSChannel. The DM service should only update MDM_WNSConfiguration:

- When MDM_WNSConfiguration.ConfigurationStatus does not equal **1**, or
- If the PFN changes.

The following partial SyncML message shows how to send the AppId and PFN to the OMA-DM agent. Note: The PFN must be lowercased before sending it to the OMA-DM agent.

© 2014 Microsoft. All rights reserved.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MDM_WNSConfiguration/MDM_WNSConfiguration.AppId=%22{AppId}%22/Exec=UpdateConfiguration
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>ConfigString={AppId};{PFN}</Data>
  </Item>
</Exec>
```

The Data element will resemble the following:

```
<Data>
ConfigString=s-1-15-2-3219717196-666686559-712272758-1360859528-513151160-3699258166-
1325108402;15494windowsstorewns.wnsmdmpartner_skcpvdt8tnyse
</Data>
```

The OMA-DM agent will asynchronously attempt to establish a channel with WNS when it receives the UpdateConfiguration request. If the DM service immediately requests the unique channel URI, it may not be established yet.

To send a push notification to the device, the DM service must first authenticate with WNS using the AppId and client secret. Once authenticated, the DM service will receive a token that it can use to initiate a raw push notification for any channel URI. When the DM service wishes to initiate a device management session with a device, it can use its token and the device's channel URI and start to send notifications to the device.

The DM service must send raw push notifications only. The payload of the raw notification can be blank.

Because a device may currently not be connected to the WNS cloud, it is possible to configure the raw notification request to get status information back from the WNS cloud. The DM service can receive the connection status when it sends a push notification using the device's channel URI with the X-WNS-RequestForStatus header. This instructs WNS to return a status the service can use to determine whether the device is connected to WNS. This can also be used by the DM service to determine whether a push notification has reached the device. Additionally, if the service wishes to send a time-bound raw push notification, the service can use the X-WNS-TTL header that will provide WNS with a Time to Live binding so that the notification will expire after the time has passed. For more information on Push notification service request and response header, see this article on MSDN: http://msdn.microsoft.com/en-us/library/windows/apps/hh465435.aspx.

© 2014 Microsoft. All rights reserved.

If the customer has locked down their on-premises network, they may have to explicitly allow connections to the WNS service. The following FQDNs must be available:

- *.wns.windows.com
- *.notify.windows.com

For more information and sample code to build the server-side components that initiate raw push notifications, please see these MSDN articles:

- Push notification overiew: http://msdn.microsoft.com/library/windows/apps/hh913756.aspx
- Sending a push notification: http://msdn.microsoft.com/library/windows/apps/hh868252.aspx

Please note the following restrictions as related to push notifications and WNS:

- All push notifications are delivered by best effort. The notification is not guaranteed to be delivered to the device.
- Push for device management uses raw push notifications. This means these raw push notifications do not support or use push notification payloads.
- Each channel URI has a limit of 150 push notifications per hour
- Receipt of push notifications are sensitive to the Battery Sense and Data Sense settings on the device. For example, if the battery drops below certain thresholds, the device's persistent connection with WNS will be terminated. Additionally, if the user is using Data Sense and has exceeded their monthly allotment of data, the device's persistent connection with WNS will also be terminated.
- A channel URI provided to the management service by the device is only valid for 30 days, and can be revoked before the lapse of the 30 days. The device will automatically renew the channel URI after 15 days and trigger a management session on successful renewal of the channel URI. We strongly recommend that during every management session, the management service queries the channel URI value to ensure that it has received the latest value. This ensures that the management service will not attempt to use a channel URI that has expired.
- Push notification does not replace running the OMA-DM agent in the maintenance task.
- PFNs can be revoked by WNS if improper use of PFNs regarding push notification is detected. Any devices under management using the flagged PFN will no longer have push initiated device management support.

Please be aware that DM push notification does not support configuration using WAP Provisioning XML and cannot be configured during an enrollment session. Configuration of push notification can only occur during a management session.

Note that in push notification with enrollment-on-behalf-of (EOBO), make sure that the admin account is logged out and only the managed user is logged in, otherwise it will fail.

© 2014 Microsoft. All rights reserved.

DM password policy

Windows 8.1 supports three types of user accounts on a device: Local Accounts, Microsoft Accounts and Domain Joined Accounts. The OMA-DM agent is disabled when a device is domain joined which means the management service cannot set password policy for a domain joined account. Windows 8.1 RT and Windows 8.1 Home SKUs cannot be domain joined.

The OMA-DM agent supports setting password policy for the managed user on the device. Some password policy properties set the policy for the device which means all users configured on the device will be subject to the policy.

The following table lists valid ranges for each password policy property:

Password policy	Local account	Microsoft account
Minimum Password Length	0 – 14 characters	0 – 8 characters
Password Complexity	3 character groups	2 character groups
Password History	0 – 24 passwords	N/A
Password Expiration	0 – 999 Days	N/A
Idle time until lock	1 – 120 minute(s)	1 – 120 minute(s)
Failed password attempts before wipe	4 – 16	4 – 16
Disallow convenience logon	Yes	Yes

Password complexity groups are defined as:

- lowercase alphabetical characters
- uppercase alphabetical characters
- numbers
- nonalphanumeric characters

The current password policy management has some known caveats:

1. Setting a more restrictive password policy on the device does not immediately force the user into a change password state. The next time that a user logs on to the device they are asked to change their password. The new password must be compliant with the password policy.
2. Once **DisallowConvenienceLogon** is set on a device, it can't be turned back off through OMA-DM management.
3. The DM service cannot query the current password policy on the user or device.

© 2014 Microsoft. All rights reserved.

4. Windows 8.1 does not support an unassisted device wipe. If the user does not enter the correct password beyond the "failed password attempts" policy, the device will reboot.
5. Setting password complexity on a Local Account will set a three-group complexity for all local accounts on the device together with the following restrictions:
6. The password can't contain the user's account name or parts of the user's full name that exceed two consecutive characters
7. The password must be at least six characters long
8. Local account password policy does not allow setting a less restrictive policy. For example, if the password minimum length is first set to 8 the DM service can't change the minimum length to six.
9. The only supported value for setting MinPasswordComplexCharacters is 1. MDMAgent.exe will reject any other value and return a 404 error.

The DM service targets the remote WMI class MDM_EASPolicy to set password policy.

```
// MinPasswordLength = "0".."14" local accounts  "0" .. "8" connected accounts
// DisallowConvenienceLogon = true
// AutolockTimeout = "1m".."120m"
// MaxHistory = "1"..”24” supported on local accounts only
// Expiration = "1d"..”999d” supported on local accounts only
// MaxAttemptsBeforeWipe = "4".."16"
// MinPasswordComplexCharacters = "1" character groups
// character groups
//   lowercase alphabetical characters
//   uppercase alphabetical characters
//   numbers
//   nonalphanumeric characters
// Ex. character group = 3 complexity = "lowercase alphabetical" and "uppercase alphabetical"
//       and "numbers"
//
// Ex. NamedValuesList=MinPasswordLength,8;AutolockTimeout,5
[Description("This class provides EAS Password Policy configuration on the device")]
class MDM_EASPolicy
{
    [Key,Description("The key to identify the instance of MDM_EASPolicy class")]
    Uint32  Key;
    [static, Description("Set password related settings values")]
    uint32 SetValues([In,Description("Settings name value pairs in formt :
<name1>,<value1>;<name2>,<value2>;...")] string NamedValuesList);
};
```

© 2014 Microsoft. All rights reserved.

The following partial SyncML example demonstrates setting a minimum password length and auto-lock time-out.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_EASPolicy/MDM_EASPolicy.Key=%221%22/Exec=SetValues
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>NamedValuesList=MinPasswordLength,8;AutolockTimeout,5</Data>
  </Item>
</Exec>
```

© 2014 Microsoft. All rights reserved.

DM application management

An DM service can install applications, query previously installed applications and delete previously installed applications on the device. Applications are installed for the managed user only.

Two types of applications are supported:

1) Modern Windows applications
2) Weblinks (for example, software as a service [SaaS] applications)

Modern Windows applications

Before a modern Windows application can be installed on a device through the OMA-DM agent, the DM service must enable application sideloading on the device and install an application sideloading key. Sideloading keys are required to install line of business applications on Windows 8.1 RT, Windows 8.1 Home, and Windows 8.1 Professional SKUs. The IT manager must purchase sideloading keys or acquire them if they have a volume purchase license with Microsoft. The DM service must provide a mechanism to upload the keys so they can be distributed on devices that need access to the enterprise modern applications.

The DM service will target the remote WMI class MDM_SideLoader to install sideloading keys, install the code signing certificate used to sign the enterprise modern applications and un-activate sideloading of enterprise modern applications. Note: activating the key also enables sideloading on the device. un-activate also removes a previously installed sideloading key.

```
[Description("This class provides methods for activating sideloading of LOB applications")]
class MDM_SideLoader
{
  [key, Description("The key to identify the instance of MDM_SideLoader class.")]
  uint32 key;
  [Description("This property provides a hash of the installed product key.")]
  string ProductKeyHash;
  [static,Description("A method for activating sideloading and installing the product key")]
  uint32 ActivateKey([In,Description("Sideloading key")] string ProductKey);
  [static,Description("A method for adding the package signing certificate")]
  uint32 AddCertificate([In,Description("Certificate In Base-64 encoded format")] string
CertificateBlob);
  [static,Description("A method to disable sideloading LOB applications")]
  uint32 UnActivateLOB();
};
```

© 2014 Microsoft. All rights reserved.

The following partial SyncML example demonstrates activating sideloading and installing a sideloading key.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_SideLoader/MDM_SideLoader.key=%221%22/Exec=ActivateKey
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>ProductKey={sideloading key}</Data>
  </Item>
</Exec>
```

The DM service can check if a sideloading key is already installed on the device by issuing the following partial SyncML Get command.

```
<Get>
  <CmdID>10</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_SideLoader/MDM_SideLoader.Key=%221%22/ProductKeyHash
      </LocURI>
    </Target>
  </Item>
</Get>
```

All modern Windows applications must be signed in order to install on a device. Enterprise modern Windows applications are normally in house developed line of business (LOB) applications. In house developers design, code and test the applications before turning them over to the enterprise's IT manager responsible for application management. The IT manager is responsible for signing those applications with a code signing certificate. A DM service could choose to sign the application packages and bundles in the service itself. The signed application is validated against the code signing certificate before it is allowed to install on the device. This means the code signing certificate must also be installed on all devices where the applications will be installed.

The following partial SyncML example demonstrates installing a code signing certificate.

© 2014 Microsoft. All rights reserved.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_SideLoader/MDM_SideLoader.key=%221%22/Exec=AddCertificate
      </LocURI>
    </Target>
    <Data>CertificateBlob={base64 encoded certificate}</Data>
  </Item>
</Exec>
```

Normally, installing the sideloading key and installing the code signing certificate is done once per device.

The DM service will target the remote WMI class MDM_AppInstallJob to install a modern application on a device.

```
[Description("This class provides the ability to create a job for installing or un-installing
an application and also provides the job status and error code if any")]
class MDM_AppInstallJob
{
  [Key,Description("Unique Id for Application install/un-install")]
  string JobID;
  [Description("Package full name")]
  string PackageFullName;
  [Description("Status of the job")]
  uint32 Status;
  [Description("Error code")]
  uint32 LastError;
  [Description("Progress of the application job")]
  uint32 Progress;
  [Description("Job creation time")]
  datetime CreationTime;
  [Description("List of Urls to download content from")]
  string DownloadUrlList[];
  [Description("List of dependent application frameworks")]
  string Dependencies[];
  [Description("List of content download Urls for dependent frameworks")]
  string DependencyUrlLists[];
  [Description("Action type of install, uninstall or upgrade")]
  uint32 ActionType;
  [Description("Application type of Windows modern app, web application or remote
application")]
  uint32 JobType;
  [Description("Windows modern app deployment options")]
  uint32 DeploymentOptions;
  [static,Description("Method to create application job")]
  uint32 CreateJob([In,Description("Job details as xml string")] string JobData);
};
```

© 2014 Microsoft. All rights reserved.

The following partial SyncML example demonstrates installing a modern enterprise application.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_AppInstallJob/MDM_AppInstallJob.JobID={unique job id}/Exec=CreateJob
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>JobData={serialized xml job data string}</Data>
  </Item>
</Exec>
```

The JobID key in the LocURI target above must be unique for each application.

At a bare minimum the Job data XML must contain the following elements:

```
<AppInstallJob id="{unique job id}">
  <Application PackageFullName="ac90d71c-1014-4530-a465-
aadd7f8c4916_1.0.0.0_neutral__g4ype1skzj3jy" ActionType="1" DeploymentOptions="1"
IsBundle="false">
    <ContentURLList>
      <ContentURL>
https://manage.contoso.com/mobiledownloadservice/WinRT/getapp?AppId=bbff96c1-3e29-4742-92fa-
a98b575ed05b&FileName=c234efa984ffa10fec78d2022d04457e.appx.bin
      </ContentURL>
    </ContentURLList>
    <FrameworkDependencies />
  </Application>
</AppInstallJob>
```

© 2014 Microsoft. All rights reserved.

The PackageFullName is not included in the application manifest file and must be created by the MDM service. The following algorithm can be used as an example.

```
//PackageFullName = <Name>_<Version>_<Architecture>_< Resources>_<Base32(SHA256(Publisher))>

public static class Windows8AppPublisherHash
{
   /// <summary>
   /// PublisherId = Base32 encoding of first 8 bytes of SHA256 hash of publisher
   /// </summary>
   static uint PUBLISHER_HASHED_BYTES = 8;

   /// <summary>
   /// Given a publisher string, uses Windows' algorithms to generate a publisher id.
   /// This id is one of the 5 identity tuples used for detecting a Windows8App application
   /// </summary>
   /// <param name="publisher">The Publisher attribute of the Identity element of an
AppxManifest.xml document</param>
   /// <returns>Encoded publisher string</returns>
   /// </remarks>
   public static String GetPublisherId(String publisher)
   {
      UnicodeEncoding unicode = new UnicodeEncoding();
      string publisherId;

      using (SHA256CryptoServiceProvider sha256 = new SHA256CryptoServiceProvider())
      {
         byte[] hash = sha256.ComputeHash(unicode.GetBytes(publisher));
         char[] chars = Base32Encode(hash, PUBLISHER_HASHED_BYTES);
         publisherId = new String(chars);
      }

      return publisherId;
   }

   /// <summary>
   /// Performs a lookup on an array and returns the char at the specified offset
   /// </summary>
   /// <param name="value">value to look up</param>
   /// <returns>char at the offset</returns>
   private static char ValueToDigit(byte value)
   {
      // base 32 alphabet variant is 0-9, A-Z except for i, l, o, and u.
      char[] base32DigitList = "0123456789abcdefghjkmnpqrstvwxyz".ToCharArray();
      return base32DigitList[value];
   }

   /// <summary>
   /// Convert a byte array into printable characters by using each
   /// sequence of 5 bits as an index into a table of printable characters.
   /// </summary>
   /// <param name="bytes">bytes to encode</param>
   /// <param name="byteCount">number of bytes to encode</param>
   /// <returns>Base32</returns>
   private static char[] Base32Encode(byte[] bytes, uint byteCount)
   {
      uint wcharCount = 0;
      uint wcharsIdx = 0;
      uint numBits = byteCount * 8;

      wcharCount = numBits / 5;
```

© 2014 Microsoft. All rights reserved.

```
      if (numBits % 5 != 0)
      {
         ++wcharCount;
      }

      char[] wchars = new char[wcharCount];

      // Consider groups of five bytes.  This is the smallest number of bytes that has a
number of bits
      // that's evently divisible by five.
      // Every five bits starting with the most significant of the first byte are made into a
base32 value.
      // Each value is used to index into the alphabet array to produce a base32 digit.
      // When out of bytes but the corresponding base32 value doesn't yet have five bits, 0 is
used.
      // Normally in these cases a particular number of '=' characters are appended to the
resulting base32
      // string to indicate how many bits didn't come from the actual byte value.  For our
purposes no
      // such padding characters are necessary.
      //
      // Bytes:         aaaaaaaa  bbbbbbbb  cccccccc  dddddddd  eeeeeeee
      // Base32 Values: 000aaaaa  000aaabb  000bbbbb  000bcccc  000ccccd  000ddddd 000ddeee
000eeeee
      //
      // Combo of byte    a & F8    a & 07    b & 3E    b & 01    c & 0F    d & 7C   d &
03   e & 1F
      // values except              b & C0              c & F0    d & 80             e &
E0
      // for shifting
      // Make sure the following math doesn't overflow.

      for (uint byteIdx = 0; byteIdx < byteCount; byteIdx += 5)
      {
         byte firstByte = bytes[byteIdx];
         byte secondByte = (byteIdx + 1) < byteCount ? bytes[byteIdx + 1] : (byte)0;

         wchars[wcharsIdx++] = ValueToDigit((byte)((firstByte & 0xF8) >> 3));
         wchars[wcharsIdx++] = ValueToDigit((byte)(
            ((firstByte & 0x07) << 2) | ((secondByte & 0xC0) >> 6)));

         if (byteIdx + 1 < byteCount)
         {
            byte thirdByte = (byteIdx + 2) < byteCount ? bytes[byteIdx + 2] : (byte)0;

            wchars[wcharsIdx++] = ValueToDigit((byte)((secondByte & 0x3E) >> 1));
            wchars[wcharsIdx++] = ValueToDigit((byte)
               (((secondByte & 0x01) << 4) | ((thirdByte & 0xF0) >> 4)));

            if (byteIdx + 2 < byteCount)
            {
               byte fourthByte = (byteIdx + 3) < byteCount ? bytes[byteIdx + 3] : (byte)0;

               wchars[wcharsIdx++] = ValueToDigit((byte)
                  (((thirdByte & 0x0F) << 1) | ((fourthByte & 0x80) >> 7)));

               if (byteIdx + 3 < byteCount)
               {
                  byte fifthByte = (byteIdx + 4) < byteCount ? bytes[byteIdx + 4] : (byte)0;

                  wchars[wcharsIdx++] = ValueToDigit((byte)((fourthByte & 0x7C) >> 2));
```

© 2014 Microsoft. All rights reserved.

```
                    wchars[wcharsIdx++] = ValueToDigit((byte)
                       (((fourthByte & 0x03) << 3) | ((fifthByte & 0xE0) >> 5)));

                    if (byteIdx + 4 < byteCount)
                    {
                       wchars[wcharsIdx++] = ValueToDigit((byte)(fifthByte & 0x1F));
                    }
                 }
              }
           }
        }

        return wchars;
     }
}
```

ActionType	Description
1	Install application
2	Un-install a previously installed application
3	Update a previously installed application

DeploymentOptions	Description
0	None
1	If this package, or any package that depends on this package, is currently in use, the processes associated with the package are shut down forcibly so that registration can continue. This is the recommended option for line-of-business (LOB) applications.
2	When you set this option, the app is installed in development mode. Use this option to enable key app development scenarios. You can't use this option in conjunction with a bundle package. If you use this option with a bundle package, your call returns ERROR_INSTALL_FAILED.
32	When you set this option, the app is instructed to skip resource applicability checks. This effectively stages or registers all resource packages that a user passes in to the command, which forces applicability for all packages contained in a bundle. If a user passes in a bundle, all contained resource packages will be registered.

For more information on deployment options consult this article on MSDN: http://msdn.microsoft.com/library/windows/apps/windows.management.deployment.deploymentoptions.aspx

© 2014 Microsoft. All rights reserved.

The following is an example of **AppInstallJob** XML serialized to a string.

```
&lt;AppInstallJob id="{unique job id}"&gt;&lt;Application PackageFullName="ac90d71c-1014-4530-
a465-aadd7f8c4916_1.0.0.0_neutral__g4ype1skzj3jy" ActionType="1" DeploymentOptions="1"
IsBundle="false"&gt;&lt;ContentURLList&gt;&lt;ContentURL&gt;https://manage.contoso.com/mobiled
ownloadservice/WinRT/getapp?AppId=bbff96c1-3e29-4742-92fa-
a98b575ed05b&amp;amp;FileName=c234efa984ffa10fec78d2022d04457e.appx.bin&lt;/ContentURL&gt;&lt;
/ContentURLList&gt;&lt;FrameworkDependencies /&gt;&lt;/Application&gt;&lt;/AppInstallJob&gt;
```

Framework dependencies

Some modern applications will have dependencies on external frameworks. This is especially true for modern applications written using the C++ language. The frameworks an application depends on can be discovered in the application manifest. The DM service should first query the device to see if the framework has already been installed on the device. If the framework doesn't exist on the device, the DM service must include a valid URI(s) to the framework in the AppInstallJob XML. The management agent will download the application and all framework dependencies. Dependencies should be restricted to those that are part of the application definition – extra dependencies that are not part of the application definition may cause an installation error. Once it has all the resources locally, it calls the appx packager which installs the frameworks and the application.

The following SyncML demonstrates a query to see if the framework has been installed on a device.

```
<Get>
  <CmdID>{unique command id}</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MDM_ApplicationFramework/MDM_ApplicationFramework.PackageName=%22Microsoft.VCLibs.120.
00.Debug%22,MinimumPackageVersion=%2212.0.21005.1%22,PackagePublisher=%22CN=Microsoft%20Corpor
ation,%20O=Microsoft%20Corporation,%20L=Redmond,%20S=Washington,%20C=US%22/PackageName
      </LocURI>
    </Target>
  </Item>
</Get>
```

The management agent will respond to the **Get** command with a Status of 200 if the framework has already been installed.

The following SyncML demonstrates how to install a framework on to a device.

© 2014 Microsoft. All rights reserved.

```
<Exec>
  <CmdID>7</CmdID>
  <Item>
    <Target>
      <LocURI>./cimv2/MDM_AppInstallJob/MDM_AppInstallJob.JobID=%22{unique job
id}%22/Exec=CreateJob</LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>JobData=&lt;AppInstallJob id="{unique job id}"&gt;&lt;Application
PackageFullName="9b3e4255-42a7-4eb5-abbd-
4cad3a507f69_2013.1115.1833.1331_neutral_~_vr9dnaefwk72e" ActionType="1" DeploymentOptions="1"
IsBundle="true"&gt;&lt;ContentURLList&gt;&lt;ContentURL&gt;https://contoso.com/downloadservice
/getapp?AppId=b10cf04c-d73a-4319-a092-
4506b6bf8ca2&amp;amp;FileName=4729b8db95bbf946d5f3d81e08ccde62.appxbundle.bin&lt;/ContentURL&g
t;&lt;/ContentURLList&gt;&lt;FrameworkDependencies&gt;&lt;FrameworkDependency
PackageFullName="Microsoft.Media.PlayReadyClient.2_2.8.1947.0_x86__8wekyb3d8bbwe"&gt;&lt;Conte
ntURLList&gt;&lt;ContentURL&gt;https://contoso.com/downloadservice/getapp?AppId=b10cf04c-d73a-
4319-a092-
4506b6bf8ca2&amp;amp;FileName=408844a8c1a4654f719e2e9a0c297cc8.appx.bin&lt;/ContentURL&gt;&lt;
/ContentURLList&gt;&lt;/FrameworkDependency&gt;&lt;FrameworkDependency
PackageFullName="Microsoft.VCLibs.120.00.Debug_12.0.21005.1_x86__8wekyb3d8bbwe"&gt;&lt;Content
URLList&gt;&lt;ContentURL&gt;https://contoso.com/downloadservice/getapp?AppId=b10cf04c-d73a-
4319-a092-
4506b6bf8ca2&amp;amp;FileName=a9700457d3b5363aa0937d9aa7334475.appx.bin&lt;/ContentURL&gt;&lt;
/ContentURLList&gt;&lt;/FrameworkDependency&gt;&lt;/FrameworkDependencies&gt;&lt;/Application&
gt;&lt;/AppInstallJob&gt;</Data>
  </Item>
</Exec>
```

The AppInstallJob from above before serialization.

© 2014 Microsoft. All rights reserved.

```
<AppInstallJob  id="{unique job id}">
  <Application
    PackageFullName="9b3e4255-42a7-4eb5-abbd-
4cad3a507f69_2013.1115.1833.1331_neutral_~_vr9dnaefwk72e"
    ActionType="1"
    DeploymentOptions="1"
    IsBundle="true">
    <ContentURLList>
      <ContentURL>
        https://contoso.com/downloadservice/getapp?AppId=b10cf04c-d73a-4319-a092-
4506b6bf8ca2&amp;amp;FileName=4729b8db95bbf946d5f3d81e08ccde62.appxbundle.bin
      </ContentURL>
    </ContentURLList>
    <FrameworkDependencies>
      <FrameworkDependency
PackageFullName="Microsoft.Media.PlayReadyClient.2_2.8.1947.0_x86__8wekyb3d8bbwe">
        <ContentURLList>
          <ContentURL>
            https://contoso.com/downloadservice/getapp?AppId=b10cf04c-d73a-4319-a092-
4506b6bf8ca2&amp;amp;FileName=408844a8c1a4654f719e2e9a0c297cc8.appx.bin
          </ContentURL>
        </ContentURLList>
      </FrameworkDependency>
      <FrameworkDependency
PackageFullName="Microsoft.VCLibs.120.00.Debug_12.0.21005.1_x86__8wekyb3d8bbwe">
        <ContentURLList>
          <ContentURL>
            https://contoso.com/downloadservice/getapp?AppId=b10cf04c-d73a-4319-a092-
4506b6bf8ca2&amp;amp;FileName=a9700457d3b5363aa0937d9aa7334475.appx.bin
          </ContentURL>
        </ContentURLList>
      </FrameworkDependency>
    </FrameworkDependencies>
  </Application>
</AppInstallJob>
```

MSADP applications

The DM service can also install Microsoft Store Application Developer Program (MSADP) applications, which are for EDU customers only. The Windows Store introduced a new feature where application developers can agree to allow their free applications to be distributed outside of the Windows store. The applications can be side-loaded and included in a custom image or installed via MDM using the Windows built-in management agent. The MDM agent does not require a sideloading key in order to install MSADP apps. Customers must request access for MSADP applications. Once the customer has access to the MSADP application catalog, they can download an application package and its accompanying license file as a single zipped file. The DM service must provide a mechanism to upload the package and the license file into the service. It should also be noted that all MSADP applications are signed by the Windows store. No additional signing is necessary by the customer or the DM service. Supporting MSADP applications is a simple change for the DM service. A XML element has been introduced in the AppInstallJob XML called a License. The license XML blob must not be reformatted or contain whitespaces. The XML must be inserted actually as it is in the license file.

© 2014 Microsoft. All rights reserved.

The example below demonstrates adding the License xml element to the application job demonstrated above.

```
<AppInstallJob id="{unique job id}">
  <Application PackageFullName="ac90d71c-1014-4530-a465-
aadd7f8c4916_1.0.0.0_neutral__g4ype1skzj3jy" ActionType="1" DeploymentOptions="1"
IsBundle="false">
       …
  </Application>
  <License Version="1" Source="OEM" xmlns="urn:schemas-microsoft-
com:windows:store:licensing:ls">
    {license xml blob}
  </License>
</AppInstallJob>
```

The DM service can request the status of an application install job. The partial SyncML demonstrates retrieving job status.

```
<Get>
  <CmdID>24</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MDM_AppInstallJob/MDM_AppInstallJob.JobID={unique job id}/Status
      </LocURI>
    </Target>
   </Item>
</Get>
```

© 2014 Microsoft. All rights reserved.

The following partial SyncML demonstrates the OMA-DM agent returning status.

```
<Status>
  <CmdID>2</CmdID>
  <CmdRef>24</CmdRef>
  <Cmd>Get</Cmd>
  <Data>200</Data>
</Status>
<Results>
  <CmdID>3</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>24</CmdRef>
  <Item>
    <Source>
      <LocURI>
        ./cimv2/MDM_AppInstallJob/MDM_AppInstallJob.JobID={unique job id}/Status
      </LocURI>
    </Source>
    <Meta>
      <Format xmlns="syncml:metinf">int</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>10</Data>
  </Item>
</Results>
```

The following table lists the possible **Status** values.

Status	Description
10	Initialized
20	Download in progress
30	Download failed
40	Download complete
50	Install in progress
60	Install failed
70	Install complete
80	Uninstall in progress
90	Uninstall failed
100	Uninstall complete
110	Hash mismatch
120	Sideloading is not enabled

© 2014 Microsoft. All rights reserved.

Status	Description
130	MSADP app license insall failed

The DM service can return an instance list of all applications installed on the device through MDM. The following partial SyncML demonstrates retrieving a list of installed applications.

```
<Get>
  <CmdID>3</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_AppInstallJob
      </LocURI>
    </Target>
   </Item>
</Get>
```

© 2014 Microsoft. All rights reserved.

The following partial SyncML demonstrates the OMA-DM agent returning status and a list of installed applications.

```
<Status>
  <CmdID>2</CmdID>
  <CmdRef>3</CmdRef>
  <Cmd>Get</Cmd>
  <Data>200</Data>
</Status>
<Results>
  <CmdID>3</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>3</CmdRef>
  <Item>
    <Source>
      <LocURI>
        ./cimv2/MDM_AppInstallJob
      </LocURI>
    </Source>
    <Meta>
      <Format xmlns="syncml:metinf">node</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>
./cimv2/MDM_AppInstallJob/MDM_AppInstallJob.JobID=ScopeId_0E5497DD-A1AD-43AC-BBB4-
73F7B13EF079:DeploymentType_af6d1a87-c710-4fc4-b266-1ae2d7106479:1
    </Data>
  </Item>
  <Item>
    <Source>
      <LocURI>
        ./cimv2/MDM_AppInstallJob
      </LocURI>
    </Source>
    <Meta>
      <Format xmlns="syncml:metinf">node</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>
./cimv2/MDM_AppInstallJob/MDM_AppInstallJob.JobID=ScopeId_0E5497DD-A1AD-43AC-BBB4-
73F7B13EF878:DeploymentType_af6d1a87-c710-4fc4-b266-1ae2d7106580:1
    </Data>
  </Item>
</Results>
```

The OMA-DM agent also supports deleting or un-installing a single enterprise line of business from the device. The same AppInstallJob XML that was used to install the application is used but the ActionType must be set to **2**. The partial SyncML demonstrates deleting a single applications.

```
<AppInstallJob id="{unique job id}">
  <Application PackageFullName="ac90d71c-1014-4530-a465-
aadd7f8c4916_1.0.0.0_neutral__g4ype1skzj3jy" ActionType="2" IsBundle="false">
    <FrameworkDependencies />
  </Application>
</AppInstallJob>
```

© 2014 Microsoft. All rights reserved.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_AppInstallJob/MDM_AppInstallJob.JobID={unique job id}/Exec=CreateJob
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>JobData={serialized xml job data string}</Data>
  </Item>
</Exec>
```

Web link applications

The following example shows how the DM service can install web links to the secondary start screen on a Windows 8.1-based device.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_AppInstallJob/MDM_AppInstallJob.JobID={unique job id}/Exec=CreateJob
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>JobData={serialized xml job data string}</Data>
  </Item>
</Exec>
```

At a bare minimum the Job data XML must contain the elements in the following example.

```
<AppInstallJob id="{unique job id}">
  <WebApplication PackageFullName="Test Web App – bing.com" ActionType="1"
DeploymentOptions="1">
    <ContentURLList>
      <ContentURL>http://bing.com</ContentURL>
    </ContentURLList>
  </WebApplication>
</AppInstallJob>
```

© 2014 Microsoft. All rights reserved.

The following example shows the above XML serialized to a string.

```
&lt;AppInstallJob id="{unique job id}"&gt;&lt;WebApplication PackageFullName="Test Web App -
bing.com" ActionType="1"
DeploymentOptions="1"&gt;&lt;ContentURLList&gt;&lt;ContentURL&gt;http://bing.com&lt;/ContentUR
L&gt;&lt;/ContentURLList&gt;&lt;/WebApplication&gt;&lt;/AppInstallJob&gt;
```

© 2014 Microsoft. All rights reserved.

DM device restrictions

The OMA-DM agent supports two different types of device restrictions – query/set and query only. The first type is a restriction the DM service can "set" on the device effectively disabling a hardware or software feature on the device. The first type of restrictions fall into the "control" camp. The IT manager can control a hardware or software feature on the device. The second type of restriction is one where the DM service can query the hardware or software feature on the device. The DM service can't disable or turn off the feature but it can report the status to an IT manager who then through policy can determine the action to take on a device that's out-of-compliance based on a feature that's enabled/disabled on the device. This type of restriction falls into the "governance" camp.

Some of the device restriction properties are exposed through the remote WMI class MDM_Restrictions.

```
[Description("This class provides configuration to restrict settings on the device")]
class MDM_Restrictions
{
    [Key,Description("The key to identify the instance of MDM_Restrictions class")]
    Uint32  Key;
    [Description("This property allows 'Windows Error Reporting' to be enabled or disabled on
the device")]
    boolean  DiagnosticsSubmissionEnabled;
    [Description("This property allows 'Data Roaming' to be enabled or disabled on the
device")]
    boolean  DataRoamingEnabled;
    [Read,Description("This property returns whether atleast one Bluetooth device is found and
is enabled on the device")]
    boolean  BluetoothEnabled;
    [Read,Description("This property returns whether atleast one wifi network adapter is found
and is enabled on the device")]
    boolean  WifiEnabled;
    [Read,Description("This property returns whether the 'Sync your settings' option under
'Change PC Settings' is turned on")]
    boolean  PCSettingsSyncEnabled;
    [Read,Description("This property returns whether the 'Sync settings over metered
connection' option under 'Change PC Settings' is turned on")]
    boolean  PCSettingsMeteredNetworkSyncEnabled;
    [Read,Description("This property returns whether the 'Passwords' option under 'Change PC
Settings' is turned on")]
    boolean  PCSettingsPasswordSyncEnabled;
    [Description("This property returns the Enterprise Client Sync product's url for which the
client will sync to")]
    string EcsSyncUrl;
    [Description("This property returns whether the Enterprise Client Sync product will be
auto provisioned")]
    boolean EcsAutoProvisionEnabled;
    [Description("This propery returns the current 'User Access Control' level on the
client"),
     Values {"Always Notify", "Notify App Changes", "Notify App Changes (No Dim)", "Never
Notify"}]
    Uint32  UserAccountControlStatus;
    [Description("True if Smart Screen is enabled")]
    boolean SmartScreenEnabled;
};
```

© 2014 Microsoft. All rights reserved.

The table lists the restriction properties in the MDM_Restrictions class along with a description and if the DM service can Get and/or Set the property.

Feature	Description	Get	Set
DiagnosticsSubmissionEnabled	Controls Windows Error Reporting on the device.	X	X
DataRoamingEnabled	Controls the data roaming setting on the device.	X	X
BlueToothEnabled	Returns if Bluetooth is enabled on the device.	X	
WifiEnabled	Returns if Wi-Fi is enabled on the device	X	
PCSettingsSyncEnabled	Returns true if settings are allowed to synchronization to a personal OneDrive account. Returns false if this setting has been disabled.	X	
PCSettingsMeteredNetworkSyncEnabled	Returns true if settings are allowed to synchronize over a metered network. Returns false if the setting has been disabled	X	
EcsAutoProvisionEnabled	Returns true if enterprise workfolders are setup to auto-provision on the device. False if the setting has been disabled.	X	
SmartScreenEnabled	Returns true if smart screen has been enabled on the device. False if the setting has been disabled.	X	

© 2014 Microsoft. All rights reserved.

The DM service must get an instance of the MDM_Restrictions class before it can send **Get** and **Replace** commands for the restriction properties. The partial SyncML example demonstrates how to retrieve an instance of the MDM_Restrictions class using a Get command.

```
<Get>
  <CmdID>7</CmdID>
  <Item>
    <Target>
      <LocURI>./cimv2/MDM_Restrictions</LocURI>
    </Target>
  </Item>
</Get>
```

The client will respond to the **Get** Command on the MDM_Restrictions class with **Status** and **Results** commands similar to the following.

```
<Status>
  <CmdID>2/CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>7</CmdRef>
  <Cmd>Get</Cmd>
  <Data>200</Data>
</Status>
<Results>
  <CmdID>3</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>7</CmdRef>
  <Item>
    <Source>
      <LocURI>./cimv2/MDM_Restrictions</LocURI>
    </Source>
    <Meta>
      <Format xmlns="syncml:metinf">node</Format>
    </Meta>
    <Data>MDM_Restrictions.Key="{instance id}"</Data>
  </Item>
</Results>
```

The DM service can now target an instance of the MDM_Restrictions class and send Get and Replace commands to get/set device restrictions. The following partial SyncML example demonstrates getting the DataRoamingEnabled property.

```
<Get>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_Restrictions/MDM_Restrictions.Key=%22{instance id}%22/DataRoamingEnabled
      </LocURI>
    </Target>
  </Item>
</Get>
```

Here is the MOF code for MDM_RestrictionsUser class:

```
[dynamic, provider("MDMSettingsProv")]class MDM_RestrictionsUser
```

© 2014 Microsoft. All rights reserved.

```
{
  Uint32  Key;
  boolean PCSettingsSyncEnabled;
  boolean PCSettingsMeteredNetworkSyncEnabled;
  boolean PCSettingsPasswordSyncEnabled;
};
```

The following table shows the properties for MDM_RestrictionsUser class.

Property	Description
Key	Identifies the MDM_RestrictionsUser class instance.
PCSettingsSyncEnabled	True if the Sync your settings option under Change PC Settings is on.
PCSettingsMeteredNetworkSyncEnabled	True if the Sync settings over metered connection option under Change PC Settings is on.
PCSettingsPasswordSyncEnabled	True if the Passwords option under Change PC Settings is on.

© 2014 Microsoft. All rights reserved.

DM device security status

The OMA-DM agent provides capabilities for the DM service to query and in some cases set security related properties on the Windows platform.

Device security properties are exposed through the remote WMI class MDM_SecurityStatus.

```
[Description("This class provides security health metrics on the device")]
class MDM_SecurityStatus
{
    [Key,Description("The key to identify the instance of MDM_SecurityStatus class")]
    Uint32  Key;
    [Description("This property provides the status of the firewall"),
     Values {"Good", "NotMonitored", "Poor", "Snooze"}]
    Uint32  FirewallStatus;
    [Read,Description("This property provides the status of Windows updates"),
     Values {"Disable Automatic Updates", "Notify for download and notify for install", "Auto
download and notify for install", "Auto download and schedule the install"}]
    Uint32  AutoUpdateStatus;
    [Read,Description("This property provides the status of antivirus"),
     Values {"Good", "NotMonitored", "Poor", "Snooze"}]
    Uint32  AntiVirusStatus;
    [Read,Description("This property provides the status of antivirus updates"),
     Values {"On", "Off", "Snoozed", "Expired"}]
    Uint32  AntiVirusSignatureStatus;
    [Read,Description("This property provides the status of bitlocker encryption")]
    boolean RequireEncryption;
    [Description("This property gets/sets the maintenance window start hour values 0 to 23")]
    Uint32  MaintenanceScheduleStartHour;
    [Description("This property gets/sets the maintenance window start delay pattern e.g.
PT30M")]
    string  MaintenanceScheduleDelayPattern;
    [Description("This property gets/sets whether to wake up the machine to start
maintenance")]
    boolean  MaintenanceScheduleAllowWakeup;
    [Description("This property gets/sets whether to make microsoft accounts optional to use
modern applications")]
    boolean  IsMicrosoftAccountOptional;
    [Description("This property gets/sets the application content uri rules. E.g.
'https://mail.microsoft.com/owa'")]
    string  ApplicationContentUriRules;
};
```

© 2014 Microsoft. All rights reserved.

The table lists the security properties in the MDM_Restrictions class along with a description. All MDM_SecurityStatusProperties are read only.

Feature	Description
FirewallStatus	Status of the built-in firewall 0 – Firewall is on and monitoring 1 – Firewall has been disabled 2 – Firewall is not monitoring all networks or some rules have been turned off 3 – Firewall is temporarily not monitoring all networks
AutoupdateStatus	0 – Windows update is enabled and configured to automatically download and install the updates. 1 – Windows update is enabled and configured to let the user choose to download updates and then install them. 2 – Windows update is disabled on the device. 3 – Windows update is enabled and configured to automatically download and schedule the install.
AntiVirusStatus	Returns the status of the antivirus
AntiVirusSignatureStatus	Returns the status of the antivirus signature updates

© 2014 Microsoft. All rights reserved.

The DM service must get an instance of the MDM_SecurityStatus class before it can send Get commands for the security status properties. The partial SyncML example demonstrates how to retrieve an instance of the MDM_SecurityStatus class using a Get command.

```
<Get>
  <CmdID>7</CmdID>
  <Item>
    <Target>
      <LocURI>./cimv2/MDM_SecurityStatus</LocURI>
    </Target>
  </Item>
</Get>
```

The client will respond to the Get Command on the MDM_SecurityStatus class with Status and Results commands similar to the following:

```
<Status>
  <CmdID>2/CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>7</CmdRef>
  <Cmd>Get</Cmd>
  <Data>200</Data>
</Status>
<Results>
  <CmdID>3</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>7</CmdRef>
  <Item>
    <Source>
      <LocURI>./cimv2/MDM_SecurityStatus</LocURI>
    </Source>
    <Meta>
      <Format xmlns="syncml:metinf">node</Format>
    </Meta>
    <Data>MDM_SecurityStatus.Key="{instance id}"</Data>
  </Item>
</Results>
```

The DM service can now target an instance of the MDM_SecurityStatus class and send Get commands to get real-time information on security components in the system. The following partial SyncML example demonstrates getting the FirewallStatus property.

```
<Get>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_SecurityStatus/MDM_SecurityStatus.Key=%22{instance id}%22/FirewallStatus
      </LocURI>
    </Target>
  </Item>
</Get>
```

© 2014 Microsoft. All rights reserved.

DM browser settings management

The OMA-DM agent provides the capability for a device management solution to manage the browser settings on a device. The current version of the OMA-DM agent only manages the desktop version of Internet Explorer. Any browser setting change through OMA-DM will not affect the modern Internet Explorer application. With this capability DM admins can force browser settings such as: force fraud warnings, block popups, and disable scripting in the browser. The DM service manages browser settings by targeting two remote WMI classes: MDM_BrowserSettings and MDM_BrowserSecurityZones. Both classes are shown below:

```
 [Description("This class provides configuration to modify browser security zone settings on
the device")]
class MDM_BrowserSecurityZones
{
    [Key,
    Description("A namespace that may belong in a security zone")]
    string         Namespace;
    [Key,
    ValueMap {"1", "2", "3", "4"},
    Values {"Intranet", "Trusted", "Internet", "Untrusted"},
    Description("Key identifier for the security zone")]
    Uint32              Zone;
    [Description("True if the namespace exists in the specified security zone")]
    boolean             Exists;
};

[Description("This class provides configuration to modify browser security settings on the
device")]
class MDM_BrowserSettings
{
    [Key,Description("The key to identify the instance of MDM_BrowserSettings class")]
    Uint32         Key;
    [Description("True if Force Fraud Warning is enabled")]
    boolean             ForceFraudWarning;
    [Description("True if Autofill is enabled")]
    boolean             AutofillEnabled;
    [Description("True if scripting is enabled for the Internet security zone")]
    boolean             InternetScriptingEnabled;
    [Description("True if plugins are enabled")]
    boolean             InternetPluginsEnabled;
    [Description("True if popups are disabled")]
    boolean             InternetBlockPopups;
    [Description("True if the always send do not track header setting is enabled")]
    boolean             AlwaysSendDoNotTrackHeader;
    [Description("True if the Intranet security zone is enabled")]
    boolean             IntranetSecurityZoneEnabled;
    [Description("True if the go to Intranet for single word setting is enabled")]
    boolean             GoToIntranetForSingleWord;
    [ValueMap {"0", "1", "2"},
    Values {"High", "Medium-high", "Medium"},
    Description("Security level for the Internet security zone")]
    Uint32              InternetZoneSecurityLevel;
    [ValueMap {"0", "1", "2", "3", "4"},
    Values {"High", "Medium-high", "Medium", "Medium-low", "Low"},
    Description("Security level for the Intranet security zone")]
```

© 2014 Microsoft. All rights reserved.

```
    Uint32              IntranetZoneSecurityLevel;
    [ValueMap {"0", "1", "2", "3", "4"},
    Values {"High", "Medium-high", "Medium", "Medium-low", "Low"},
    Description("Security level for the Restricted Sites security zone")]
    Uint32              RestrictedSitesZoneSecurityLevel;
    [ValueMap {"0", "1", "2", "3", "4"},
    Values {"High", "Medium-high", "Medium", "Medium-low", "Low"},
    Description("Security level for the Trusted Sites security zone")]
    Uint32              TrustedSitesZoneSecurityLevel;
};
```

Following the targeting pattern seen in most of the other examples the DM service must first get an instance of the remote WMI object before sending Get/Replace Commands to the device. The partial SyncML example demonstrates how to retrieve an instance of the MDM_BrowserSettings class using a Get command.

```
<Get>
  <CmdID>8</CmdID>
  <Item>
    <Target>
      <LocURI>./cimv2/MDM_BrowserSettings</LocURI>
    </Target>
  </Item>
</Get>
```

The OMA-DM agent will respond to the Get Command for the MDM_BrowserSettings class with Status and Results commands similar to the following.

© 2014 Microsoft. All rights reserved.

```
<Status>
  <CmdID>2/CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>8</CmdRef>
  <Cmd>Get</Cmd>
  <Data>200</Data>
</Status>
<Results>
  <CmdID>3</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>8</CmdRef>
  <Item>
    <Source>
      <LocURI>./cimv2/MDM_BrowserSettings</LocURI>
    </Source>
    <Meta>
      <Format xmlns="syncml:metinf">node</Format>
    </Meta>
    <Data>MDM_BrowserSettings.Key="{instance id}"</Data>
  </Item>
</Results>
```

The DM service can now target an instance of the MDM_BrowserSettings class and send Get and Replace SyncML Commands to manage browser settings. The following partial SyncML example demonstrates getting the ForceFraudWarning property.

```
<Get>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
     ./cimv2/MDM_BrowserSettings/MDM_BrowserSettings.Key=%22{instance id}%22/ForceFraudWarning
      </LocURI>
    </Target>
  </Item>
</Get>
```

The following demonstrates getting the InternetZoneSecurityLevel and then setting it to the "Low" level.

```
<Get>
  <CmdID>10</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_BrowserSettings/MDM_BrowserSettings.Key=%221%22/InternetZoneSecurityLevel
      </LocURI>
    </Target>
  </Item>
</Get>
```

© 2014 Microsoft. All rights reserved.

```
<Status>
  <CmdID>2</CmdID>
  <CmdRef>10</CmdRef>
  <Cmd>Get</Cmd>
  <Data>200</Data>
</Status>
<Results>
  <CmdID>3</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>10</CmdRef>
  <Item>
    <Source>
      <LocURI>
        ./cimv2/MDM_BrowserSettings/MDM_BrowserSettings.Key=%221%22/InternetZoneSecurityLevel
      </LocURI>
    </Source>
    <Meta>
      <Format xmlns="syncml:metinf">int</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>2</Data>
  </Item>
</Results>
```

SyncML demonstrating setting the “InternetZoneSecurityLevel” to “High”

```
<Replace>
  <CmdID>11</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_BrowserSettings/MDM_BrowserSettings.Key=%221%22/InternetZoneSecurityLevel
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">int</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>0</Data>
  </Item>
</Replace>
```

© 2014 Microsoft. All rights reserved.

DM parental control

The OMA-DM agent includes in the approved list, the Windows parental control remote WMI classes. These classes are only valid for "standard" rights users that were enrolled via the "enroll on behalf of" protocol. The parental control APIs bring a wealth of restrictions to a Windows-based device and are useful when a managed Windows is in a classroom setting or a Windows-based device is company owned and deployed for a specific task.

The following table shows the remote WMI Wpc classes are available. The LocURI path is ./cimv2/, such as ./cimv2/WpcRatingsSystem.

Windows Parental Control WMI Class	SyncML Get	SyncML Replace	SyncML Exec
WpcRating	X		
WpcRatingDescriptor	X		
WpcRatingsSystem	X		
WpcSystemSettings	X	X	X
WpcUserSettings	X	X	
WpcAppOverride	X		
WpcGameOverride	X	X	
WpcGamesRatings			
WpcGamesSettings	X	X	
WpcWebSettings	X	X	
WpcURLOverride	X	X	

© 2014 Microsoft. All rights reserved.

```
//**********************************************************************
//* MOF describing Ratings Systems
//**********************************************************************
#pragma namespace("\\\\.\\ROOT\\CIMV2\\Applications\\WindowsParentalControls")

//**********************************************************************
//* Class: WpcRating
//* Derived from:
//**********************************************************************
[Description("A rating within a ratings system"): ToInstance ToSubClass, dynamic:
DisableOverride ToInstance, provider("WpcSProv"): ToInstance ToSubClass]
class WpcRating
{
       [read: ToInstance ToSubClass, Description("Rating ID"): ToInstance ToSubClass, key,
Not_null: ToInstance ToSubClass] string ID;
       [read: ToInstance ToSubClass, Description("Long Name for this Rating"): ToInstance
ToSubClass] string LongName;
       [Description("ID of the Ratings System this rating belongs to"): ToInstance ToSubClass,
read: ToInstance ToSubClass, key, Not_null: ToInstance ToSubClass] string SystemID;
       [read: ToInstance ToSubClass, Description("Short Name for this Rating"): ToInstance
ToSubClass] string ShortName;
       [read: ToInstance ToSubClass, Description("Description of this rating"): ToInstance
ToSubClass] string Description;
       [read: ToInstance ToSubClass, Description("Has this rating been deprecated"):
ToInstance ToSubClass, Not_null: ToInstance ToSubClass] string IsDeprecated = "false";
       [read: ToInstance ToSubClass, Description("Link to updated rating"): ToInstance
ToSubClass] string UpdatedRating;
       [read: ToInstance ToSubClass, Description("Relative level index of this rating"):
ToInstance ToSubClass, Not_null: ToInstance ToSubClass] uint32 Level;
       [read: ToInstance ToSubClass, Description("Recommended minimum age"): ToInstance
ToSubClass] uint32 MinAge = 0;
       [read: ToInstance ToSubClass, Description("Path to the logo for the ratings system"):
ToInstance ToSubClass] string IconPath;
       [read: ToInstance ToSubClass, Description("Resource ID of the logo for the ratings
system"): ToInstance ToSubClass] string IconResourceID;
       [read: ToInstance ToSubClass, Description("Resource Type of the logo for the ratings
system"): ToInstance ToSubClass] string IconType;
};

//**********************************************************************
//* Class: WpcRatingsDescriptor
//* Derived from:
//**********************************************************************
[Description("Optional textual descriptor for a rating system"): ToInstance ToSubClass,
dynamic: DisableOverride ToInstance, provider("WpcSProv"): ToInstance ToSubClass]
class WpcRatingsDescriptor
{
       [read: ToInstance ToSubClass, Description("ID for descriptor"): ToInstance ToSubClass,
key, Not_null: ToInstance ToSubClass] string ID;
       [key, Description("GUID representing the ratings system the descriptor belongs to"):
ToInstance ToSubClass, read: ToInstance ToSubClass, Not_null: ToInstance ToSubClass] string
SystemID;
       [read: ToInstance ToSubClass, Description("Name of the descriptor"): ToInstance
ToSubClass, Not_null: ToInstance ToSubClass] string Name;
       [read: ToInstance ToSubClass, Description("Description of the descriptor"): ToInstance
ToSubClass] string Description;
       [read: ToInstance ToSubClass, Description("Has the description been deprecated"):
ToInstance ToSubClass, Not_null: ToInstance ToSubClass] boolean IsDeprecated = FALSE;
       [read: ToInstance ToSubClass, Description("Link to updated descriptor"): ToInstance
ToSubClass] string UpdatedDescriptor;
       [read: ToInstance ToSubClass, Description("Path to the logo for the ratings system"):
ToInstance ToSubClass] string IconPath;
```

© 2014 Microsoft. All rights reserved.

```
	[read: ToInstance ToSubClass, Description("Resource ID of the logo for the ratings
system"): ToInstance ToSubClass] string IconResourceID;
	[read: ToInstance ToSubClass, Description("Resource Type of the logo for the ratings
system"): ToInstance ToSubClass] string IconType;
	[read: ToInstance ToSubClass, Description("Extended properties for this descriptor"):
ToInstance ToSubClass] uint32 Properties;
};

//************************************************************************
//* Class: WpcRatingsSystem
//* Derived from:
//************************************************************************
[Description("Third Party Ratings System"): ToInstance ToSubClass, dynamic: DisableOverride
ToInstance, provider("WPCSprov"): ToInstance ToSubClass]
class WpcRatingsSystem
{
	[key, read: ToInstance ToSubClass, Description("GUID identifying the ratings system"):
ToInstance ToSubClass] string ID;
	[read: ToInstance ToSubClass, Description("Type of the ratings system"): ToInstance
ToSubClass] uint32 Type;
	[read: ToInstance ToSubClass, Description("Long name for the system"): ToInstance
ToSubClass] string LongName;
	[read: ToInstance ToSubClass, Description("Short name for the ratings system"):
ToInstance ToSubClass] string ShortName;
	[read: ToInstance ToSubClass, Description("Description of the ratings system"):
ToInstance ToSubClass, Not_null: ToInstance ToSubClass] string Description;
	[read: ToInstance ToSubClass, Description("Web address for the ratings system"):
ToInstance ToSubClass] string WebAddress;
	[read: ToInstance ToSubClass, Description("Path to the logo for the ratings system"):
ToInstance ToSubClass] string LogoPath;
	[read: ToInstance ToSubClass, Description("Resource ID of the logo for the ratings
system"): ToInstance ToSubClass] string LogoResourceID;
	[read: ToInstance ToSubClass, Description("Resource type of the logo for the ratings
system"): ToInstance ToSubClass] string LogoResourceType;
	[read: ToInstance ToSubClass, Description("Version of this data"): ToInstance
ToSubClass] string Version;
};



The following syntax is from WpcSettings.mof
// WPC Class definitions

#pragma namespace("\\\\.\\ROOT\\CIMV2\\Applications\\WindowsParentalControls")

//************************************************************************
//* Class: WpcSystemSettings
//* Derived from:
//************************************************************************
[Singleton: DisableOverride ToInstance ToSubClass, Description("Stores system-wide Wpc
settings"): ToInstance ToSubClass, provider("WpcSProv"): ToInstance, dynamic: ToInstance]
class WpcSystemSettings
{
	[read: ToInstance ToSubClass, write: ToInstance ToSubClass, Description("Link to the
current games rating system in force for the computer"): ToInstance ToSubClass] string
CurrentGamesRatingSystem;
    [read: ToInstance ToSubClass, write: ToInstance ToSubClass, Description("HTTP app filter
exemption list"): ToInstance ToSubClass] string HTTPExemptionList[];
    [read: ToInstance ToSubClass, Description("Windows HTTP app filter exemption list"):
ToInstance ToSubClass] string WinHTTPExemptionList[];
    [read: ToInstance ToSubClass, write: ToInstance ToSubClass, Description("HTTP URL filter
exemption list"): ToInstance ToSubClass] string URLExemptionList[];
```

© 2014 Microsoft. All rights reserved.

```
    [read: ToInstance ToSubClass, Description("Windows HTTP URL filter exemption list"):
ToInstance ToSubClass] string WinURLExemptionList[];
    [read: ToInstance ToSubClass, write: ToInstance ToSubClass, Description("Number of days
until a balloon reminder for checking the logs"): ToInstance ToSubClass] uint32
LogViewReminderInterval;
    [read: ToInstance ToSubClass, write: ToInstance ToSubClass, Description("Last time the
admin checked the log viewer"): ToInstance ToSubClass] DateTime LastLogView;
    [read: ToInstance ToSubClass, write: ToInstance ToSubClass, Description("ID of current web
filter"): ToInstance ToSubClass] string FilterID;
    [read: ToInstance ToSubClass, write: ToInstance ToSubClass, Description("Name of current
web filter"): ToInstance ToSubClass] string FilterName;
       [Description("Creates settings for a new User SID"): ToInstance ToSubClass,
Implemented: ToInstance ToSubClass, Static: ToInstance ToSubClass] uint32
AddUser([Description("SID to create settings for"): ToInstance ToSubClass, IN] string strSID);
       [Description("Removes settings for a user account"): ToInstance ToSubClass,
Implemented: ToInstance ToSubClass, Static: ToInstance ToSubClass] uint32
RemoveUser([Description("SID of the account to delete settings for"): ToInstance ToSubClass,
IN] string strSID);
};

//*********************************************************************
//* Class: WpcUserSettings
//* Derived from:
//*********************************************************************
[Description("Per-account Windows Parental Controls Settings"): ToInstance ToSubClass,
provider("WpcSprov"): ToInstance, dynamic: ToInstance]
class WpcUserSettings
{
       [key, read: ToInstance ToSubClass, Description("SID of the user account owning these
settings"): ToInstance ToSubClass] string SID;
       [read: ToInstance ToSubClass, write: ToInstance ToSubClass, Description("Is WPC enabled
for this user"): ToInstance ToSubClass] boolean WpcEnabled;
       [write: ToInstance ToSubClass, Description("Is logging enabled for this user"):
ToInstance ToSubClass, read: ToInstance ToSubClass] boolean LoggingRequired;
       [write: ToInstance ToSubClass, Description("Are hourly restrictions enabled for this
user"): ToInstance ToSubClass, read: ToInstance ToSubClass] boolean HourlyRestrictions;
       [write: ToInstance ToSubClass, Description("Are override requests enabled for this
user"): ToInstance ToSubClass, read: ToInstance ToSubClass] boolean OverrideRequests;
       [write: ToInstance ToSubClass, Description("Logon Hours mask for this user"):
ToInstance ToSubClass, read: ToInstance ToSubClass] uint32 LogonHours[7];
       [write: ToInstance ToSubClass, Description("Are application restrictions enabled for
this user"): ToInstance ToSubClass, read: ToInstance ToSubClass] boolean AppRestrictions;
};

//*********************************************************************
//* Class: WpcGameOverride
//* Derived from:
//*********************************************************************
[Description("Entry in game silo override list for Windows Parental Controls"): ToInstance
ToSubClass, dynamic: DisableOverride ToInstance, provider("WpcSProv"): ToInstance ToSubClass]
class WpcGameOverride
{
       [key, read: ToInstance ToSubClass, Description("Application ID identifying the game"):
ToInstance ToSubClass] string AppID;
       [key, read: ToInstance ToSubClass, Description("SID of the account owning this entry"):
ToInstance ToSubClass] string SID;
       [Description("Whether the entry is allowed or denied"): ToInstance ToSubClass, read:
ToInstance ToSubClass, write: ToInstance ToSubClass, MinValue(0): ToInstance ToSubClass,
MaxValue(2): ToInstance ToSubClass] uint32 Allowed;
};

//*********************************************************************
```

© 2014 Microsoft. All rights reserved.

```
//* Class: WpcAppOverride
//* Derived from:
//**************************************************************************
[Description("Entry in user silo application override list for Windows Parental Controls"):
ToInstance ToSubClass, dynamic: DisableOverride ToInstance, provider("WpcSProv"): ToInstance
ToSubClass]
class WpcAppOverride
{
       [key, read: ToInstance ToSubClass, Description("Rule ID for this entry"): ToInstance
ToSubClass] string RuleID;
       [key, read: ToInstance ToSubClass, Description("SID of the account owning this entry"):
ToInstance ToSubClass] string SID;
       [read: ToInstance ToSubClass, Description("Path to the overridden app"): ToInstance
ToSubClass] string Path;
};

//**************************************************************************
//* Class: WpcGamesSettings
//* Derived from:
//**************************************************************************
[Description("Games silo settings for Windows Parental Controls"): ToInstance ToSubClass,
dynamic: DisableOverride ToInstance, provider("WpcSProv"): ToInstance ToSubClass]
class WpcGamesSettings
{
       [key, read: ToInstance ToSubClass, Description("SID of the account owning these
settings"): ToInstance ToSubClass] string SID;
       [read: ToInstance ToSubClass, write: ToInstance ToSubClass, Description("Whether games
are permitted"): ToInstance ToSubClass] boolean Blocked;
       [read: ToInstance ToSubClass, Description("GUID of the rating system these settings are
for"): ToInstance ToSubClass] string SystemID;
       [read: ToInstance ToSubClass, Description("Whether to allow unrated games"): ToInstance
ToSubClass, write: ToInstance ToSubClass] boolean AllowUnrated;
       [read: ToInstance ToSubClass, Description("ID of the maximum rating allowed under the
current system"): ToInstance ToSubClass, write: ToInstance ToSubClass] string MaxAllowed;
       [read: ToInstance ToSubClass, Description("Collection of denied descriptors"):
ToInstance ToSubClass, write: ToInstance ToSubClass] string DeniedDescriptors[];
};

//**************************************************************************
//* Class: WpcURLOverride
//* Derived from:
//**************************************************************************
[Description("Entry in web silo URL override list for Windows Parental Controls"): ToInstance
ToSubClass, dynamic: DisableOverride ToInstance, provider("WpcSProv"): ToInstance ToSubClass]
class WpcURLOverride
{
       [key, read: ToInstance ToSubClass, Description("URL to be affected"): ToInstance
ToSubClass] string URL;
       [key, read: ToInstance ToSubClass, Description("SID of the account owning this entry"):
ToInstance ToSubClass] string SID;
       [Description("Whether the entry is allowed or denied"): ToInstance ToSubClass, read:
ToInstance ToSubClass, write: ToInstance ToSubClass, MinValue(0): ToInstance ToSubClass,
MaxValue(2): ToInstance ToSubClass] uint32 Allowed;
};

//**************************************************************************
//* Class: WpcWebSettings
//* Derived from:
//**************************************************************************
[Description("Per-account Windows Parental Controls Web Settings"): ToInstance ToSubClass,
provider("WpcSprov"): ToInstance, dynamic: ToInstance]
class WpcWebSettings
```

© 2014 Microsoft. All rights reserved.

```
{
       [key, Description("SID of the user account owning these settings"): ToInstance
ToSubClass, read: ToInstance ToSubClass] string SID;
    [Description("Is filter on for this user"): ToInstance ToSubClass, read: ToInstance
ToSubClass, write: ToInstance ToSubClass] boolean FilterOn;
    [Description("Filter level for this user"): ToInstance ToSubClass, read: ToInstance
ToSubClass, write: ToInstance ToSubClass, MinValue(0): ToInstance ToSubClass, MaxValue(5):
ToInstance ToSubClass] Uint32 FilterLevel;
       [Description("Are file downloads blocked for this user"): ToInstance ToSubClass, read:
ToInstance ToSubClass, write: ToInstance ToSubClass] boolean FileDownloadsBlocked;
       [Description("Are unrated sites blocked for this user"): ToInstance ToSubClass, read:
ToInstance ToSubClass, write: ToInstance ToSubClass] boolean BlockUnrated;
    [Description("Blocked Category list"): ToInstance ToSubClass, read: ToInstance ToSubClass,
write: ToInstance ToSubClass] uint32 BlockedCategoryList[];
};


// End of WPC Class Definitions
```

© 2014 Microsoft. All rights reserved.

DM certificate management

The OMA-DM agent supports installing root and CA certificates via SyncML (MDM connection). The OMA-DM agent also supports provisioning client or identity certificates with the help of the Simple Certificate Enrollment Protocol (SCEP).

Root/CA certificates

The first section covers root and CA certificates; how to install them, how to query for a known certificate, and how to delete a certificate that was previously installed through the management service. The remote WMI class MDM_Certificate is the resource a DM service targets for managing root and CA certificates. Valid store names are: "Root" and "CA".

```
[Description("This class provides the ability to install and delete the certificates on the
device")]
class MDM_Certificate
{
    [key, Values {"1", "2"}, ValueMap {"ContextUser", "ContextMachine"},
Description("Certificate store location")]
    uint8      StoreLocation;

    [key, Description("Internal store name to install the certificate")]
    string       StoreName;

    [key, Description("Thumbprint of the certificate")]
    string      Thumbprint;

    [read, Description("Is certificate installed")]
    boolean     IsInstalled;

    [Description("Base-64 encoded raw certificate blob")]
    string      Blob;
};
```

Notice MDM_Certificate is a multi-key class. The following SyncML demonstrates how to install a root certificate in the machine wide certificate store. The StoreLocation, StoreName and Thumbprint properties are set in the Target LocURI. The certificate itself is base64 encoded and set in the Data element of the SyncML message.

```
<Replace>
  <CmdID>2</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MDM_Certificate/MDM_Certificate.StoreLocation=%222%22,StoreName=%22Root%22,Thumbprint=
%225BE128213D05DC6CB87A059469130FC6686992EF%22/Blob
      </LocURI>
    </Target>
    <Data>
MIIB2TCCAUagAwIBAgIQyRjUagpgKYJP1AZXnPL4SDAJBgUrDgMCHQUAMBwxGjAYBgNVBAMTEVNDX09ubGluZV9Jc3N1aW
5nMB4XDTA4MTEyNjAyMDAzMloXDTE0MDMyNjAyMDAzMlowHDEaMBgGA1UEAxMRU0NfT25saW5lX0lzc3VpbmcwgZ8wDQYJ
KoZIhvcNAQEBBQADgY0AMIGJAoGBAM6CkO62YpeXEl8cwkN7ycAfl/L2Z9bdbyfqxGLx89zwSiUBrvoEDtnt9fWGkTQ5LV
z+lJy/k/TRLiM/3h94FT00QzBkE6yICLq7UCX5z8LRFEHSsGGaBgkotrkPsX3pG6agczNqF2h4Bngq+4BVbsA2PMpsXiVt
```

© 2014 Microsoft. All rights reserved.

```
WUtVHt45PD/JAgMBAAGBEQBawy5oQj84R6nsmIFiEHNpghEAWsMuaEI/OEep7JiBYhBzaTAJBgUrDgMCHQUAA4GBAC3315
98+rVBg+ipGb0Q4rreQFBlkFWa2Xv4EaOOxInq0PMtRxCxMCuLJIkSeg8IMSaAU0xGZYPaHSNHF1vUqrUkLEjjyXNa5URb
o322KCR2iTu5URddxD1FCE5SXwzZ/YHD3U9GvKfSQXZNDiF+VI8mtNll5jUhjzaIFtWL4dMe
    </Data>
  </Item>
</Replace>
```

The DM service can delete the certificate from the example above by issuing the following (partial) SyncML.

```
<Delete>
  <CmdID>3</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MDM_Certificate/MDM_Certificate.StoreLocation=%222%22,StoreName=%22Root%22,Thumbprint=
%225BE128213D05DC6CB87A059469130FC6686992EF%22
      </LocURI>
    </Target>
  </Item>
</Delete>
```

Also, the DM service can query if the certificate used in the above examples is installed on the device. The partial SyncML demonstrates how.

```
<Get>
  <CmdID>3</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MDM_Certificate/MDM_Certificate.StoreLocation=%222%22,StoreName=%22Root%22,Thumbprint=
%225BE128213D05DC6CB87A059469130FC6686992EF%22/IsInstalled
      </LocURI>
    </Target>
  </Item>
</Get>
```

Client certificates

Client certificates are installed via the Simple Certificate Enrollment Protocol (SCEP). The DM service sends a certificate template payload to the OMA-DM agent. The OMA-DM agent then hands the payload to the SCEP client on the device. The payload contains all the information the SCEP client needs to request a certificate from a SCEP or NDES server.

© 2014 Microsoft. All rights reserved.

The following partial SyncML demonstrates a certificate enrollment request.

```
<Add>
  <CmdID>20</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_CertificateEnrollment.RequestID="e74ae2c3-50b8-4036-a51e-
604cbffdea3b",StoreLocation="1",EnhancedKeyUsages="1.3.6.1.5.5.7.3.2",Issuers="CN=CertificateA
uthority"
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">xml</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>
       <CertificateRequest>
             <ConfigurationParameters
xmlns="http://schemas.microsoft.com/SystemCenterConfigurationManager/2012/03/07/CertificateEnr
ollment/ConfigurationParameters">
                <ExpirationThreshold>20</ExpirationThreshold>
                <RetryCount>1</RetryCount>
                <RetryDelay>1</RetryDelay>
                <TemplateName>SMS_ClientCopy</TemplateName>
                <KeyStorageProviderSetting>2</KeyStorageProviderSetting>
                <KeyUsage>160</KeyUsage>
                <KeyLength>1024</KeyLength>
                <HashAlgorithms>
                   <HashAlgorithm>SHA-1</HashAlgorithm>
                </HashAlgorithms>
                <CAThumbprint>6429CC067E892A2E63A53A9A332CE5DB1B04F82C</CAThumbprint>
                <ValidityPeriod>1</ValidityPeriod>
                <ValidityPeriodUnit>Years</ValidityPeriodUnit>
                <EKUMapping>
                   <EKUMap>
                      <EKUName>Client Authentication</EKUName>
                      <EKUOID>1.3.6.1.5.5.7.3.2</EKUOID>
                   </EKUMap>
                </EKUMapping>
             </ConfigurationParameters>
             <RequestParameters>
                <CertificateRequestToken>...</CertificateRequestToken>
                <SubjectName>CN=User</SubjectName>
                <SubjectAlternativeName>
                   <SANs>
                      <SAN NameFormat="33554432" AltNameType="11"
OID="1.3.6.1.4.1.311.20.2.3">User@certmgmt.contoso.com</SAN>
                   </SANs>
                </SubjectAlternativeName>
                <NDESUrl>http://ndes7.contoso.com/certsrv/mscep/mscep.dll</NDESUrl>
             </RequestParameters>
       </CertificateRequest>
    </Data>
  </Item>
</Add>
```

The following table describes properties used in a certificate enrollment request.

© 2014 Microsoft. All rights reserved.

Property	Description
RequestID	Unique Certificate Enrollment Request ID. Typically a GUID to track each certificate request status/compliance.
StoreLocation	Certificate store location where to check/install certificates issued to devices/users. Possible values {1, 2} means {"ContextUser", "ContextMachine"}.
EnhancedKeyUsages	Used as certificate selection criteria: enhanced key usage OIDs with + as delimiter. The certificate selected must match on all specified EKUs
Issuers	Used as certificate selection criteria: Issuer subject names with | as delimiter. The certificate selected must match one of specified issuers. Issuer subject names are case sensitive. Example: "CN=Contoso Root CA; OU=Servers; O=Contoso, Ltd; C=US | CN=Litware Corporate Root CA; O=Litware, Inc."

CertificateRequest.ConfigurationParameters

Parameter	Description
TemplateName	Name of the PKI enterprise CA certificate template to use in certificate enrollment request.
EKUMapping	List of EKU OIDs to be added in certificate enrollment request
KeyStorageProviderSetting	Setting that determines the behavior of where private keys should be stored/installed on devices such as TPM chip or sofware based key storage provider. • 1 = InstallToTPM_FailIfNotPresent • 2 = InstallToTPM_IfPresent • 3 = InstallToSoftwarekeyStorageProvider
KeyUsage	Private key usage flags as defined in http://msdn.microsoft.com/en-us/library/windows/desktop/aa379410(v=vs.85).aspx
KeyLength	Required number of bits in RSA private key
HashAlgorithms	The supported hash algorithm strings that must be used for certificate enrollment request to NDES/SCEP server.
RetryCount	Number of minutes delay between retry attempts
RetryDelay	Number of minutes delay between retry attempts
ValidityPeriod	Certificate validity period to use for the certificate enrollment request as

© 2014 Microsoft. All rights reserved.

Parameter	Description
	defined in http://msdn.microsoft.com/en-us/library/cc249868.aspx
ValidityPeriodUnits	Certificate validity period units to use for the certificate enrollment request as defined in http://msdn.microsoft.com/en-us/library/cc249868.aspx
CAThumbprint	Hex encoded thumbprint to uniquely identify the issuing CA associated with NDES/SCEP server.
ExpirationThreshold	Configuration threshold to determine when the issued certificate is about to expire and needs to be renewed by the device. This is the validity period remaining in percentage. 0 means already expired. It is % of remaining validity period between ValidFrom and ValidTo fields.

CertificateRequest.RequestParameters

The parameters below are unique to each certificate enrollment request instance.

Parameter	Description
CertificateRequestToken	Opaque SCEP challenge blob required for issuing certificate via NDES/SCEP protocol
SubjectName	Subject name to be used for the certificate enrollment
SubjectAlternativeName	List of subject alternative names to be used for the certificate enrollment request
NDESUrl	List of ; delimited NDES/SCEP server URLs for submitting SCEP certificate enrollment request

© 2014 Microsoft. All rights reserved.

DM Wi-Fi profile management

The OMA-DM agent supports installing WIFI profiles onto the device. The OMA-DM agent designers chose to leverage the existing Windows WLAN_Profiles as the schema within the SyncML to describe the profile. For a complete description of the WLAN_Profile schema please see MSDN.

WIFI profiles are installed by targeting the remote WMI class MDM_WirelessProfileXml. The class is documented below.

```
[InPartition {"local-user"},Description("This class provides ability to manage wireless
profiles through WLAN xml")]
class MDM_WirelessProfileXml
{
    [key, Description("Name of a wireless LAN profile.")]
    string            Name;
    [Description("Wireless Profile xml for this connection.")]
    string            ProfileXml;
};
```

The following partial SyncML demonstrates installing a basic WIFI profile. The Data element contains a serialized WLAN_Profile XML document.

```
<Replace>
  <CmdID>6</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_WirelessProfileXml.Name=%22Test%20WiFi%20Profilek%22/ProfileXml
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">xml</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>&lt;WLANProfile
xmlns="http://www.microsoft.com/networking/WLAN/profile/v1"&gt;&lt;name&gt;WiFi
Network&lt;/name&gt;&lt;SSIDConfig&gt;&lt;SSID&gt;&lt;name&gt;12345&lt;/name&gt;&lt;/SSID&gt;&
lt;nonBroadcast&gt;false&lt;/nonBroadcast&gt;&lt;/SSIDConfig&gt;&lt;connectionType&gt;ESS&lt;/
connectionType&gt;&lt;connectionMode&gt;manual&lt;/connectionMode&gt;&lt;autoSwitch&gt;false&l
t;/autoSwitch&gt;&lt;MSM&gt;&lt;security&gt;&lt;authEncryption&gt;&lt;authentication&gt;WPA2&l
t;/authentication&gt;&lt;encryption&gt;AES&lt;/encryption&gt;&lt;useOneX&gt;true&lt;/useOneX&g
t;&lt;FIPSMode
xmlns="http://www.microsoft.com/networking/WLAN/profile/v2"&gt;false&lt;/FIPSMode&gt;&lt;/auth
Encryption&gt;&lt;PMKCacheMode&gt;disabled&lt;/PMKCacheMode&gt;&lt;OneX
xmlns="http://www.microsoft.com/networking/OneX/v1"&gt;&lt;cacheUserData&gt;false&lt;/cacheUse
rData&gt;&lt;authMode&gt;machineOrUser&lt;/authMode&gt;&lt;EAPConfig&gt;&lt;EapHostConfig
xmlns="http://www.microsoft.com/provisioning/EapHostConfig"&gt;&lt;EapMethod&gt;&lt;Type
xmlns="http://www.microsoft.com/provisioning/EapCommon"&gt;25&lt;/Type&gt;&lt;VendorId
xmlns="http://www.microsoft.com/provisioning/EapCommon"&gt;0&lt;/VendorId&gt;&lt;VendorType
xmlns="http://www.microsoft.com/provisioning/EapCommon"&gt;0&lt;/VendorType&gt;&lt;AuthorId
xmlns="http://www.microsoft.com/provisioning/EapCommon"&gt;0&lt;/AuthorId&gt;&lt;/EapMethod&gt
;&lt;Config xmlns="http://www.microsoft.com/provisioning/EapHostConfig"&gt;&lt;Eap
xmlns="http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1"&gt;&lt;Type&gt;25&
lt;/Type&gt;&lt;EapType
```

© 2014 Microsoft. All rights reserved.

```
xmlns="http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV1"&gt;&lt;ServerValida
tion&gt;&lt;DisableUserPromptForServerValidation&gt;false&lt;/DisableUserPromptForServerValida
tion&gt;&lt;ServerNames&gt;&lt;/ServerNames&gt;&lt;/ServerValidation&gt;&lt;FastReconnect&gt;t
rue&lt;/FastReconnect&gt;&lt;InnerEapOptional&gt;false&lt;/InnerEapOptional&gt;&lt;Eap
xmlns="http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1"&gt;&lt;Type&gt;26&
lt;/Type&gt;&lt;EapType
xmlns="http://www.microsoft.com/provisioning/MsChapV2ConnectionPropertiesV1"&gt;&lt;UseWinLogo
nCredentials&gt;false&lt;/UseWinLogonCredentials&gt;&lt;/EapType&gt;&lt;/Eap&gt;&lt;EnableQuar
antineChecks&gt;false&lt;/EnableQuarantineChecks&gt;&lt;RequireCryptoBinding&gt;false&lt;/Requ
ireCryptoBinding&gt;&lt;PeapExtensions&gt;&lt;PerformServerValidation
xmlns="http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV2"&gt;true&lt;/Perform
ServerValidation&gt;&lt;AcceptServerName
xmlns="http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV2"&gt;true&lt;/AcceptS
erverName&gt;&lt;/PeapExtensions&gt;&lt;/EapType&gt;&lt;/Eap&gt;&lt;/Config&gt;&lt;/EapHostCon
fig&gt;
&lt;/EAPConfig&gt;&lt;/OneX&gt;&lt;/security&gt;&lt;/MSM&gt;&lt;/WLANProfile&gt;
    </Data>
  </Item>
</Replace>
```

The WLAN_Profile XML document for the previous examples.

```
<WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/profile/v1">
  <name>Test WiFi Profile</name>
  <SSIDConfig>
    <SSID>
      <name>12345</name>
    </SSID>
    <nonBroadcast>false</nonBroadcast>
  </SSIDConfig>
  <connectionType>ESS</connectionType>
  <connectionMode>manual</connectionMode>
  <autoSwitch>false</autoSwitch>
  <MSM>
    <security>
      <authEncryption>
        <authentication>WPA2</authentication>
        <encryption>AES</encryption>
        <useOneX>true</useOneX>
        <FIPSMode xmlns="http://www.microsoft.com/networking/WLAN/profile/v2">false</FIPSMode>
      </authEncryption>
      <PMKCacheMode>disabled</PMKCacheMode>
      <OneX xmlns="http://www.microsoft.com/networking/OneX/v1">
        <cacheUserData>false</cacheUserData>
        <authMode>machineOrUser</authMode>
        <EAPConfig>
          <EapHostConfig xmlns="http://www.microsoft.com/provisioning/EapHostConfig">
            <EapMethod>
              <Type xmlns="http://www.microsoft.com/provisioning/EapCommon">25</Type>
              <VendorId xmlns="http://www.microsoft.com/provisioning/EapCommon">0</VendorId>
              <VendorType
xmlns="http://www.microsoft.com/provisioning/EapCommon">0</VendorType>
              <AuthorId xmlns="http://www.microsoft.com/provisioning/EapCommon">0</AuthorId>
            </EapMethod>
            <Config xmlns="http://www.microsoft.com/provisioning/EapHostConfig">
              <Eap
xmlns="http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1">
                <Type>25</Type>
```

© 2014 Microsoft. All rights reserved.

```
                <EapType
xmlns="http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV1">
                  <ServerValidation>
<DisableUserPromptForServerValidation>false</DisableUserPromptForServerValidation>
                    <ServerNames></ServerNames>
                  </ServerValidation>
                  <FastReconnect>true</FastReconnect>
                  <InnerEapOptional>false</InnerEapOptional>
                  <Eap
xmlns="http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1">
                    <Type>26</Type>
                    <EapType
xmlns="http://www.microsoft.com/provisioning/MsChapV2ConnectionPropertiesV1">
                      <UseWinLogonCredentials>false</UseWinLogonCredentials>
                    </EapType>
                  </Eap>
                  <EnableQuarantineChecks>false</EnableQuarantineChecks>
                  <RequireCryptoBinding>false</RequireCryptoBinding>
                  <PeapExtensions>
                    <PerformServerValidation
xmlns="http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV2">true
                    </PerformServerValidation>
                    <AcceptServerName
xmlns="http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV2">true
                    </AcceptServerName>
                  </PeapExtensions>
                </EapType>
              </Eap>
            </Config>
          </EapHostConfig>
        </EAPConfig>
      </OneX>
    </security>
  </MSM>
</WLANProfile>
```

MSDN contains many WLAN_profile samples.

© 2014 Microsoft. All rights reserved.

The following is a sample WAP-Enterprise with TLS profile.

```
<WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/profile/v1">
    <name>SampleWPAEnterpriseTLS</name>
    <SSIDConfig>
        <SSID>
            <name>SampleWPAEnterpriseTLS</name>
        </SSID>
        <nonBroadcast>false</nonBroadcast>
    </SSIDConfig>
    <connectionType>ESS</connectionType>
    <connectionMode>auto</connectionMode>
    <autoSwitch>false</autoSwitch>
    <MSM>
        <security>
            <authEncryption>
                <authentication>WPA</authentication>
                <encryption>TKIP</encryption>
                <useOneX>true</useOneX>
            </authEncryption>
            <OneX xmlns="http://www.microsoft.com/networking/OneX/v1">
                <EAPConfig>
                    <EapHostConfig xmlns="http://www.microsoft.com/provisioning/EapHostConfig"

xmlns:eapCommon="http://www.microsoft.com/provisioning/EapCommon"

xmlns:baseEap="http://www.microsoft.com/provisioning/BaseEapMethodConfig">

                        <EapMethod>
                            <eapCommon:Type>13</eapCommon:Type>
                            <eapCommon:AuthorId>0</eapCommon:AuthorId>
                        </EapMethod>
                        <Config
xmlns:baseEap="http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1"

xmlns:eapTls="http://www.microsoft.com/provisioning/EapTlsConnectionPropertiesV1">

                            <baseEap:Eap>
                                <baseEap:Type>13</baseEap:Type>
                                <eapTls:EapType>
                                    <eapTls:CredentialsSource>
                                        <eapTls:CertificateStore />
                                    </eapTls:CredentialsSource>
                                    <eapTls:ServerValidation>

<eapTls:DisableUserPromptForServerValidation>false</eapTls:DisableUserPromptForServerValidatio
n>
                                        <eapTls:ServerNames />
                                    </eapTls:ServerValidation>
                                   <eapTls:DifferentUsername>false</eapTls:DifferentUsername>
                                </eapTls:EapType>
                            </baseEap:Eap>
                        </Config>
                    </EapHostConfig>
                </EAPConfig>
            </OneX>
        </security>
    </MSM>
</WLANProfile>
```

© 2014 Microsoft. All rights reserved.

A DM service can also delete a WIFI profile previously installed by the OMA-DM agent. The OMA-DM agent will not delete a profile it did not install. The following partial SyncML demonstrates deleting the WIFI profile used in the examples above.

```
<Delete>
  <CmdID>18</CmdID>
  <Item>
    <Target>
      <LocURI>
       ./cimv2/MDM_WirelessProfileXml/MDM_WirelessProfileXml.Name=%22Test%20WiFi%20Profilek%22
      </LocURI>
    </Target>
  </Item>
</Delete>
```

© 2014 Microsoft. All rights reserved.

DM VPN profile management

The OMA-DM agent supports installing VPN profiles onto the device. Support is for the embedded VPN providers only. At this time the following VPN vendors have embedded providers in Windows: Microsoft, F5, Sonicwall, Juniper and Checkpoint. VPN profiles are installed by targeting the remote WMI class MSFT_VpnConnection. For more information, see this article on MSDN: http://msdn.microsoft.com/library/jj206647(v=vs.85).aspx

```
[ClassVersion("1.0.0")]class MSFT_VpnConnection
{
  boolean AllUserConnection;
  string  Name;
  string  Profile;
};
```

The following partial SyncML demonstrates installing a basic VPN profile called “TestVPN”. The Data element contains a serialized VpnProfile object.

© 2014 Microsoft. All rights reserved.

```
<Exec>
  <CmdID>1</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MSFT_VpnConnection/MSFT_VpnConnection.Name=%22TestVPN%22,AllUserConnection=%220%22/Exe
c=Set
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>
Profile=&lt;VpnProfile&gt;&lt;VpnConfiguration&gt;&lt;Name&gt;TestVPN&lt;/Name&gt;&lt;AllUserC
onnection&gt;false&lt;/AllUserConnection&gt;&lt;ServerAddress&gt;1.2.3.4&lt;/ServerAddress&gt;
&lt;TunnelType&gt;Ikev2&lt;/TunnelType&gt;&lt;AuthenticationMethod&gt;&lt;Method&gt;Eap&lt;/Me
thod&gt;&lt;/AuthenticationMethod&gt;&lt;SplitTunneling&gt;false&lt;/SplitTunneling&gt;&lt;Rem
emberCredential&gt;false&lt;/RememberCredential&gt;&lt;UseWinlogonCredential&gt;false&lt;/UseW
inlogonCredential&gt;&lt;IdleDisconnectSeconds&gt;0&lt;/IdleDisconnectSeconds&gt;&lt;DnsSuffix
&gt;test.vpn.com&lt;/DnsSuffix&gt;&lt;VpnServerList&gt;&lt;ServerName&gt;1.2.3.4&lt;/ServerNam
e&gt;&lt;FriendlyName&gt;VPNServer&lt;/FriendlyName&gt;&lt;/VpnServerList&gt;&lt;VpnProxy&gt;&
lt;AutoDetect&gt;false&lt;/AutoDetect&gt;&lt;BypassProxyForLocal&gt;false&lt;/BypassProxyForLo
cal&gt;&lt;/VpnProxy&gt;&lt;/VpnConfiguration&gt;&lt;EapConfiguration&gt;&lt;EapHostConfig
xmlns="http://www.microsoft.com/provisioning/EapHostConfig"&gt;&lt;EapMethod&gt;&lt;Type
xmlns="http://www.microsoft.com/provisioning/EapCommon"&gt;26&lt;/Type&gt;&lt;VendorId
xmlns="http://www.microsoft.com/provisioning/EapCommon"&gt;0&lt;/VendorId&gt;&lt;VendorType
xmlns="http://www.microsoft.com/provisioning/EapCommon"&gt;0&lt;/VendorType&gt;&lt;AuthorId
xmlns="http://www.microsoft.com/provisioning/EapCommon"&gt;0&lt;/AuthorId&gt;&lt;/EapMethod&gt
;&lt;Config xmlns="http://www.microsoft.com/provisioning/EapHostConfig"&gt;&lt;Eap
xmlns="http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1"&gt;&lt;Type&gt;26&
lt;/Type&gt;&lt;EapType
xmlns="http://www.microsoft.com/provisioning/MsChapV2ConnectionPropertiesV1"&gt;&lt;UseWinLogo
nCredentials&gt;false&lt;/UseWinLogonCredentials&gt;&lt;/EapType&gt;&lt;/Eap&gt;&lt;/Config&gt
;&lt;/EapHostConfig&gt;
          &lt;/EapConfiguration&gt;&lt;/VpnProfile&gt;
    </Data>
  </Item>
</Exec>
```

The following example shows the un-serialized instance of the document above.

```
<VpnProfile>
  <VpnConfiguration>
    <Name>TestVPN</Name>
    <AllUserConnection>false</AllUserConnection>
    <ServerAddress>1.2.3.4</ServerAddress>
    <TunnelType>Ikev2</TunnelType>
    <AuthenticationMethod>
      <Method>Eap</Method>
    </AuthenticationMethod>
    <SplitTunneling>false</SplitTunneling>
    <RememberCredential>false</RememberCredential>
    <UseWinlogonCredential>false</UseWinlogonCredential>
    <IdleDisconnectSeconds>0</IdleDisconnectSeconds>
    <DnsSuffix>test.vpn.com</DnsSuffix>
    <VpnServerList>
      <ServerName>1.2.3.4</ServerName>
      <FriendlyName>VPNServer</FriendlyName>
    </VpnServerList>
    <VpnProxy>
```

© 2014 Microsoft. All rights reserved.

```
      <AutoDetect>false</AutoDetect>
      <BypassProxyForLocal>false</BypassProxyForLocal>
    </VpnProxy>
  </VpnConfiguration>
  <EapConfiguration>
    <EapHostConfig xmlns="http://www.microsoft.com/provisioning/EapHostConfig">
      <EapMethod>
        <Type xmlns="http://www.microsoft.com/provisioning/EapCommon">26</Type>
        <VendorId xmlns="http://www.microsoft.com/provisioning/EapCommon">0</VendorId>
        <VendorType xmlns="http://www.microsoft.com/provisioning/EapCommon">0</VendorType>
        <AuthorId xmlns="http://www.microsoft.com/provisioning/EapCommon">0</AuthorId>
      </EapMethod>
      <Config xmlns="http://www.microsoft.com/provisioning/EapHostConfig">
        <Eap xmlns="http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1">
          <Type>26</Type>
          <EapType
xmlns="http://www.microsoft.com/provisioning/MsChapV2ConnectionPropertiesV1">
            <UseWinLogonCredentials>false</UseWinLogonCredentials>
          </EapType>
        </Eap>
      </Config>
    </EapHostConfig>
  </EapConfiguration>
</VpnProfile>
```

Listed here are example VPN profile templates for the built-in VPN providers.

© 2014 Microsoft. All rights reserved.

F5

The following is an example VPN Profile template for F5. Note: Italicized values must be replaced with deployment specific values.

```
<VpnProfile>
  <VpnConfiguration>
    <Name>F5VPNConnectionName</Name>
    <ServerAddress>ServerFQDN</ServerAddress>
    <AutoTriggerEnabled>false</AutoTriggerEnabled>
    <SplitTunneling>true</SplitTunneling>
    <RememberCredential>false</RememberCredential>
    <IdleDisconnectSeconds>0</IdleDisconnectSeconds>
    <DnsSuffix />
    <ProvisioningAuthority />
    <VpnServerList />
    <Route/>
    <Trigger>
      <ClassicAppIDList>
        <ClassicAppID>C:\Windows\System32\mstsc.exe</ClassicAppID>
      </ClassicAppIDList>
      <ModernAppIDList>
        <ModernAppID>microsoft.bingfinance_8wekyb3d8bbwe</ModernAppID>
      </ModernAppIDList>
      <NrptRuleList>
        <DnsSuffix>.CorpFQDN</DnsSuffix>
        <DnsServer>DNSServerIP1</DnsServer>
        <DnsServer>DNSServerIP2</DnsServer>
        <DnsServer>DNSServerIP3</DnsServer>
      </NrptRuleList>
      <DnsSuffixSearchList>
        <DnsSuffix>.CorpFQDN</DnsSuffix>
      </DnsSuffixSearchList>
      <TrustedNetworkList>
        <TrustedNetwork>.CorpFQDN</TrustedNetwork>
      </TrustedNetworkList>
    </Trigger>
    <VpnProxy />
  </VpnConfiguration>
  <ThirdPartyParams>
    <PluginAppID>f5.vpn.client_cw5n1h2txyewy</PluginAppID>
    <CustomConfiguration>
      <f5-vpn-conf>
        <single-sign-on-credential />
        <client-certificate>
          <issuer>CertIssuerRootCA</issuer>
        </client-certificate>
      </f5-vpn-conf>
    </CustomConfiguration>
  </ThirdPartyParams>
</VpnProfile>
```

© 2014 Microsoft. All rights reserved.

Juniper

The following is an example VPN Profile template for Juniper: Note: italicized values must be replaced with deployment specific values.

```
<VpnProfile>
  <VpnConfiguration>
    <Name>JuniperVPNConnectionName</Name>
    <ServerAddress>ServerFQDN</ServerAddress>
    <AutoTriggerEnabled>false</AutoTriggerEnabled>
    <SplitTunneling>true</SplitTunneling>
    <RememberCredential>false</RememberCredential>
    <IdleDisconnectSeconds>0</IdleDisconnectSeconds>
    <DnsSuffix />
    <ProvisioningAuthority />
    <VpnServerList />
    <Route/>
    <Trigger>
      <ClassicAppIDList>
        <ClassicAppID>C:\Windows\System32\mstsc.exe</ClassicAppID>
      </ClassicAppIDList>
      <ModernAppIDList>
        <ModernAppID>microsoft.bingfinance_8wekyb3d8bbwe</ModernAppID>
      </ModernAppIDList>
      <NrptRuleList>
        <DnsSuffix>.CorpFQDN</DnsSuffix>
        <DnsServer>DNSServerIP1</DnsServer>
        <DnsServer>DNSServerIP2</DnsServer>
        <DnsServer>DNSServerIP3</DnsServer>
      </NrptRuleList>
      <DnsSuffixSearchList>
        <DnsSuffix>.CorpFQDN</DnsSuffix>
      </DnsSuffixSearchList>
      <TrustedNetworkList>
        <TrustedNetwork>.CorpFQDN</TrustedNetwork>
      </TrustedNetworkList>
    </Trigger>
    <VpnProxy />
  </VpnConfiguration>
  <ThirdPartyParams>
    <PluginAppID>JuniperNetworks.JunosPulseVpn_cw5n1h2txyewy</PluginAppID>
    <CustomConfiguration>
      <pulse-schema>
       <isSingleSignOnCredential>true</isSingleSignOnCredential>
      </pulse-schema>
    </CustomConfiguration>
  </ThirdPartyParams>
</VpnProfile>
```

© 2014 Microsoft. All rights reserved.

Checkpoint

The following is an example VPN Profile template for Checkpoint: Note: italicized values must be replaced with deployment specific values.

```
<VpnProfile>
  <VpnConfiguration>
    <Name>checkpointVPNConnectionName</Name>
    <ServerAddress>ServerFQDN</ServerAddress>
    <AutoTriggerEnabled>false</AutoTriggerEnabled>
    <SplitTunneling>true</SplitTunneling>
    <RememberCredential>false</RememberCredential>
    <IdleDisconnectSeconds>0</IdleDisconnectSeconds>
    <DnsSuffix />
    <ProvisioningAuthority />
    <VpnServerList />
    <Route />
    <Trigger>
      <ClassicAppIDList>
        <ClassicAppID>C:\Windows\System32\mstsc.exe</ClassicAppID>
      </ClassicAppIDList>
      <ModernAppIDList>
        <ModernAppID>microsoft.bingfinance_8wekyb3d8bbwe</ModernAppID>
      </ModernAppIDList>
      <NrptRuleList>
        <DnsSuffix>.CorpFQDN</DnsSuffix>
        <DnsServer>DNSServerIP1</DnsServer>
        <DnsServer>DNSServerIP2</DnsServer>
        <DnsServer>DNSServerIP3</DnsServer>
      </NrptRuleList>
      <DnsSuffixSearchList>
        <DnsSuffix>.CorpFQDN </DnsSuffix>
      </DnsSuffixSearchList>
      <TrustedNetworkList>
        <TrustedNetwork>.CorpFQDN</TrustedNetwork>
      </TrustedNetworkList>
    </Trigger>
    <VpnProxy />
  </VpnConfiguration>
  <ThirdPartyParams>
    <PluginAppID>CheckPoint.VPN_cw5n1h2txyewy</PluginAppID>
    <CustomConfiguration>
      <CheckPointVPN port="443" name="checkpoint.Inbox.Selfhost" debug="2" timeout="30" cn=""
fingerprint="" auth="t" regkey="" p12file="" p12pass="" sso="true" lowcost="false" />
    </CustomConfiguration>
  </ThirdPartyParams>
</VpnProfile>
```

© 2014 Microsoft. All rights reserved.

SonicWall

The following is an example VPN Profile template for SonicWall: Note: italicized values must be replaced with deployment specific values.

```
<VpnProfile>
  <VpnConfiguration>
    <Name>SonicWallVPNConnectionName</Name>
    <ServerAddress>ServerFQDN</ServerAddress>
    <AutoTriggerEnabled>false</AutoTriggerEnabled>
    <SplitTunneling>true</SplitTunneling>
    <RememberCredential>false</RememberCredential>
    <IdleDisconnectSeconds>0</IdleDisconnectSeconds>
    <DnsSuffix />
    <ProvisioningAuthority />
    <VpnServerList />
    <Route />
    <Trigger>
      <ClassicAppIDList>
        <ClassicAppID>C:\Windows\System32\mstsc.exe</ClassicAppID>
      </ClassicAppIDList>
      <ModernAppIDList>
        <ModernAppID>microsoft.bingfinance_8wekyb3d8bbwe</ModernAppID>
      </ModernAppIDList>
      <NrptRuleList>
        <DnsSuffix>.CorpFQDN</DnsSuffix>
        <DnsServer>DNSServerIP1</DnsServer>
        <DnsServer>DNSServerIP2</DnsServer>
        <DnsServer>DNSServerIP3</DnsServer>
      </NrptRuleList>
      <DnsSuffixSearchList>
        <DnsSuffix>.CorpFQDN</DnsSuffix>
      </DnsSuffixSearchList>
      <TrustedNetworkList>
        <TrustedNetwork>corp.microsoft.com</TrustedNetwork>
      </TrustedNetworkList>
    </Trigger>
    <VpnProxy />
  </VpnConfiguration>
  <ThirdPartyParams>
    <PluginAppID>SonicWALL.MobileConnect_cw5n1h2txyewy</PluginAppID>
    <CustomConfiguration>
      <MobileConnect>
        <Compression>false</Compression>
        <SmartCardRequired>true</SmartCardRequired>
        <ClientCertAutoSelect>true</ClientCertAutoSelect>
        <debugLogging>True</debugLogging>
        <packetCapture>False</packetCapture>
      </MobileConnect>
    </CustomConfiguration>
  </ThirdPartyParams>
</VpnProfile>
```

© 2014 Microsoft. All rights reserved.

A DM service can also delete a VPN profile previously installed by the OMA-DM agent. The OMA-DM agent will not delete a profile it did not install. The following partial SyncML demonstrates deleting the VPN profile used in the examples above.

```
<Delete>
  <CmdID>28</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MSFT_VpnConnection/MSFT_VpnConnection.Name=%22TestVPN%22
      </LocURI>
    </Target>
  </Item>
</Delete>
```

Application Triggered VPN

A DM service can trigger a VPN connection(s) to automatically “connect” when a configured modern application is started. Once connected, the VPN connection is global to the device. The OMA-DM agent does not support associating a VPN connection to a specific application or container. If multiple VPN profiles have been installed on the device through device management, they ALL will connect when a “trigger” application is started.

© 2014 Microsoft. All rights reserved.

The following remote WMI class MDM_VpnApplicationTrigger is the target for adding/removing applications that will trigger a VPN connection.

```
[Description("This class provides the ability to add an application as a trigger to MDM
installed VPN profiles")]
class MDM_VpnApplicationTrigger
{
    [key, Description("Package family name of the Application")]
    string ApplicationID;

    [Description("Is application trigger added to all MDM installed VPN profiles")]
    boolean           TriggerEnabledInAllMDMProfiles;
};
```

The following partial SyncML demonstrates adding a modern application trigger.

```
<Add>
  <CmdID>30</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_VpnApplicationTrigger.ApplicationID=%22contoso.companyapp_8wekyb3d8bbwe%22
      </LocURI>
    </Target>
  </Item>
</Add>
```

Deleting the previous application trigger is accomplished with the following partial SyncML.

```
<Delete>
  <CmdID>30</CmdID>
  <Item>
    <Target>
      <LocURI>
        ./cimv2/MDM_VpnApplicationTrigger.ApplicationID=%22contoso.companyapp_8wekyb3d8bbwe%22
      </LocURI>
    </Target>
  </Item>
</Delete>
```

© 2014 Microsoft. All rights reserved.

DM device actions

The OMA-DM agent supports several device actions. An action can force the device into a state such as “locked” or un-enroll the device from management.

The DM service will target the remote WMI class MDM_Client for most of the device actions.

```
[Description("This class provides identification for a user/device pair. It also provides
methods for unenrolling the device from management, locking the device and resetting a
password")]
class MDM_Client
{
  [Key,Description("The key to identify the instance of MDM_Client class, This is a device
id")]
  string DeviceClientID;

  [Description("This property contains the local workstation SID")]
  string DomainSID;

  [Description("This property contains the textual name of the local Operating System")]
  string PlatformID;

  [Description("This property contains the FQDN of the device")]
  string DeviceName;

  [Description("This property contains a textual description of the main processor on the
device")]
  string ProcessorDescription;

  [Description("This property contains the SID of the local user account associated to the
device")]
  string UserSid;

  [Description("This property contains the local Operating System version in the following
format: major.minor.revision")]
  string Version;

  [static,Description("This method provides ability to unenroll the device")]
  uint32 SendUnenrollRequest([In,Description("Id of the device")] string DeviceClientID);

  [static,Description("This method provides ability to lock the device")]
  uint32 LockWorkstation();

  [static,Description("This method provides ability to reset a user’s password with a known
password")]
  uint32 ResetUserPassword([In,Description("Userupn password pair. E.g. x@y.com;pwd1")] string
ConfigString);
};
```

© 2014 Microsoft. All rights reserved.

The DM service needs the DeviceID in order to send device actions to the device through the remote WMI class MDM_Client. The partial SyncML example demonstrates how to retrieve the DeviceID using a Get command.

```
<Get>
  <CmdID>6</CmdID>
  <Item>
    <Target>
      <LocURI>./cimv2/MDM_Client</LocURI>
    </Target>
  </Item>
</Get>
```

The client will respond to the Get Command on the MDM_Client class with Status and Results commands similar to the following:

```
<Status>
  <CmdID>2/CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>6</CmdRef>
  <Cmd>Get</Cmd>
  <Data>200</Data>
</Status>
<Results>
  <CmdID>3</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>6</CmdRef>
  <Item>
    <Source>
      <LocURI>./cimv2/MDM_Client</LocURI>
    </Source>
    <Meta>
      <Format xmlns="syncml:metinf">node</Format>
    </Meta>
    <Data>MDM_Client.DeviceID="e49e0231-67bf-4161-b69f-cb5928f63bff"</Data>
  </Item>
</Results>
```

Device lock

The OMA-DM agent supports device lock if the device was enrolled into management using the standard enrollment protocol. Device lock is not supported if the device was enrolled using the “enroll on behalf of” protocol.

© 2014 Microsoft. All rights reserved.

The following partial SyncML example demonstrates remote locking a device.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MDM_Client/MDM_Client.DeviceClientID=%22e49e0231-67bf-4161-b69f-cb5928f63bff
%22/Exec=LockWorkstation
      </LocURI>
    </Target>
  </Item>
</Exec>
```

The OMA-DM agent will respond with a Status command which the DM service can use to report status on the lock.

Un-enrollment

If the device was enrolled into management using the standard enrollment protocol, the end user can un-enroll the device by navigating to the “Workplace” modern setting and tapping the “turn off management” button. Unfortunately, the current version of the OMA-DM agent does not send an Alert command to the DM service declaring the user has un-enrolled the device.

A device can also be un-enrolled by the managing DM service. The following partial SyncML example demonstrates un-enrolling the device through a remote OMA command.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MDM_Client/MDM_Client.DeviceClientID=%22e49e0231-67bf-4161-b69f-cb5928f63bff
%22/Exec=SendUnenrollRequest
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>
      DeviceClientId=22e49e0231-67bf-4161-b69f-cb5928f63bff
    </Data>
  </Item>
</Exec>
```

The OMA-DM agent will respond with a Status command which the DM service can use to report status on the the device un-enrollment.

The MDM Server can issue the Un-Enrollment and Password reset commands even in the Maintenance mode of operation.

© 2014 Microsoft. All rights reserved.

Password reset

The OMA-DM agent supports a help desk aided password reset for devices that were enrolled using the EOBO protocol. The DM service can reset the managed user's password to a password communicated out-of-band by an IT manager/Help Desk to the managed user on the device. When the user logs on using the new password, they will be immediately be forced into a change password state. Password Reset is not supported when the device was enrolled using the standard end user method.

The following partial SyncML example demonstrates sending a password reset command to the device through a remote OMA command.

```
<Exec>
  <CmdID>{unique command id in message}</CmdID>
  <Item>
    <Target>
      <LocURI>
./cimv2/MDM_Client/MDM_Client.DeviceClientID=%22e49e0231-67bf-4161-b69f-
cb5928f63bff%22/Exec=ResetUserPassword
      </LocURI>
    </Target>
    <Meta>
      <Format xmlns="syncml:metinf">chr</Format>
      <Type xmlns="syncml:metinf">text/plain</Type>
    </Meta>
    <Data>
      ConfigString={User UPN;New UserPassword}
    </Data>
  </Item>
</Exec>
```

A ConfigString example: joe@contoso.com;n3wpassw0rd1

The OMA-DM agent will respond with a Status command which the DM service can use to report status on the reset password command.

The MDM Server can issue the Un-Enrollment and Password reset commands even in the Maintenance mode of operation.

© 2014 Microsoft. All rights reserved.

Agent logging

For capturing and viewing logs generated by the built-in management agent and its components you will need access to the traceview.exe utility and the debug symbol files (.pdb) for the Windows build running on the device.

Traceview.exe can be downloaded as part of the Windows Driver Kit (WDK). The symbols files can be requested from Microsoft if needed.

You can enable advanced logging by adding the following keyvalue to the registry.

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\MDM\Debug Settings]
"EnableSyncmlLogging"=dword:00000001
Start the traceview.exe utility from an elevated command prompt and add the following providers:

- mdmagent.pdb
- mdmappprov.pdb
- mdmsettingsprov.pdb

Start the log capture and execute the scenario.

© 2014 Microsoft. All rights reserved.

Reference

- [MS-WSTEP]: WS-Trust X.509v3 Token Enrollment Extensions (http://msdn.microsoft.com/library/dd340609.aspx)
- OMA Device Management Protocol v1.2 (http://technical.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DM&file=V1_2_1-20080617-A/OMA-TS-DM_Protocol-V1_2_1-20080617-A.pdf)
- OMA Device Management Security (http://technical.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DM&file=V1_2_1-20080617-A/OMA-TS-DM_Security-V1_2_1-20080617-A.pdf)
- OMA DM Standardized Objects (http://technical.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DM&file=V1_2_1-20080617-A/OMA-TS-DM_StdObj-V1_2_1-20080617-A.pdf)
- OMA DM Representation protocol (http://technical.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DM&file=V1_2_1-20080617-A/OMA-TS-DM_RepPro-V1_2_1-20080617-A.pdf)
- OMA DM Tree and Description (http://technical.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DM&file=V1_2_1-20080617-A/OMA-TS-DM_TNDS-V1_2-20070209-A.pdf)
- OMA DM Bootstrap (http://technical.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DM&file=V1_2_1-20080617-A/OMA-TS-DM_Bootstrap-V1_2_1-20080617-A.pdf)
- Application Characteristic for OMA Device Management (http://technical.openmobilealliance.org/Technical/release_program/docs/copyrightclick.aspx?pck=DM&file=V1_2_1-20080617-A/OMA-SUP-ac_w7_dm-V1_0_1-20080617-A.txt)
- [SCEP]: Simple Certificate Enrollment Protocol (http://tools.ietf.org/html/draft-nourse-scep-23)

Q&A

This section lists the common questions MDM vendors may have and corresponding answers.

Question	Answer
Can enrollment be initiated via SMS?	This is not supported in Windows 8.1.
Can the DM client support using a proxy to make connections to the MDM server for authentication and check-in?	HTTP proxy over Wi-Fi is supported. Proxy authentication is not supported in Windows 8.1.

© 2014 Microsoft. All rights reserved.

Question	Answer
Do you support SSL offloading?	This is not supported in Windows 8.1.
Will the discovery request accept self-signed certs or prompt the user?	No, the discovery request does not support self-signed certs in Windows 8.1.
Can we have our own client instead of going through the UI provided by OS?	This is not supported in Windows 8.1.
Can a device be enrolled with multiple companies? Can a user create multiple "company apps" accounts?	In Windows 8.1, we support one company account per device for the enterprise device management server.
Can a device be owned by multiple users?	One device could be used by multiple users. However, at any one time, there can be only one managed account. The user needs to turn off management for the old account before adding a new account.
Unenrollment/disassociation seems to happen during a normal maintenance session - can an unenrollment command be pushed to the device?	Yes. Push notification will wake up the OMA-DM agent to check-in with the management server. The management server will then tell the device to un-enroll from management.
Are the enterprise apps installed through the MDM server removed when the device is unenrolled?	Yes.
Can you be prevented from unenrolling or unregistering?	Yes. If the user is a low rights user and enrolled using the EOBO protocol.
What kinds of errors are reported by the device while the MDM server configures the device?	During a DM session, the phone reports various error codes depending on which DM command is sent by the server. For information about common errors the device could send, see the OMA DM representation doc. Most are very straightforward.
Is there any way we can send user-facing messages to the user?	Not at this time.
Is there international roaming detection support or notification of the current carrier and country/region (MCC/MNC)?	This is not supported in Windows 8.1
Can the MDM server get the location of the device from the MDM agent?	This is not supported in Windows 8.1.

© 2014 Microsoft. All rights reserved.

Question	Answer
When a certificate expires, the client can be offline or the device can be turned off. Does letting the certificate expire require the user to remove the MDM relationship (along with all the apps and settings)?	When the certificate expires, the relationship is not removed, but the user cannot launch the installed enterprise app and the agent cannot communicate with the server (client authentication will fail) until the certificate is renewed.
Is it possible to get an application inventory of the entire device or just enterprise-deployed apps?	The DM service can only inventory enterprise-deployed apps.
Is it possible to get logs for enrollment or an DM session through either tethering or over the air?	This is not supported in Windows 8.1.
Can the DM interval/synch policy/certificate renewal period be changed after enrollment?	These are only set during enrollment.
Is there an alert outside "company apps" settings to warn the user about required renewal?	Yes. Before the certificate expires, the device will prompt the user to go to the settings page to provide an updated password when the company apps account is about to expire.
What happens to the settings or apps installed through the DM client after the certificate is expired?	Apps will remain on the device but fail to launch. Policies specified by the DM service will remain as valid. Re-installing a valid cert will re-enable the apps.
Is there a unique identifier that could be retrieved by both company applications and the DM server?	No. Modern applications cannot discover the unique device ID on the device.

© 2014 Microsoft. All rights reserved.

Support

If you still have questions after reading through this white paper, please post your questions to the Developing MDM Solutions support forum, created exclusively for MDM development discussions.

For paid development support, please submit a support ticket. Under **Problem type**, choose **Developing MDM solutions,** and then choose the correct category for your specific issue.

© 2014 Microsoft. All rights reserved.