[MS-WUSP]:
Windows Update Services:
Client-Server Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
03/14/2007	1.0		Version 1.0 release
04/10/2007	1.1		Version 1.1 release
05/18/2007	1.2		Version 1.2 release
06/08/2007	1.2.1	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
07/10/2007	2.0	Major	Changed to unified format; updated technical content.
08/17/2007	3.0	Major	Updated and revised the technical content.
09/21/2007	4.0	Major	Updated and revised the technical content.
10/26/2007	4.0.1	Editorial	Revised and edited the technical content.
01/25/2008	4.0.2	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

The Windows Update Services: Client-Server Protocol enables machines to discover and download software updates over the Internet by using the **SOAP** and HTTP protocols (as specified in [SOAP1.1], [SOAP1.2/1], [SOAP1.2/2], and [RFC2616], respectively). This protocol is implemented by all Windows releases from Windows 2000 SP3.

This document includes the following:

- How messages are encapsulated on the wire and message syntax in section 2.
- Protocol details including abstract data models and message processing rules in section 3.
- Protocol examples in section 4.
- Security considerations for implementers in section 5.
- An appendix of Windows behavior in section 6.
- An index in section 7.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Globally Unique Identifier (GUID)
UncPath
Universal Naming Convention (UNC)

The following terms are specific to this document:

Client Computer: A computer that gets its **updates** from an **update server**. A client can be a desktop computer, a server, or the **update server**.

Client Web Service: A **Web service** on the **update server** that enables clients to obtain **update metadata**.

Component Based Servicing (CBS): A file format containing information used by Windows Vista to install operating system components.

Conjunctive Normal Form (CNF): A logical formula consisting of a conjunction of disjunctions of terms in which no disjunction contains a conjunction. For example, A OR (B AND C) is not in **CNF**, whereas the equivalent (A OR B) AND (A OR C) is in **CNF**.

Content: A package consisting of all files associated with an **update** that is to be installed on a **client computer**.

Content Directory: A location on the **update server** identified by an HTTP URL that is used to expose update **content** to clients.

Deployment: An administratively-specified decision to make a specific **update revision** available to a specific **target group**.

Locale: A set of language-related preferences that influence the presentation of user interface elements, such as human-readable strings.

Man-in-the-Middle: A computer security attack in which an attacker is able to read, insert, and modify at will messages between two parties without either party knowing that the link between them is compromised. The attacker must be able to observe and intercept messages going between the two parties.

Metadata: XML-formatted data containing information on an **update**.

Microsoft Windows Installer (MSI): A file format containing information used by the Windows Installer to install software and software updates. For more information, see [MSI].

Prerequisite: A relationship from a **revision** to a set of updates specified in **conjunctive normal form (CNF)**. For example, (U6 | U8) & (U2) & (U5 | U3). A client will not treat a **revision** as requiring installation unless its **prerequisites** are satisfied (that is, at least one **update** in each **CNF** disjunctive clause is installed on the client).

Prerequisite Graph: A directed graph with **revisions** as vertices and **prerequisite** relationships as edges.

Reporting Web Service: A **Web service** used by clients to report status to the server.

Revision: A specific version of an **update** identified by a combination of a **globally unique identifier (GUID)** UpdateID and a 32-bit RevisionNumber.

Revision ID: A compact server-assigned, 32-bit identifier for a **revision** that is used to identify the **revision** during client/server communication.

Self-Update: A process by which a client first communicates with the **update server** to detect **updates** to the executable files that implement the client role on computers running Windows, and then applies those updated executable files before carrying on further communication.

Self-Update Content: A form of **update content** comprised of executable files that are to be installed by clients during the **self-update** process.

Self-Update Tree: A **virtual directory** on the server that exposes **self-update content**. The client automatically checks for and applies **self-update content** early in the client/server communications.

SimpleAuth Web Service: A **Web service** on the server that is used to authorize what clients should get **metadata** for what **revisions**.

Simple Object Access Protocol (SOAP): An XML-based protocol for exchanging information in distributed systems, as specified in [SOAP1.1].

Target Group: A named collection of **client computers** whose members are defined administratively.

UncPath: The location of a file in a network of computers, as specified in **Universal Naming Convention (UNC)** syntax.

Universal Naming Convention (UNC): A standard naming format for specifying the location of network resources such as shared files or devices on a network. The format is "\\<servername>\<share>\<filename>" where <servername> is a NetBIOS name, FQDN domain name, or IPv4 address; <share> is a logical share point for accessing <servername>, and <filename> is the name of the file or device.

Update: The combination of **metadata** and associated **content** for a software update.

Update Server: A computer that implements the Windows Update Services: Client-Server Protocol to provide **updates** to **client computers** and other **update servers**.

Virtual Directory: An HTTP URL representing the root of a location to which **content** may be published administratively.

Web Service: A software entity that responds to **Simple Object Access Protocol (SOAP)** messages, as specified in [SOAP1.1] and [WSDL].

Web Service Definition Language (WSDL): An XML-based standard for specifying message-based distributed services, as specified in [WSDL].

Windows Server Update Services (WSUS): An optional component of Windows 2000 Server and Windows Server 2003 that may be installed to enable a computer to operate as an **update server**. See [WSUS].

Windows Update Agent (WUA): A component of Windows 2000 Server SP3 and later operating systems that enables a computer to operate as a client of an **update server**.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as specified in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-DRSR] Microsoft Corporation, "Directory Replication Service (DRS) Remote Protocol Specification", July 2006.

[MS-ERREF] Microsoft Corporation, "Windows Error Codes", January 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-GPOL] Microsoft Corporation, "Group Policy: Core Protocol Specification", July 2006.

[MS-LCID] Microsoft Corporation, "Windows Language Code Identifier (LCID) Reference", March 2007.

[MS-SECO] Microsoft Corporation, "Windows Security Overview", December 2006.

[MS-WSUSSS] Microsoft Corporation, "Windows Update Services: Server-Server Protocol Specification", July 2006.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D., "Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[SOAP1.2/1] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624

[SOAP1.2/2] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[XPATH] Clark, J. and DeRose, S., "XML Path Language (XPath), Version 1.0", W3C Recommendation, November 1999, http://www.w3.org/TR/xpath

1.2.2 Informative References

[AUPOLICY] Microsoft Corporation, "Configure Automatic Updates by Using Group Policy", http://technet2.microsoft.com/WindowsServer/en/Library/51c8a814-6665-4d50-a0d8-2ae27e69ca7c1033.mspx

[MSI] Microsoft Corporation, "Windows Installer 3.0 Redistributable", http://www.microsoft.com/downloads/details.aspx?amp;displaylang=en&familyid=5fbc 5470-b259-4733-a914-a956122e08e8&displaylang=en

[WSUS] Microsoft Corporation, "Windows Server Update Services", http://www.microsoft.com/windowsserversystem/updateservices/default.mspx

1.3 Protocol Overview (Synopsis)

The Windows Update Services family of protocols provides support for central publication and distribution of software updates from server machines to client machines, and for hierarchical synchronization of available software components between servers.

This specification defines the Windows Update Services: Client-Server Protocol, which enables client machines to determine available, applicable software updates, and to download those updates for installation.

The Windows Update Services: Client-Server Protocol is a SOAP-based protocol that uses HTTP 1.1 as its transport. The protocol includes three distinct phases.

1. **Self-Update:** The client consults the server to determine if updated executable files are available for the client implementation of the Windows Update Services: Client-Server Protocol. If so, the client updates itself to operate using the updated executable files before continuing to communicate with the server.

2. **Metadata Synchronization:** The client synchronizes **update metadata** from the **update server** by calling a sequence of **Web service** methods, as specified in section 3.1.5. The metadata describes various characteristics of the update including its title, description, rules for determining if the update is applicable to a computer, and instructions for installing the update **content**.

- To reduce network overhead and increase performance, the protocol facilitates the caching of update metadata on clients.
- To further reduce the amount of update metadata that clients must synchronize, update metadata is divided into fragments. Each client synchronizes only the fragments that it needs. In particular:
  - The client invokes the **SyncUpdates (section 2.2.3.4)** method, which returns to the client a "core" fragment. This fragment contains sufficient update metadata for a client to evaluate if the update content is required.
  - If the client determines that update content is required, it then invokes the **GetExtendedUpdateInfo (section 2.2.3.7)** method to obtain additional metadata fragments.

3. **Content Synchronization:** Finally, the client may request update content comprised of any files associated with the updates required by the client.
4. **Reporting:** The client reports events to the server that provide information on its update-related activities (for example, content download succeeded or failed, content install succeeded or failed). Reports are generated asynchronously from the rest of the protocol.

A UML sequence diagram can be found in section 3.1.1.

This specification details the protocol mechanisms that enable clients to download **self-update** binaries, synchronize update metadata, and download update content. It also details the protocol mechanisms for enabling clients to report events to servers.<1>

1.4 Relationship to Other Protocols

The reporting and metadata synchronization protocols include Web services that use SOAP (as specified in [SOAP1.1]) over HTTP or HTTPS (as specified in [RFC2616]) for communication. The self-update and content synchronization protocols use HTTP 1.1 (as specified in [RFC2616]).

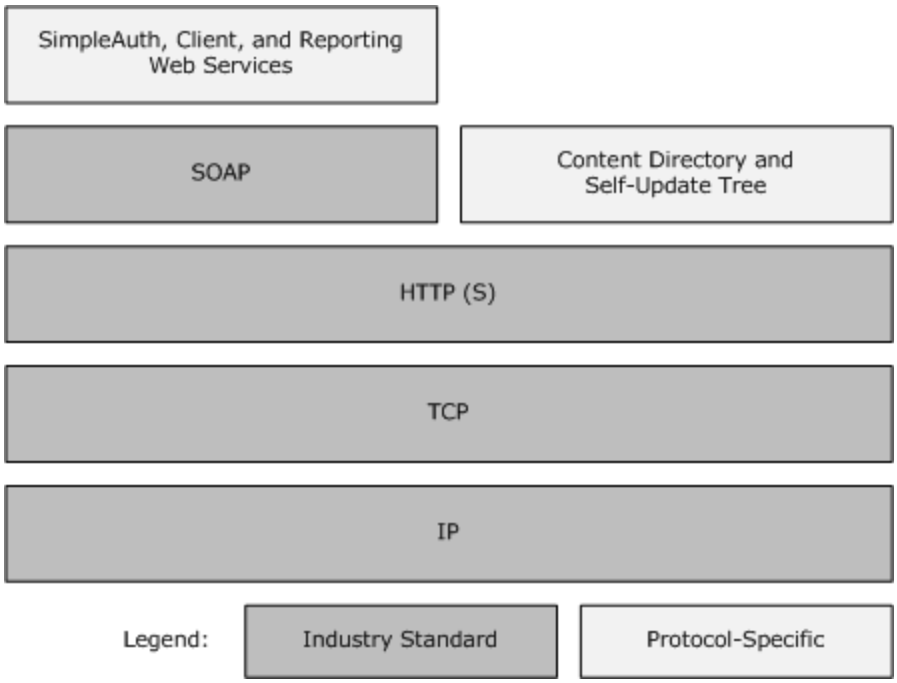

Figure 1: Relationship between protocols related to Windows Update Services: Client-Server Protocol

Content download (both update content and **self-update content**) uses HTTP 1.1 HEAD and GET (Range) requests (as specified in [RFC2616] sections 9.3 and 9.4).

This specification is closely related to the Windows Update Services: Server-Server Protocol, as specified in [MS-WSUSSS], which defines mechanisms for synchronizing updates within a hierarchical configuration of update servers.

1.5 Prerequisites/Preconditions

The Windows Update Services: Client-Server Protocol imposes the following requirements on server implementations.

- This document specifies how the binaries and metadata are distributed using the Client-Server Communications Protocol. It does not specify the format of the binaries or metadata themselves, but it assumes that the metadata is well-formed XML and is compatible with the XPATH queries specified in section 3.1.1.1 for populating the server data model. In all other respects, the binaries and metadata are treated as opaque by the server.

The Windows Update Services: Client-Server Protocol imposes the following requirements on client implementations.

- Clients must be configured to obtain updates from the server. This configuration MAY be performed manually or, in managed environments, MAY be performed using any appropriate form of centralized machine configuration.<2>

1.6 Applicability Statement

The Windows Update Services: Client-Server Protocol is applicable in environments where there is a need for centralized, systematic distribution of software updates to managed **client computers**.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas.

- **Protocol versions:** There are three versions of the protocol specified in this documentation: versions 3.0, 1.0, and 0.9. These versions differ only in the ways specified in section 2.1.
- **Supported transports:** Both versions of the protocol use HTTP and SOAP for communications. The versions differ only in the ways specified in section 2.1.
- **Capability negotiation:** As specified in section 1.3, clients initiate communication by obtaining updated executable files, which implement the most recent protocol behavior required by the server. This process is specified in section 2.1 and eliminates any need for explicit capability negotiation.

1.8 Vendor-Extensible Fields

The Windows Update Services: Client-Server Protocol does not define any vendor-extensible fields.

1.9 Standards Assignments

The following standard XML namespaces are used in this protocol.

http://schemas.xmlsoap.org/wsdl/http/

http://www.w3.org/2001/XMLSchema

http://schemas.xmlsoap.org/wsdl/soap/

http://schemas.xmlsoap.org/wsdl/soap12/

http://schemas.xmlsoap.org/soap/encoding/

http://schemas.xmlsoap.org/wsdl/

2 Messages

The Windows Update Services: Client-Server Protocol MUST be carried out over SOAP (as specified in [SOAP1.1]) and HTTP (as specified in [RFC2616]) and consists of the following set of Web services and **virtual directories**.

- **Self-update tree**: A virtual directory containing the client self-update binaries, as specified in section 2.2.6.
- **Content directory**: A virtual directory containing content, as specified in section 2.2.6.
- **SimpleAuth Web service**: A Web service that clients consult to obtain cached state for use by servers in restricting availability of updates to groups of clients, as specified in section 2.2.2.
- **Windows Update Agent (WUA)**: A Web service that synchronizes metadata to the client, as specified in section 2.2.3.
- **Reporting Web service**: A Web service that clients contact to report selected events containing information on their update activity, as specified in section 2.2.4.

The following sections specify the use of the transports listed above and the syntax of these Web services.

2.1 Transport

The Windows Update Services: Client-Server Protocol is carried out over a set of Web services and virtual directories.

- Each Web service MUST support Simple Object Access Protocol (SOAP) (as specified in [SOAP1.1]) over HTTP (as specified in [RFC2616]) over TCP/IP. Each Web service SHOULD support HTTPS for securing its communication with clients.<3>
- Each virtual directory MUST support HTTP (as specified in [RFC2616]) over TCP/IP. Each virtual directory SHOULD NOT require the use of HTTPS to maximize server performance.<4>

The following TCP ports MUST be exposed by the server as endpoints for the HTTP over TCP/IP transport.

- **commonPort:** Used for self-update and Web services communication.<5>
- **contentPort:** Used by the virtual directory that contains content.<6>

The following virtual directories MUST be exposed by the server as endpoints for the HTTP and SOAP over HTTP transports.

- **Self-update tree**: This virtual directory, as specified in section 2.2.6, MUST be exposed at URL http://serverUrl:[commonPort]/SelfUpdate
- **Content directory**: This virtual directory, as specified in section 2.2.6, MUST be exposed at URL http://serverUrl:[contentPort]/Content
- **SimpleAuth Web service**: This virtual directory, as specified in section 2.2.2, MUST be exposed at URL http[s]://serverUrl:[commonPort]/SimpleAuthWebService/SimpleAuth.asmx
- **Client Web service**: This virtual directory, as specified in section 2.2.3, MUST be exposed at URL http[s]://serverUrl:[commonPort]/ClientWebService/Client.asmx

- **Reporting Web service**: This virtual directory, as specified in section 2.2.4, MUST be exposed at URL http[s]://serverUrl:[commonPort]/ReportingWebService/ReportingWebService.asmx

Windows Update Services: Client-Server Protocol version 0.9 is a legacy version of the protocol that requires the following:

- Requires that commonPort be configured to port 80.
- Requires two special self-update files, Sus20Version.xml and iuident.cab, to be accessible in the virtual directory at URL http://serverUrl:[commonPort]. The **self-update tree** is specified in section 2.2.6.

To allow older versions of Windows to self-update from a server, the server SHOULD also expose the self-update tree on port 80 to the self-update portion of the legacy protocol.<7>

To optimize network bandwidth, the client implementation MAY request that the reply be compressed by specifying the encoding format in the HTTP Accept-Encoding request-header field (as specified in [RFC2616] section 14.3). The update server SHOULD encode the reply using the requested format.<8>

2.2 Message Syntax

This section specifies the syntax of SOAP messages, which are part of the Windows Update Services: Client-Server Protocol. The following rules apply to all SOAP messages in the protocol.

- The <soap:header> element (as specified in [SOAP1.1] section 4.2 and [SOAP1.2/1] section 5.2) MUST NOT be used.
- The <soap:binding> element of the **Web Service Definition Language (WSDL)** MUST specify style="document", as specified in [WSDL] section 3.3.
- The <soap:body> element of the WSDL MUST specify use="literal", as specified in [WSDL] section 3.5.

In the sections that follow, excerpts are given from the WSDL file for the Windows implementation of the Windows Update Services: Client-Server Protocol. In the WSDL file, the attributes **minOccurs** and **maxOccurs** denote the number of elements allowed, so that zero is ≤ **minOccurs** ≤ **maxOccurs**, as specified in [WSDL].

For specific WSDL elements, the protocol specifies additional restrictions beyond those specified by the WSDL syntax of the elements. For instance, in some cases, the protocol always requires the presence of an element in a message, even though its WSDL specification has a **minOccurs** attribute set to 0. In other cases, the protocol requires stronger typing on elements than is specified by the WSDL for the elements.

In all such cases, the additional restrictions are specified immediately after the WSDL is given.

2.2.1 Common Data Types

The following table shows the standard XML namespaces used within the Windows Update Services: Client-Server Protocol and the alias (prefix) used in the remaining sections of this protocol specification.

Alias (prefix)	XML namespace
http	http://schemas.xmlsoap.org/wsdl/http/

Alias (prefix)	XML namespace
s	http://www.w3.org/2001/XMLSchema
soap	http://schemas.xmlsoap.org/wsdl/soap/
soap12	http://schemas.xmlsoap.org/wsdl/soap12/
soapenc	http://schemas.xmlsoap.org/soap/encoding/
wsdl	http://schemas.xmlsoap.org/wsdl/

The following table shows the Microsoft-defined XML namespaces used within the Windows Update Services: Client-Server Protocol and the alias (prefix) used in the remaining sections of this protocol specification.

Alias (prefix)	XML namespace
s2	http://microsoft.com/wsdl/types/
s1	The target namespace. The XML namespace it refers to depends on the WSDL file it is used in. For **SimpleAuth Web Service** WSDL, it is http://www.microsoft.com/SoftwareDistribution/Server/SimpleAuthWebService. For **Client Web Service** WSDL, it is http://www.microsoft.com/SoftwareDistribution/Server/ClientWebService.

The following sections define the common data types that are used in this protocol.

2.2.1.1 ArrayOfInt

An array of integer values used in messages within the protocol.

Defined in namespace: http://www.microsoft.com/SoftwareDistribution

```
<s:complexType name="ArrayOfInt">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="int"
       type="s:int" />
  </s:sequence>
</s:complexType>
```

2.2.1.2 ArrayOfString

An array of string values used in messages within the protocol.

Defined in namespace: http://www.microsoft.com/SoftwareDistribution

```
<s:complexType name="ArrayOfString">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="string"
       nillable="true" type="s:string" />
  </s:sequence>
```

```
</s:complexType>
```

2.2.1.3 AuthorizationCookie

An object returned by the server on successful completion of the GetAuthorizationCookie (section 2.2.2.1) operation.

Defined in namespace: http://www.microsoft.com/SoftwareDistribution

```
<s:complexType name="AuthorizationCookie">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="PlugInId"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="CookieData"
       type="s:base64Binary" />
  </s:sequence>
</s:complexType>
```

PlugInId: Name identifying the Authorization PlugIn issuing the AuthorizationCookie.

CookieData: An opaque sequence of one or more bytes containing implementation-specific authorization and authentication information for use by the server. This element MUST be present.<9>

2.2.1.4 Cookie

Used by the server to store client authorization, authentication, and protocol state information in a format opaque to the client.

Defined in namespace: http://www.microsoft.com/SoftwareDistribution

```
<s:complexType name="Cookie">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="Expiration"
       type="s:dateTime" />
    <s:element minOccurs="0" maxOccurs="1" name="EncryptedData"
       type="s:base64Binary" />
  </s:sequence>
</s:complexType>
```

Expiration: A clear-text copy of the time this cookie expires. The actual time the cookie expires MAY be stored in EncryptedData to prevent client tampering.

EncryptedData: An opaque sequence of one or more bytes that contain implementation-specific authorization, authentication, and protocol state information for use by the server. This element MUST be present.<10>

2.2.1.5 UpdateIdentity

A **globally unique identifier (GUID)** for a specific **revision** of an update.

Defined in namespace: http://www.microsoft.com/SoftwareDistribution

```
<s:complexType name="UpdateIdentity">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="UpdateID"
       type="s2:guid" />
    <s:element minOccurs="1" maxOccurs="1" name="RevisionNumber"
       type="s:int" />
  </s:sequence>
</s:complexType>
```

UpdateID: A GUID that identifies an update.

RevisionNumber: A 32-bit number that uniquely identifies a specific version of an update. These types are specified in section 3.1.1.

2.2.1.6 ArrayOfBase64Binary

An array of binary values encoded in base 64.

Defined in namespace: http://www.microsoft.com/SoftwareDistribution

```
<s:complexType name="ArrayOfBase64Binary">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
               name="base64Binary" nillable="true"
                                   type="s:base64Binary" />
  </s:sequence>
</s:complexType>
```

2.2.2 SimpleAuth Web Service

The SimpleAuth Web service is used for restricting availability of updates to groups of clients.

2.2.2.1 GetAuthorizationCookie

Synopsis:

This method provides a mechanism for the server to authenticate and authorize client access to the client Web service.

```
<wsdl:operation name="GetAuthorizationCookie">
```

The SOAP operation is defined as given below.

```
<soap:operation soapAction="http://www.microsoft.com/
   SoftwareDistribution/Server/SimpleAuthWebService/GetAuthorizationCookie"
   style="document" />
```

Request:

```
<s:element name="GetAuthorizationCookie">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="clientId"
         type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="targetGroupName"
         type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="dnsName"
         type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

clientID: A globally unique string that the client SHOULD generate to identify itself. This element MUST be present.<11>

targetGroupName: A named collection of computers (**target group**) in which the client claims membership.<12>

dnsName: The client's DNS name. A fully-qualified domain name if the client is domain-joined, else the client computer name. This element MUST be present.

Response:

The server MUST return a result with the following syntax.

```
<s:element minOccurs="0" maxOccurs="1"
   name="GetAuthorizationCookieResult" type="s1:AuthorizationCookie" />
```

GetAuthorizationCookieResult: Upon successful completion of this operation, this element MUST be returned. The syntax for the AuthorizationCookie type MUST be as specified in section 2.2.1.3.

2.2.2.2 Ping

Synopsis:

This method does not participate in the Windows Update Services: Client-Server Protocol. Clients SHOULD NOT invoke it, and the server MAY ignore calls to this method.<13>

```
<wsdl:operation name="Ping">
```

2.2.3 Client Web Service

The Client Web service is used for synchronizing metadata to the client.

2.2.3.1 ms-GetConfig

Synopsis:

This method returns information on the server's registration, authorization, and reporting requirements.

```
<wsdl:operation name="GetConfig">
```

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/
   SoftwareDistribution/Server/ClientWebService/GetConfig"
   style="document" />
```

Request:

```
<s:element name="GetConfig">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="protocolVersion"
         type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

protocolVersion: The protocol version used by the client as a two-part version string where the two parts are separated by a period. The client MUST pass "1.6"" in version 3.0 of the protocol. Older versions of the protocol MUST pass "1.0".<14>

Response: The server MUST return a result with the following syntax.

```
<s:element name="GetConfigResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetConfigResult"
         type="s1:Config" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetConfigResult: On successful completion of this operation, this element MUST be returned. The syntax of the GetConfigResult type MUST be as follows:

```
<s:complexType name="Config">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="LastChange"
       type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="IsRegistrationRequired"
       type="s:boolean" />
    <s:element minOccurs="0" maxOccurs="1" name="AuthInfo"
       type="s1:ArrayOfAuthPlugInInfo" />
    <s:element minOccurs="0" maxOccurs="1" name="AllowedEventIds"
       type="s1:ArrayOfInt" />
    <s:element minOccurs="0" maxOccurs="1" name="Properties"
       type="s1:ArrayOfConfigurationProperty" />
  </s:sequence>
</s:complexType>
```

LastChange: The last time configuration data changed on the server.

IsRegistrationRequired: Specifies whether the server requires registration (as specified in section 2.2.3.3). Set TRUE to indicate registration is required. Set FALSE to indicate that registration should not be performed.

AuthInfo: Contains an array of authorization plug-ins supported by the server. On successful execution of this operation, this array MUST contain exactly one element. Its format (ArrayOfAuthPlugInInfo) is as follows:

```
<s:complexType name="ArrayOfAuthPlugInInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="AuthPlugInInfo"
       nillable="true" type="s1:AuthPlugInInfo" />
  </s:sequence>
</s:complexType>
```

AuthPlugInInfo: This field MUST be present and MUST contain exactly one element in the array. It provides information about the Authorization PlugIn available on the server. Its format is as follows:

```
<s:complexType name="AuthPlugInInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="PlugInID"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="ServiceUrl"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Parameter"
       type="s:string" />
  </s:sequence>
</s:complexType>
```

PlugInID: This MUST be present and MUST have the value "SimpleTargeting". The client MUST use this PlugInID in subsequent GetCookie (section 2.2.3.2) calls.

ServiceUrl: This MUST be set to the URL of the SimpleAuth Web service. It is a partial URL that can be appended to http://<server>:<server port>/ to form the full URL to be used for SimpleAuth Web service.

Parameter: Unused. It MUST NOT be present and MUST be ignored upon receipt.

AllowedEventIds: Contains an array of event identifiers specifying events required by the reporting Web service of the server.

Properties: A set of properties used in the protocol. Its format (ArrayOfConfigurationProperties) is in the following example. A ConfigurationProperty with Name set to "MaxExtendedUpdatesPerRequest" MUST be present. The other listed ConfigurationProperties SHOULD be present in version 3.0 of the protocol, but MUST NOT be present in prior versions.

```
<s:complexType name="ArrayOfConfigurationProperty">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
       name="ConfigurationProperty" nillable="true"
       type="s1:ConfigurationProperty" />
  </s:sequence>
```

```
</s:complexType>
```

ConfigurationProperty: Its format is as follows:

```
<s:complexType name="ConfigurationProperty">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="Name"
       type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="Value"
       type="s:string" />
  </s:sequence>
</s:complexType>
```

Name	Value
MaxExtendedUpdatesPerRequest	This element specifies the maximum number of revisionIDs that the server allows the client to specify in the GetExtendedUpdateInfo (section 2.2.3.7) method.
PackageServerShare	This element is a **UncPath** that specifies the repair path.
ProtocolVersion	This element specifies the protocol version number that the Windows Update Services: Client-Server Protocol server is using. It MUST be "3.0".
IsInventoryRequired	The value MUST be "0".
ClientReportingLevel	The value SHOULD be "2".<15>

2.2.3.2 GetCookie

Synopsis:

A client invokes this method to obtain or renew a cookie containing opaque implementation-specific authorization, authentication, and state information for use by the server.

```
<wsdl:operation name="GetCookie">
```

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/
   SoftwareDistribution/Server/ClientWebService/GetCookie"
   style="document" />
```

Request:

```
<s:element name="GetCookie">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="authCookies"
```

```
          type="s1:ArrayOfAuthorizationCookie" />
        <s:element minOccurs="0" maxOccurs="1" name="oldCookie"
          type="s1:Cookie" />
        <s:element minOccurs="1" maxOccurs="1" name="lastChange"
          type="s:dateTime" />
        <s:element minOccurs="1" maxOccurs="1" name="currentTime"
          type="s:dateTime" />
        <s:element minOccurs="0" maxOccurs="1" name="protocolVersion"
          type="s:string" />
      </s:sequence>
    </s:complexType>
  </s:element>
```

authCookies: Specifies an array of authorization cookies. Its format (ArrayOfAuthorizationCookie) is in the following example. Upon successful completion of this operation, this element MUST be present and MUST contain exactly one AuthorizationCookie (section 2.2.1.3).

```
<s:complexType name="ArrayOfAuthorizationCookie">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
      name="AuthorizationCookie" nillable="true"
      type="s1:AuthorizationCookie" />
  </s:sequence>
</s:complexType>
```

AuthorizationCookie: Authorization cookie MUST be as specified in section 2.2.1.3.

oldCookies: Optionally specifies an existing cookie (that MUST have been obtained from a previous method call to GetCookie, GetFileLocations (section 2.2.3.8), or SyncUpdates (section 2.2.3.4)) that needs renewal by the server.

lastChange: Specifies the value returned from the client's most recent call to the GetConfig (section 2.2.3.1) method.

currentTime: The current time on the client.

protocolVersion: The protocol version used by the client as a two-part version string where the two parts are separated by a period. The client MUST pass "1.6"" in version 3.0 of the protocol. Older versions of the protocol MUST pass "1.0".<16>

Response: The server MUST return a result with the following syntax.

```
<s:element name="GetCookieResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetCookieResult"
        type="s1:Cookie" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetCookieResult: On successful completion of this operation, this element MUST be returned. The format for this element MUST be as specified in section 2.2.1.4.

2.2.3.3 RegisterComputer

Synopsis:

The client invokes this method to perform registration with the server by providing information about its operating system, hardware, and network parameter configuration.

```
<wsdl:operation name="RegisterComputer">
```

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/
   SoftwareDistribution/Server/ClientWebService/RegisterComputer"
   style="document" />
```

Request:

```
<s:element name="RegisterComputer">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie"
         type="s1:Cookie" />
      <s:element minOccurs="0" maxOccurs="1" name="computerInfo"
         type="s1:ComputerInfo" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: Specifies a cookie that MUST have been obtained from a previous call to GetCookie (section 2.2.3.2), GetFileLocations (section 2.2.3.8), or SyncUpdates (section 2.2.3.4). This element MUST be present.

computerInfo: Information about the client computer. Its format is as follows:

```
<s:complexType name="ComputerInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="DnsName"
       type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="OSMajorVersion"
       type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="OSMinorVersion"
       type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="OSBuildNumber"
       type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="OSServicePackMajorNumber"
       type="s:short" />
    <s:element minOccurs="1" maxOccurs="1" name="OSServicePackMinorNumber"
       type="s:short" />
    <s:element minOccurs="0" maxOccurs="1" name="OSLocale"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="ComputerManufacturer"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="ComputerModel"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="BiosVersion"
```

```
      type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="BiosName"
      type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="BiosReleaseDate"
      type="s:dateTime" />
    <s:element minOccurs="0" maxOccurs="1" name="ProcessorArchitecture"
      type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="SuiteMask"
      type="s:short" />
    <s:element minOccurs="0" maxOccurs="1" name="OldProductType"
      type="s:unsignedByte" />
    <s:element minOccurs="0" maxOccurs="1" name="NewProductType"
      type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="SystemMetrics"
      type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="ClientVersionMajorNumber"
      type="s:short" />
    <s:element minOccurs="0" maxOccurs="1" name="ClientVersionMinorNumber"
      type="s:short" />
    <s:element minOccurs="0" maxOccurs="1" name="ClientVersionBuildNumber"
      type="s:short" />
    <s:element minOccurs="0" maxOccurs="1" name="ClientVersionQfeNumber"
      type="s:short" />  </s:sequence>
</s:complexType>
```

DnsName: For domain-joined clients, specifies the fully-qualified DNS name of the client machine. For non-domain-joined clients, specifies the client computer name (NETBIOS). This element MUST be present.

OSMajorVersion: The client operating system major version number.

OSMinorVersion: The client operating system minor version number.

OSBuildNumber: The client operating system build number.

OSServicePackMajorNumber: The client operating system service pack major number.

OSServicePackMinorNumber: The client operating system service pack minor number.

OSLocale: The client operating system **locale**, as specified in [MS-LCID].

ComputerManufacturer: The manufacturer of the client computer.

ComputerModel: The model of the client computer.

BiosVersion: The version of the client computer's BIOS firmware.

BiosName: The name of the client computer's BIOS firmware.

BiosReleaseDate: The release date of the client computer's BIOS firmware.

ProcessorArchitecture: The architecture of the client computer's CPU.

SuiteMask, OldProductType, NewProductType, SystemMetrics: Operating system properties that help the server identify the name of the client's operating system. These elements SHOULD be present in version 3.0 of the protocol, but MUST NOT be present in prior versions.<17>

This table details how the **WSUS** server determines the name of the client's operating system. Where the values in the cells below are omitted, the WSUS server does not use that information in determining the operating system name.

Name of client's operating system	OS major version	OS minor version	Suite mask	Old product type	New product type	System metrics	Processor architecture
Windows Vista Business Edition	6				0x00000006		
Windows Vista Business N Edition	6				0x00000010		
Windows Cluster Server Edition	6				0x00000012		
Windows Server Datacenter Edition Full Installation	6				0x00000008		
Windows Server Datacenter Edition Core Installation	6				0x0000000C		
Windows Vista Enterprise Edition	6				0x00000004		
Windows Server Enterprise Edition Full Installation	6				0x0000000A		
Windows Server Enterprise Edition Core Installation	6				0x0000000E		
Windows Server Enterprise Edition For Itanium Based Systems	6				0x0000000F		
Windows Vista Home Basic Edition	6				0x00000002		
Windows Vista Home Basic	6				0x00000005		

Name of client's operating system	OS major version	OS minor version	Suite mask	Old product type	New product type	System metrics	Processor architecture
N Edition							
Windows Vista Home Premium Edition	6				0x00000003		
Windows Home Server Edition	6				0x00000013		
Windows Server For Small Business Edition	6				0x00000018		
Windows Small Business Server	6				0x00000009		
Windows Small Business Server Premium Edition	6				0x00000019		
Windows Server Standard Edition Full Installation	6				0x00000007		
Windows Server Standard Edition Core Installation	6				0x0000000D		
Windows Vista Starter Edition	6				0x0000000B		
Windows Storage Server Enterprise Edition	6				0x00000017		
Windows Storage Server Express	6				0x00000014		

Name of client's operating system	OS major version	OS minor version	Suite mask	Old product type	New product type	System metrics	Processor architecture
Edition							
Windows Storage Server Standard Edition	6				0x00000015		
Windows Storage Server Workgroup Edition	6				0x00000016		
Windows Vista Ultimate Edition	6				0x00000001		
Windows Web Server Edition	6				0x00000011		
Windows 6	6						
Windows Server 2003 R2 Web Edition	5		(suiteMask & 0x00000400) != 0			89	
Windows Server 2003, Web Edition	5		(suiteMask & 0x00000400) != 0				
Windows Server 2003 R2 Compute Cluster Edition	5		(suiteMask & 0x00004000) != 0			89	
Windows Server 2003, Compute Cluster Edition	5		(suiteMask & 0x00004000) != 0				
Windows Server 2003 R2 Datacenter Edition	5	2	(suiteMask & 0x00000080) != 0			89	Contains "X86"
Windows Server 2003 R2 x64 Datacenter	5	2	(suiteMask & 0x00000080) != 0			89	Contains "AMD64"

Name of client's operating system	OS major version	OS minor version	Suite mask	Old product type	New product type	System metrics	Processor architecture
Edition							
Windows Server 2003, Datacenter Edition	5	2	(suiteMask & 0x00000080) != 0				Contains "X86"
Windows Server 2003, Datacenter x64 Edition	5	2	(suiteMask & 0x00000080) != 0				Contains "AMD64"
Windows Server 2003 Datacenter Edition Itanium	5	2	(suiteMask & 0x00000080) != 0				Contains "IA64"
Windows 200 0 Datacenter Server	5		(suiteMask & 0x00000080) != 0				
Windows Server 2003 R2 Enterprise Edition	5	2	(suiteMask & 0x00000002) != 0			89	Contains "X86"
Windows Server 2003 R2 x64 Enterprise Edition	5	2	(suiteMask & 0x00000002) != 0			89	Contains "AMD64"
Windows Server 2003, Enterprise Edition	5	2	(suiteMask & 0x00000002) != 0				Contains "X86"
Windows Server 2003, Enterprise x64 Edition	5	2	(suiteMask & 0x00000002) != 0				Contains "AMD64"
Windows Server 2003 Enterprise Edition Itanium	5	2	(suiteMask & 0x00000002) != 0				Contains "IA64"
Windows 200 0 Advanced Server	5		(suiteMask & 0x00000002) != 0				

Name of client's operating system	OS major version	OS minor version	Suite mask	Old product type	New product type	System metrics	Processor architecture
Windows XP Embedded	5		(suiteMask & 0x00000040) != 0				
Windows XP Home Edition	5		(suiteMask & 0x00000200) != 0				
Windows Storage Server 2003 R2	5		(suiteMask & 0x00002000) != 0			89	
Windows Storage Server 2003	5		(suiteMask & 0x00002000) != 0				
Windows 2000 Professional	5	0		1			
Windows 2000 Server	5	0		> 1			
Windows XP Media Center Edition	5	1				87	
Windows XP Starter Edition	5	1				88	
Windows XP Tablet PC Edition	5	1				86	
Windows XP Professional	5	1					Contains "X86"
Windows XP 64-bit Edition for Itanium systems	5	1					Contains "IA64"
Windows XP Professional x64 Edition	5	2		1			Contains "AMD64"
Windows XP 64 bit Edition Version 2003	5	2		1			Contains "IA64"
Windows Server 2003 R2 Standard	5	2				89	Contains "X86"

Name of client's operating system	OS major version	OS minor version	Suite mask	Old product type	New product type	System metrics	Processor architecture
Edition							
Windows Server 2003 R2 x64 Standard Edition	5	2				89	Contains "AMD64"
Windows Server 2003, Standard Edition	5	2		> 1			Contains "X86"
Windows Server 2003, Standard x64 Edition	5	2		> 1			Contains "AMD64"

ClientVersionMajorNumber: The **WUA** major version number. This is the first part in the four-part WUA version string. This element SHOULD be present in version 3.0 of the protocol, but MUST NOT be present in prior versions.

ClientVersionMinorNumber: The WUA minor version number. This is the second part in the four-part WUA version string. This element SHOULD be present in version 3.0 of the protocol, but MUST NOT be present in prior versions.

ClientVersionBuildNumber: The WUA build number. This is the third part in the four-part WUA version string. This element SHOULD be present in version 3.0 of the protocol, but MUST NOT be present in prior versions.

ClientVersionQfeNumber: The WUA qfe number. This is the fourth part in the four-part WUA version string. This element SHOULD be present in version 3.0 of the protocol, but MUST NOT be present in prior versions.<18>

Response:

```
<s:element name="RegisterComputerResponse">
  <s:complexType />
</s:element>
```

This type has no fields.

2.2.3.4 SyncUpdates

Synopsis:

This method is invoked to perform synchronization of metadata describing software update content. The syntax of this method refers to the following concepts as specified in sections 3.1.1 and 3.2.1.

- The integer-valued **Revision ID** used to identify an update revision.

- The string-valued HardwareID, which identifies a hardware device installed on the client machine.
- The integer-valued **deployment** ID, which identifies a deployment.
- The **prerequisite** relationship between updates.
- The client metadata cache.

```
<wsdl:operation name="SyncUpdates">
```

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/
   SoftwareDistribution/Server/ClientWebService/SyncUpdates"
   style="document" />
```

Request:

```
<s:element name="SyncUpdates">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie"
         type="s1:Cookie" />
      <s:element minOccurs="0" maxOccurs="1" name="parameters"
         type="s1:SyncUpdateParameters" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: Specifies a cookie that MUST have been obtained from a previous call to GetCookie (section 2.2.3.2), GetFileLocations (section 2.2.3.8), or SyncUpdates. This element MUST be present.

parameters: Additional parameters to this method. This element MUST be present. Its format is as follows:

```
<s:complexType name="SyncUpdateParameters">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ExpressQuery"
       type="s:boolean" />
    <s:element minOccurs="0" maxOccurs="1" name="InstalledNonLeafUpdateIDs"
       type="s1:ArrayOfInt" />
    <s:element minOccurs="0" maxOccurs="1" name="OtherCachedUpdateIDs"
       type="s1:ArrayOfInt" />
    <s:element minOccurs="0" maxOccurs="1" name="SystemSpec"
       type="s1:ArrayOfDevice" />
    <s:element minOccurs="0" maxOccurs="1" name="CachedDriverIDs"
       type="s1:ArrayOfInt" />
    <s:element minOccurs="1" maxOccurs="1" name="SkipSoftwareSync"
       type="s:boolean" />
  </s:sequence>
</s:complexType>
```

ExpressQuery: This parameter MUST be absent or set to FALSE by the client. It MUST be ignored by the update server.

InstalledNonLeafUpdateIDs: Contains an array of revision IDs of all non-leaf (in the **prerequisite graph**) revisions in the client cache that are installed on the client. These IDs MUST have been obtained from the UpdateInfo.ID returned from previous calls to this method.

OtherCachedUpdateIDs: Contains an array of revision IDs of other revisions in the client cache. These IDs MUST have been obtained from the UpdateInfo.ID returned from previous calls to this method.

SystemSpec: Specifies the client's existing hardware devices and installed drivers. This information is used in the driver synchronization query to determine if a more closely-matching driver is available on the server. Its format (ArrayOfDevice) is as follows:

```
<s:complexType name="ArrayOfDevice">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="Device"
       nillable="true" type="s1:Device" />
  </s:sequence>
</s:complexType>
```

Device: The SystemSpec is an array of devices. Its format is as follows:

```
<s:complexType name="Device">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="HardwareIDs"
       type="s1:ArrayOfString" />
    <s:element minOccurs="0" maxOccurs="1" name="CompatibleIDs"
       type="s1:ArrayOfString" />
    <s:element minOccurs="0" maxOccurs="1" name="installedDriver"
       type="s1:InstalledDriver" />
  </s:sequence>
</s:complexType>
```

HardwareIDs: An array of HardwareID values that identify the devices hardware supported by this driver.

CompatibleIDs: An array of HardwareID values that identify the compatible hardware for this device driver.

installedDriver: If a driver is already installed for this device, this describes properties of that driver. Its format is as follows:

```
<s:complexType name="InstalledDriver">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="MatchingID"
       type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="DriverVerDate"
       type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="DriverVerVersion"
       type="s:long" />
    <s:element minOccurs="0" maxOccurs="1" name="Class" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Manufacturer"
       type="s:string" />
```

```
    <s:element minOccurs="0" maxOccurs="1" name="Provider"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Model"
       type="s:string" />
  </s:sequence>
</s:complexType>
```

MatchingID: The HardwareID or compatibleID corresponding to the installed driver.

DriverVerDate: The release date of the driver.

DriverVerVersion: The software version of the driver.

Class: The driver class (for example, Printer, Display, and so on), as specified by the driver during installation (typically in the driver INF file).

Manufacturer: The company that created the driver.

Provider: The company providing the driver.

Model: The hardware model that the driver targets.

CachedDriverIDs: The revision IDs of all driver revisions in the client cache. These IDs MUST have been obtained from the UpdateInfo.ID returned from previous calls to this method.

SkipSoftwareSync: Specifies if this request is for a software or driver metadata sync.

Response:

```
<s:element name="SyncUpdatesResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SyncUpdatesResult"
         type="s1:SyncInfo" />
    </s:sequence>
  </s:complexType>
</s:element>
```

SyncUpdatesResult: Upon successful completion of this operation, this element MUST be returned. The client SHOULD interpret this result, as specified in section 3.1.5.6. Its format is as follows:

```
<s:complexType name="SyncInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="NewUpdates"
       type="s1:ArrayOfUpdateInfo" />
    <s:element minOccurs="0" maxOccurs="1" name="OutOfScopeRevisionIDs"
       type="s1:ArrayOfInt" />
    <s:element minOccurs="0" maxOccurs="1" name="ChangedUpdates"
       type="s1:ArrayOfUpdateInfo" />
    <s:element minOccurs="1" maxOccurs="1" name="Truncated"
       type="s:boolean" />
    <s:element minOccurs="0" maxOccurs="1" name="NewCookie"
       type="s1:Cookie" />
  </s:sequence>
</s:complexType>
```

NewUpdates: An array of revisions to be added to the client cache. Its format is as follows:

```
<s:complexType name="ArrayOfUpdateInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="UpdateInfo"
       nillable="true" type="s1:UpdateInfo" />
  </s:sequence>
</s:complexType>
```

UpdateInfo: Information about an update revision. Its format is as follows:

```
<s:complexType name="UpdateInfo">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="Deployment"
       type="s1:Deployment" />
    <s:element minOccurs="1" maxOccurs="1" name="IsLeaf"
       type="s:boolean" />
    <s:element minOccurs="0" maxOccurs="1" name="Xml"
       type="s:string" />
  </s:sequence>
</s:complexType>
```

ID: Specifies the revision ID of this update revision. This ID will be passed as Parameters.InstalledNonLeafUpdateIDs, Parameters.OtherCachedUpdateIDs, or Parameters.CachedDriverIDs in subsequent calls to this method.

Deployment: Information about how this revision was deployed to this client. Its format MUST be as follows:

```
<s:complexType name="Deployment">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="Action"
       type="s1:DeploymentAction" />
    <s:element minOccurs="0" maxOccurs="1" name="Deadline"
       type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="IsAssigned"
       type="s:boolean" />
    <s:element minOccurs="0" maxOccurs="1" name="LastChangeTime"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="DownloadPriority"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="HardwareIds"
       type="s1:ArrayOfString" />
  </s:sequence>
</s:complexType>
```

ID: The server-assigned ID for this deployment.

Action: The action the client SHOULD perform on this revision: Install, Uninstall, PreDeploymentCheck (which means do not offer the update, just report back on the status), Block

(which means that the update must not be deployed, and is used to override another deployment), Evaluate (which means do not offer the update and do not report back on the status), or Bundle (which means that the update should not be offered for install—it is only deployed because it is bundled by some other explicitly deployed update). The enumeration (DeploymentAction) for this element is as follows:

```
<s:simpleType name="DeploymentAction">
  <s:restriction base="s:string">
    <s:enumeration value="Install" />
    <s:enumeration value="Uninstall" />
    <s:enumeration value="PreDeploymentCheck" />
    <s:enumeration value="Block" />
    <s:enumeration value="Evaluate" />
    <s:enumeration value="Bundle" />
  </s:restriction>
</s:simpleType>
```

Deadline: Optionally specifies the time by which the deployment action SHOULD occur, in the syntax specified for s:dateTime (as specified in [XMLSCHEMA2] section 3.2.7). This field MAY be omitted if there is no deadline.

IsAssigned: If set to TRUE, the revision SHOULD be installed automatically by the client. If set to FALSE, the revision SHOULD be offered to users of the client computer but not automatically installed.

LastChangeTime: Specifies when the deployment was created, in the syntax specified for s:date (as specified in [XMLSCHEMA2] section 3.2.9).

DownloadPriority: Specifies a value indicating how the client SHOULD prioritize its downloads of content for this update relative to other updates.

HardwareIds: An optional array of HardwareID values that identify the device hardware supported by this update revision. This SHOULD be present when the update revision this Deployment is associated with is a driver update.<19>

IsLeaf: Specifies whether this revision is a leaf in the prerequisite graph.

Xml: The core metadata associated with this revision. The server MUST populate this with metadata. These fragments are created as specified in section 3.1.1. The format of the fragment is opaque to the server.

OutOfScopeRevisionIDs: An array of RevisionIDs that identify revisions to be removed from the client cache.

ChangedUpdates: Changes that SHOULD be applied to the deployment or IsLeaf status for revisions in the client cache.

Truncated: Specifies that the server has truncated the set of new revisions returned. If the results have been truncated, it MUST be set to TRUE to indicate that the client SHOULD call this method again. If the results have not been truncated, it MUST be set to FALSE to indicate that the results have not been truncated, so the client SHOULD only call this method again if the method returns a new UpdateInfo with IsLeaf = FALSE.

NewCookie: An updated cookie that the client SHOULD use in subsequent calls.

2.2.3.5 SyncPrinterCatalog

Synopsis:

This method does not participate in the Windows Update Services: Client-Server Protocol. Clients SHOULD NOT invoke it, and the server SHOULD ignore calls to this method.

```
<wsdl:operation name="SyncPrinterCatalog">
```

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/
   SoftwareDistribution/Server/ClientWebService/SyncPrinterCatalog"
   style="document" />
```

2.2.3.6 RefreshCache

Synopsis:

This method is invoked by the client to update its cache of mappings between compact RevisionIDs and globally-unique update identifiers. The difference between these two types of identifiers is specified in section 3.1.1.

```
<wsdl:operation name="RefreshCache">
<soap:operation soapAction="http://www.microsoft.com/
   SoftwareDistribution/Server/ClientWebService/RefreshCache"
   style="document" />
```

Request:

```
<s:element name="RefreshCache">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie"
         type="s1:Cookie" />
      <s:element minOccurs="0" maxOccurs="1" name="globalIDs"
         type="s1:ArrayOfUpdateIdentity" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: Specifies a cookie that MUST have been obtained from a previous call to GetCookie (section 2.2.3.2), GetFileLocations (section 2.2.3.8), or SyncUpdates (section 2.2.3.4). This element MUST be present.

globalIDs: An array of UpdateIdentity elements. Its format MUST be as follows:

```
<s:complexType name="ArrayOfUpdateIdentity">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="UpdateIdentity"
       nillable="true" type="s1:UpdateIdentity" />
```

```
  </s:sequence>
</s:complexType>
```

UpdateIdentity: Identifies an update revision. Its format MUST be as specified in section 2.2.1.5:

Response:

```
<s:element name="RefreshCacheResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="RefreshCacheResult"
         type="s1:ArrayOfRefreshCacheResult" />
    </s:sequence>
  </s:complexType>
</s:element>
```

RefreshCacheResult: Upon successful completion of this operation, this element MUST be returned. Its format MUST be as follows:

```
<s:complexType name="ArrayOfRefreshCacheResult">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="RefreshCacheResult"
       nillable="true" type="s1:RefreshCacheResult" />
  </s:sequence>
</s:complexType>
```

RefreshCacheResult: Specifies information about a specific revision. Its format MUST be as follows:

```
<s:complexType name="RefreshCacheResult">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="RevisionID"
       type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="GlobalID"
       type="s1:UpdateIdentity" />
    <s:element minOccurs="1" maxOccurs="1" name="IsLeaf"
       type="s:boolean" />
    <s:element minOccurs="0" maxOccurs="1" name="Deployment"
       type="s1:Deployment" />
  </s:sequence>
</s:complexType>
```

RevisionID: Specifies the new 32-bit integer ID for the revision.

GlobalID: The UpdateIdentity for this revision. This field MUST be present. The format of this field is specified in section 2.2.1.5.

IsLeaf: Specifies whether the revision is a leaf of a hierarchy of updates.

Deployment: The deployment for this revision. This field MUST be present. Its format MUST be as specified in section 2.2.3.4.

Each of the above values is specified in more detail in section 3.1.1.

2.2.3.7 GetExtendedUpdateInfo

Synopsis:

This method is invoked to obtain detailed metadata for an update.

As specified in section 1.3, the client does not download all the metadata at once during the call to SyncUpdates (section 2.2.3.4). Rather, the metadata is divided into fragments, and only the "core" fragment, which contains just enough metadata to allow the client to evaluate if the content is needed, is returned to the client. If the client determines the content is needed, it SHOULD then call this method to obtain the additional metadata fragments that it requires.

```
<wsdl:operation name="GetExtendedUpdateInfo">
```

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/
   SoftwareDistribution/Server/ClientWebService/GetExtendedUpdateInfo"
   style="document" />
```

Request:

```
<s:element name="GetExtendedUpdateInfo">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie"
         type="s1:Cookie" />
      <s:element minOccurs="0" maxOccurs="1" name="revisionIDs"
         type="s1:ArrayOfInt" />
      <s:element minOccurs="0" maxOccurs="1" name="infoTypes"
         type="s1:ArrayOfXmlUpdateFragmentType" />
      <s:element minOccurs="0" maxOccurs="1" name="locales"
         type="s1:ArrayOfString" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: Specifies a cookie that MUST have been obtained from a previous call to GetCookie (section 2.2.3.2), GetFileLocations (section 2.2.3.8), or SyncUpdates (section 2.2.3.4). This element MUST be present.

revisionIDs: Specifies the array of revision IDs for which extended metadata fragments are to be returned.

infoTypes: Specifies the type of metadata fragments to be returned. Its format (ArrayOfXmlUpdteFragmentsType) MUST be as follows:

```
<s:complexType name="ArrayOfXmlUpdateFragmentType">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
```

```
        name="XmlUpdateFragmentType" type="s1:XmlUpdateFragmentType" />
    </s:sequence>
  </s:complexType>
```

XmlUpdateFragmentType: Species the type of metadata fragment. Its enumeration is as follows:

```
<s:simpleType name="XmlUpdateFragmentType">
  <s:restriction base="s:string">
    <s:enumeration value="Published" />
    <s:enumeration value="Core" />
    <s:enumeration value="Extended" />
    <s:enumeration value="VerificationRule" />
    <s:enumeration value="LocalizedProperties" />
    <s:enumeration value="Eula" />
  </s:restriction>
</s:simpleType>
```

Locales: Optionally specifies the locales for which localizable extended metadata MUST be returned. Localizable metadata are elements such as human-readable strings, which are represented differently between locales, as specified in [MS-LCID].

Response:

```
<s:element name="GetExtendedUpdateInfoResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
         name="GetExtendedUpdateInfoResult" type="s1:ExtendedUpdateInfo" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetExtendedUpdateInfoResult: Upon successful completion of this operation, this element MUST be returned. Its format MUST be follows:

```
<s:complexType name="ExtendedUpdateInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Updates"
       type="s1:ArrayOfUpdateData" />
    <s:element minOccurs="0" maxOccurs="1" name="FileLocations"
       type="s1:ArrayOfFileLocation" />
    <s:element minOccurs="0" maxOccurs="1" name="OutOfScopeRevisionIDs"
       type="s1:ArrayOfInt" />
  </s:sequence>
</s:complexType>
```

Updates: An array of entries containing the extended metadata requested for each update. Its format (ArrayOfUpdateData) MUST be as follows:

```
<s:complexType name="ArrayOfUpdateData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="Update"
```

```
        nillable="true" type="s1:UpdateData" />
    </s:sequence>
  </s:complexType>
```

Update: Specifies the extended metadata for an update. Its format MUST be as follows:

```
<s:complexType name="UpdateData">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="Xml" type="s:string" />
  </s:sequence>
</s:complexType>
```

ID: The revision ID of the revision.

Xml: An extended metadata fragment for this update. This element MUST be present. These fragments are created as specified in section 3.1.1. The format of the fragment is opaque to the server.

FileLocations: An array of locations for the content corresponding to each update. Its format (ArrayOfFileLocation) MUST be as follows:

```
<s:complexType name="ArrayOfFileLocation">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="FileLocation"
       nillable="true" type="s1:FileLocation" />
  </s:sequence>
</s:complexType>
```

FileLocation: Specifies the location of the file. Its format MUST be as follows:

```
<s:complexType name="FileLocation">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="FileDigest"
       type="s:base64Binary" />
    <s:element minOccurs="0" maxOccurs="1" name="Url"
       type="s:string" />
  </s:sequence>
</s:complexType>
```

FileDigest: This field MUST be present. The value MUST be the SHA-1 hash computed over the content of the file.

Url: This field MUST be present. It is an HTTP URI from which it MUST be possible to download the file.

OutOfScopeRevisionIDs: Specifies an array of revision IDs that SHOULD be purged from the client's cache because these updates are no longer in-scope for the client.

2.2.3.8 GetFileLocations

Synopsis:

Returns the URL where the specified set of files can be found.

```
<wsdl:operation name="GetFileLocations">
```

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/
   SoftwareDistribution/Server/ClientWebService/GetFileLocations"
   style="document" />
```

Request:

```
<s:element name="GetFileLocations">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie"
         type="s1:Cookie" />
      <s:element minOccurs="0" maxOccurs="1" name="fileDigests"
         type="s1:ArrayOfBase64Binary" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: Specifies a cookie that MUST have been obtained from a previous call to GetCookie (section 2.2.3.2), GetFileLocations, or SyncUpdates (section 2.2.3.4). This element MUST be present.

fileDigests: An array of digests for content corresponding to updates. Each digest MUST be specified as a SHA1-hash. The format is specified in section 2.2.1.6.

Response:

```
<s:element name="GetFileLocationsResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetFileLocationsResult"
         type="s1:GetFileLocationsResults" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetFileLocationsResults: Upon successful completion of the operation, this element MUST be returned. Its format MUST be as follows:

```
<s:complexType name="GetFileLocationsResults">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="FileLocations"
       type="s1:ArrayOfFileLocation" />
```

```
      <s:element minOccurs="0" maxOccurs="1" name="NewCookie"
         type="s1:Cookie" />
    </s:sequence>
  </s:complexType>
```

FileLocations: The file locations for the files. Its format is specified in section 2.2.3.7.

NewCookie: A replacement cookie, generated by the server, that SHOULD be provided by the client in subsequent method invocations.

2.2.3.9 Ping

Synopsis:

This method does not participate in the Windows Update Services: Client-Server Protocol. Clients SHOULD NOT invoke it, and the server MAY ignore calls to this method.<20>

```
  <wsdl:operation name="Ping">
```

2.2.4 Reporting Web Service

The reporting Web service is used by clients to report selected events that contain information on their update activity.

2.2.4.1 ReportEventBatch

Synopsis:

This method is invoked by a client to report the occurrence of one or more software-update–related events.

```
  <wsdl:operation name="ReportEventBatch">
```

The SOAP operation is defined as follows:

```
  <soap:operation soapAction="http://www.microsoft.com/
     SoftwareDistribution/ReportEventBatch" style="document" />
```

Request:

```
  <s:element name="ReportEventBatch">
    <s:complexType>
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="cookie"
           type="s1:Cookie" />
        <s:element minOccurs="1" maxOccurs="1" name="clientTime"
           type="s:dateTime" />
        <s:element minOccurs="0" maxOccurs="1" name="eventBatch"
           type="s1:ArrayOfReportingEvent" />
      </s:sequence>
```

```
    </s:complexType>
  </s:element>
```

cookie: Specifies a cookie that MUST have been obtained from a previous call to GetCookie (section 2.2.3.2), GetFileLocations (section 2.2.3.8), or SyncUpdates (section 2.2.3.4). This element MUST be present.

clientTime: The current time (when this method was called) on the client in Coordinated Universal Time (UTC).

eventBatch: An array of ReportingEvents. Its format MUST be as follows:

```
<s:complexType name="ArrayOfReportingEvent">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="ReportingEvent"
       nillable="true" type="s1:ReportingEvent" />
  </s:sequence>
</s:complexType>
```

ReportingEvent: Specifies information of an update-related event. Its format MUST be as follows:

```
<s:complexType name="ReportingEvent">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="BasicData"
       type="s1:BasicData" />
    <s:element minOccurs="0" maxOccurs="1" name="ExtendedData"
       type="s1:ExtendedData" />
    <s:element minOccurs="0" maxOccurs="1" name="PrivateData"
       type="s1:PrivateData" />
  </s:sequence>
</s:complexType>
```

BasicData: Generic data provided for all events. Its format (BasicData) MUST be as follows:

```
<s:complexType name="BasicData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="TargetID"
       type="s1:ComputerTargetIdentifier" />
    <s:element minOccurs="1" maxOccurs="1" name="SequenceNumber"
       type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="TimeAtTarget"
       type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="EventInstanceID"
       type="s2:guid" />
    <s:element minOccurs="1" maxOccurs="1" name="NamespaceID"
       type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="EventID"
       type="s:short" />
    <s:element minOccurs="1" maxOccurs="1" name="SourceID"
       type="s:short" />
    <s:element minOccurs="0" maxOccurs="1" name="UpdateID"
       type="s1:UpdateRevisionIdentifier" />
    <s:element minOccurs="1" maxOccurs="1" name="Win32HResult"
```

```
        type="s:int" />
      <s:element minOccurs="0" maxOccurs="1" name="AppName"
        type="s:string" />
    </s:sequence>
  </s:complexType>
```

TargetID: The identity of the client computer (same as the ClientID parameter to GetAuthorizationCookie (section 2.2.2.1)). Its format (ComputerTargetIdentifier) MUST be as follows:

```
<s:complexType name="ComputerTargetIdentifier">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Sid"
      type="s:string" />
  </s:sequence>
</s:complexType>
```

Sid: The unique ID for a computer, as specified in [MS-SECO].

SequenceNumber: Unused. MUST be set to 0, and MUST be ignored upon receipt.

TimeAtTarget: The time in Coordinated Universal Time (UTC) when the event was recorded by the client.

EventInstanceID: A GUID generated by the client to uniquely identify this occurrence of this event.

NamespaceID: MUST be set to 1 by all clients. The server MAY ignore events with other values.<21>

EventID: This SHOULD be set to a numeric value that identifies the type of the event that occurred on the client.<22>

SourceID: Defines the subcomponent within the client that generated the event.

UpdateID: Optionally specifies the ID of the update related to this occurrence of this event, for events related to a particular update. If the event is not associated with any particular update, it MUST be specified as {00000000-0000-0000-0000-000000000000}. Its format (UpdateRevisionIdentifier) MUST be as given below.

```
<s:complexType name="UpdateRevisionIdentifier">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="UpdateID"
      type="s2:guid" />
    <s:element minOccurs="1" maxOccurs="1" name="RevisionNumber"
      type="s:int" />
  </s:sequence>
</s:complexType>
```

UpdateID: A GUID that uniquely identifies an update.

RevisionNumber: A number that specifies the version of the update identified by this revision.

Win32HResult: Optionally specifies a Win32 HRESULT code for events that correspond to a failure. Win32 HRESULT codes are specified in [MS-ERREF].

AppName: The name of the application that triggered the client to perform the operation.

ExtendedData: Additional data associated with an event. Its format (ExtendedData) MUST be as given below.

```
<s:complexType name="ExtendedData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="ReplacementStrings"
       type="s1:ArrayOfString" />
    <s:element minOccurs="0" maxOccurs="1" name="MiscData"
       type="s1:ArrayOfString" />
    <s:element minOccurs="0" maxOccurs="1" name="ComputerBrand"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="ComputerModel"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="BiosRevision"
       type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="ProcessorArchitecture"
       type="s1:ProcessorArchitecture" />
    <s:element minOccurs="1" maxOccurs="1" name="OSVersion"
       type="s1:DetailedVersion" />
    <s:element minOccurs="1" maxOccurs="1" name="OSLocaleID"
       type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="DeviceID"
       type="s:string" />
  </s:sequence>
</s:complexType>
```

ReplacementStrings: Specifies an array of strings to be used as parameters in expanding parameterized message strings. Parameterized message strings are message strings that contain placeholders into which instance-specific strings must be inserted to obtain an expanded message string. Its format MUST be:

```
<s:complexType name="ArrayOfString">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="string"
       nillable="true" type="s:string" />
  </s:sequence>
</s:complexType>
```

MiscData: Additional data not covered by the properties listed here. These MUST be specified in the form "x=value" where x is called a tag.<23> See the MiscData table below for information on such well-known tags.

ComputerBrand: Client computer manufacturer.

ComputerModel: Client computer model name.

BiosRevision: Client BIOS firmware revision.

ProcessorArchitecture: Client CPU architecture. The value MUST be one of those enumerated:

```
<s:simpleType name="ProcessorArchitecture">
```

```
  <s:restriction base="s:string">
    <s:enumeration value="Unknown" />
    <s:enumeration value="X86Compatible" />
    <s:enumeration value="IA64Compatible" />
    <s:enumeration value="Amd64Compatible" />
  </s:restriction>
</s:simpleType>
```

OSVersion: Client operating system version. Its format (DetailedVersion) MUST be as follows:

```
<s:complexType name="DetailedVersion">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="Major" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="Minor" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="Build" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="Revision" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="ServicePackMajor"
       type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="ServicePackMinor"
       type="s:int" />
  </s:sequence>
</s:complexType>
```

OSLocaleID: The client operating system locale.

DeviceID: An event-dependent string.

- MUST be a zero-length string if the associated update is not a driver. How the client can determine if an update is a driver is specified in section 3.1.1.
- If EventID indicates a download driver success/failure event, MUST be set to "1".
- If EventID indicates a driver install success/failure event, MUST be set to a non-empty string whose format is opaque to the server.
- Else, MUST be a zero-length string.

PrivateData: This field MUST be present and empty, and MUST be ignored upon receipt. Its format (PrivateData) is as follows:

```
<s:complexType name="PrivateData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="ComputerDnsName"
       type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="UserAccountName"
       type="s:string" />
  </s:sequence>
</s:complexType>
```

ComputerDnsName: This field MUST be a zero-length string.

UserAccountName: This field MUST be a zero-length string.

MiscData Table

This table defines the tags, their descriptions, and the events for which such tags are used by WUA to populate the MiscData element within a ReportingEvent.

Friendly name	Misc data tag	Data type	Description	Associated event IDs
OSServicePackBuildNumber	A	Int	Client operating system service pack build number.	All events
ByteCount	B	Int	Number of bytes downloaded.	Download success events
RepeatFailFlag	C	Int	Number of times the update has failed to install in the past.	Install failure events
NumberApplicable	D	Int	Number of updates that were explicitly deployable, installable on the client (Needed), and deployed for install.	Detection events
LastError	F	Int	Secondary error code when Win32HResult is not sufficient to diagnose the problem.	Install and download events
ClientVersion	G	Int	Client version.	All events
ClientSamplingValue	J	Int	Random number between 0 and 999 inclusive, chosen once per computer.	All events
BiosName	K	String	Client BIOS name.	All events
BiosReleaseDate	L	DateTime	Client BIOS release date.	All events
EventType	Q	Int	If this is an install or download event, and the update being reported on is a bundle, then 2. Else 1.	All events
TargetClientVersion	S	String	WUA agent version that the agent is self-updating to.	Self-update events
InstallableUpdateIDs	U	List of GUIDs separated by ;	IDs of updates that were deployed for install, uninstall, or scan, that were found to be installable (Needed).	Status events
InstalledUpdateIDs	V	List of GUIDs separated by ;	IDs of updates that were deployed for install, uninstall, or scan, that were found to be installed.	Status events
InstalledPendingRebootIDs	W	List of GUIDs separated by ;	IDs of updates that were deployed for install, uninstall, or scan, that were found to be installed, but require reboot before taking effect.	Status events

Friendly name	Misc data tag	Data type	Description	Associated event IDs
FailedUpdateIDs	g	List of GUIDs separated by ;	IDs of updates that were deployed for install, uninstall, or scan, that were found to need installing or uninstalling, but the attempt to download, install, or uninstall the update has failed.	MS-WUSP 3.0 status events
DownloadedUpdateIDs	h	List of GUIDs separated by ;	IDs of updates that were deployed for install, that were downloaded to the client but not yet installed.	MS-WUSP 3.0 status events
MSIAction	X	String	**MSI** action that failed.	MSI/MSP install failure events
MSITransactionCode	Y	GUIDs	Unique ID identifying the transaction that the update was part of.	MSI/MSP install failure events
MSIProductCode	Z	GUIDs	Product code of the MSP.	MSI/MSP install failure events
MSIErrorRec	f	int	Extended MSI error code.	MSI/MSP install failure events
SelfUpdatePackageName	a	String	The name of the self-update package.	Self-update events
OSProductType	c	Int	Client operating system ProductType.	All events, if client is Windows Vista or later

EventID Table

This table specifies the list of events that WUA reports to a Windows Server Update Services server. The Name column is a friendly name for the event type, EventID is the numeric identifier used in ReportingEvent, and the English message template is the message that could be shown to the end user for this event type. Where a message template is parameterized (%1, %2, and so on), the Windows Server Update Services server populates the placeholders with content from the ReplacementStrings element of the ReportingEvent. The placeholders are specified starting with %1 and are not recursively evaluated. The ReplacementStrings array MUST contain the same number of elements as the number of placeholders specified in the parameterized message.

Name	Event ID	Description	Remarks	English message template
Detection Events				

Name	Event ID	Description	Remarks	English message template
AGENT_DETECTION_FINISHED	147	Detection succeeded.		Agent has finished detecting items.
AGENT_DETECTION_FAILED	148	Detection failed.		Error: Agent failed detecting with reason: %1.
Status Events				
AGENT_STATUS	153	Reports status of all updates to MS-WUSP 2.0.		Reporting client status.
AGENT_STATUS_30	156	Reports status of all updates to MS-WUSP 3.0.	This event is new in version 3.0 of the protocol.	Reporting client status.
Download Events				
AGENT_DOWNLOAD_FAILED	161	Download failed.		Error: Download failed.
AGENT_DOWNLOAD_SUCCEEDED	162	Download succeeded.		Download succeeded.
AGENT_DOWNLOAD_CANCELED	163	Download canceled.		Download canceled.
Install Events				
AGENT_INSTALLING_FAILED	182	User-initiated installation failed.		Installation Failure: Windows failed to install the following update with error %1: %2.
AGENT_INSTALLING_SUCCEEDED	183	User-initiated installation succeeded.		Installation Successful: Windows successfully installed the following update: %1.

Name	Event ID	Description	Remarks	English message template
AGENT_INSTALL_COMPLETE_ WITH_REBOOT	184	User-initiated installation succeeded, requires reboot.		Installation successful and restart required for the following update: %1.
AGENT_INSTALL_CANCEL	186	User-initiated installation canceled.		User canceled the installation.
AGENT_INSTALL_KILLED	187	User-initiated installation timed out, was terminated by the WUA.		Installation killed: Installation of the following update is killed by the agent: %2.
AU_SCHEDULED_INSTALL_SUCCESS	190	Scheduled installation succeeded.		Installation Successful: Windows successfully installed the following update: %1.
AU_SCHEDULED_INSTALL _COMPLETE_WITH_REBOOT	191	Scheduled installation succeeded, requires reboot.		Installation successful and restart required for the following update: %1.
AU_SCHEDULED_INSTALL_KILLED	192	Scheduled installation timed out, was terminated by the WUA.		Installation killed: Installation of the following update is killed by the agent: %2
AU_SCHEDULED_INSTALL_FAILED	195	Scheduled installation failed.		Installation Failure: Windows failed to install the following update with error %1: %2.
AU_SHUTDOWN_INSTALL_SUCCESS	197	Install at shutdown succeeded.		Installation Successful: Windows successfully installed the following update: %1.

Name	Event ID	Description	Remarks	English message template
AU_SHUTDOWN_INSTALL_FAILED	198	Install at shutdown failed.		Installation Failure: Windows failed to install the following update with error %1: %2.
AU_SHUTDOWN_INSTALL_COMPLETE_WITH_REBOOT	199	Install at shutdown succeeded, requires reboot.		Installation successful and restart required for the following update: %1.
AU_SHUTDOWN_INSTALL_KILLED	200	Install at shutdown timed out, was terminated by the WUA.		Installation killed: Installation of the following update is killed by the agent: %2.
AGENT_INSTALLING_FAILED_POST_REBOOT	203	Post-reboot processing of an installed update failed.	New to Windows Vista	Installation Failure Post Reboot.
AGENT_UNINSTALLING_FAILED	221	Uninstallation failed.		Uninstallation Failure: Windows failed to uninstall the following update with error %1: %2.
AGENT_UNINSTALLING_SUCCEEDED	222	Uninstallation succeeded.		Uninstallation Successful: Windows successfully uninstalled the following update: %1.
AGENT_UNINSTALL_CANCEL	223	Uninstallation canceled.		User canceled the uninstall.
AGENT_UNINSTALL_COMPLETE_WITH_REBOOT	224	Uninstallation succeeded, requires reboot.		Uninstallation successful and restart required for the following update: %1.
AGENT_UNINSTALL_KILLED	225	Uninstallation timed out, was terminated by		Uninstallation killed: Uninstallation of

Name	Event ID	Description	Remarks	English message template
		the WUA.		the following update is killed by the agent: %2.
Update Hidden/Unhidden Events				
AGENT_INSTALL_HIDE	185	User chose to hide the update from the UI.		Hide update: user hid one update.
AGENT_INSTALL_UNHIDE	196	User chose to unhide the update from the UI.		Unhide update: user unhid one update.
Misc Events				
AGENT_INSTALLING_PENDING	201	Self-update is about to begin.		Installation pending.
AU_REBOOT_COMPLETED	202	Client computer has finished rebooting following an install/uninstall.		Reboot completed.

Response:

```
<s:element name="ReportEventBatchResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="ReportEventBatchResult"
         type="s:boolean" />
    </s:sequence>
  </s:complexType>
</s:element>
```

ReportEventBatchResult: Upon successful completion of this operation, this MUST be returned. Its value MUST be TRUE if the events were successfully received by the server. Otherwise, it MUST be FALSE.

2.2.4.2 Ping

Synopsis:

This method does not participate in theWindows Update Services: Client-Server Protocol. Clients SHOULD NOT invoke it, and the server MAY ignore calls to this method.<24>

```
<wsdl:operation name="Ping">
```

2.2.5 Faults

The Windows Update Services: Client-Server Protocol allows a server to notify a client of application-level faults by generating SOAP faults by using the following XML syntax.

It does this by throwing a SOAP fault (as specified in [SOAP1.1] section 4.4) with XML, which MUST have the following format in the faults "detail" element.

<ErrorCode>errorCode</ErrorCode>

<Message>message</Message>

<ID>id</ID>

errorCode: Identifies the type of exception being thrown, the value of which MUST be one of those in the following table. The Description column in the table below defines possible actions that the client SHOULD take in response to the error.

message: An ErrorCode-specific user friendly string. This MAY be omitted.

id: A server-generated GUID that uniquely identifies the particular instance of this fault.<25>

ErrorCode	Description
InvalidCookie	The server cannot decrypt the cookie to validate it. The client MUST discard the cookie and start the initial handshake again by calling GetConfig (section 2.2.3.1), GetAuthorizationCookie (section 2.2.2.1), GetCookie (section 2.2.3.2), and RefreshCache (section 2.2.3.6).
ConfigChanged	The server configuration has changed since the last time the client called GetConfig (section 2.2.3.1). The client MUST call GetConfig, GetAuthorizationCookie (section 2.2.2.1), and GetCookie (section 2.2.3.2) again.
RegistrationRequired	Client registration is required. The client SHOULD call RegisterComputer (section 2.2.3.3) before calling SyncUpdates (section 2.2.3.4) again.
ServerChanged	The server has changed. The client MUST go through the same rest logic as it would with an InvalidCookie exception.
InternalServerError	An internal error has occurred in the server while processing the request.
CookieExpired	The client is using an expired cookie. It MUST call GetAuthorizationCookie (section 2.2.2.1) and GetCookie (section 2.2.3.2) again to renew its expired cookie.
InvalidParameters	The client has passed invalid parameters to the server. The "message" part of the exception will contain the parameter name.
InvalidAuthorizationCookie	The authorization cookie passed to GetCookie (section 2.2.3.2) is invalid.
RegistrationNotRequired	The client has called RegisterComputer (section 2.2.3.3) even though the server told it not to in the GetConfig (section 2.2.3.1) response.
ServerBusy	The server is too busy to handle this request. The client SHOULD try again

ErrorCode	Description
	later.
FileLocationChanged	The file locations have changed since the last time the client synced. The client SHOULD call GetFileLocations (section 2.2.3.8) to get the current file locations.

2.2.6 Content Directory and Client Self-Update Tree

Both the Content Directory and the Client Self-Update Tree MUST support HTTP requests, as specified in [RFC2616]. The **content directory** MUST support HTTP HEAD and GET (range) request messages.

3 Protocol Details

The Windows Update Services: Client-Server Protocol operates between a client (the initiator) and a server (the responder).

The protocol incorporates mechanisms to enable stateless server operation wherever possible. In particular, a server generates an encrypted cookie that encapsulates the server's protocol state with respect to each client, and the server requires the client to retain that cookie on its behalf. The client presents its cookie when invoking methods against the server, and the server updates the cookie as appropriate over the course of its communication with the client.

3.1 Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model and possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as the external behavior is consistent with that described in this document.

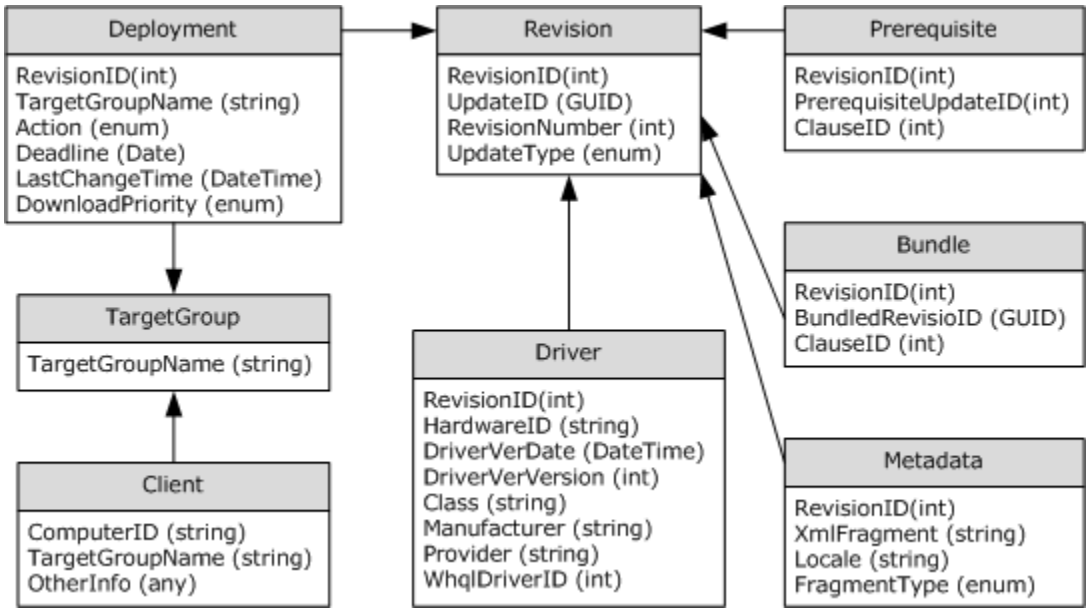

Figure 2: Server abstract data model

In the preceding diagram, each table specifies a type of entity in the model, and each arrow specifies that one type of entity always contains a reference to another.

Revision Table: A collection of entries corresponding to the revisions available on the server. Each entry MUST be uniquely identified by an UpdateID and RevisionNumber, and MUST also be independently identified uniquely by a RevisionID. Each entry SHOULD include the following elements:

- **RevisionID:** A server-assigned revision identifier. Because it is more compact than the globally unique (UpdateID, RevisionNumber) pair, the RevisionID is used as the revision identifier during client/server communications to minimize network traffic.
- **UpdateID:** A GUID that MUST be the same for all revisions of an update.
- **RevisionNumber:** Used in conjunction with the UpdateID to uniquely identify the revision.
- **UpdateType:** Specifies whether the update is a Driver, Software, Category, or Detectoid. The SyncUpdates (section 2.2.3.4) method treats Drivers and non-drivers differently, as specified in section 3.1.5.6.

Driver Table: A collection of entries corresponding to the revisions available on the server with an UpdateType of "Driver". Each entry MUST be uniquely identified by a RevisionID. Each entry MUST include the following elements:

- **RevisionID:** A reference to the revision of the driver.
- **HardwareID:** A value that identifies the device hardware supported by the driver.
- **DriverVerDate:** The release date of the driver.
- **DriverVerVersion:** The software version of the driver.
- **Class:** The device function class of the driver.
- **Manufacturer:** The company that created the driver.
- **Provider:** The company providing the driver.
- **Model:** The model of hardware that the driver targets.
- **WhqlDriverID:** The ID assigned to this driver as part of the driver certification process. A higher number means the driver was submitted to the Microsoft Update live service more recently.

Metadata Table: A collection of metadata fragments, which are derived from the revision metadata, as specified in section 3.1.1.1. Each entry MUST be uniquely identified by the combination of its RevisionID, FragmentType, and Locale. Each entry MUST include the following elements:

- **RevisionID:** A reference to the revision that this fragment was derived from.
- **XmlFragment:** The metadata fragment.
- **FragmentType:** The type of fragment: Core, Extended, LocalizedProperties, or Eula.
- **Locale:** For entries with FragmentType of LocalizedProperties or Eula, specifies the locale of the entry. NULL for entries of other FragmentTypes.

Prerequisite Table: A collection of relationships between updates. An update's prerequisites are specified in **Conjunctive Normal Form (CNF)**; for example, (U6 OR U8) AND (U2) AND (U5 OR U3). A client SHOULD NOT treat a revision as requiring installation unless its prerequisites are satisfied (that is, at least one update in each CNF disjunctive clause is installed on the client). Each entry is uniquely identified by the combination of its RevisionID, PrerequisiteUpdateID, and ClauseID. Each entry MUST include the following elements:

- **RevisionID:** A reference to a revision that has a prerequisite relationship on other updates.

- **PrerequisiteUpdateID:** A reference to a revision that is a prerequisite. Only the UpdateID is declared, but the reference MUST be implicitly to the revision with the highest RevisionNumber that has the specified UpdateID.
- **ClauseID:** Specifies the CNF "AND clause" in which the prerequisite should appear.

Bundle Table: A collection of relationships between updates. Bundled updates are specified in CNF form; for example, (R6 OR R8) AND (R2) AND (R5 OR R3). Each entry is uniquely identified by the combination of its RevisionID, BundledRevisionID, and ClauseID. Each entry MUST include the following elements:

- **RevisionID:** A reference to a revision that bundles other revisions.
- **BundledRevisionID:** A reference to a bundled revision.
- **ClauseID:** Specifies the CNF "AND clause" in which the bundled revision should appear. The ClauseID is used to group together the PrerequisiteUpdateIDs in a given disjunctive clause.

TargetGroup Table: A collection of named groups of clients. Each entry is uniquely identified by a TargetGroupName, the only element in each entry.

Client Table: A collection of clients. Each entry is uniquely identified by a ClientID. Each entry MUST include the following elements.

- **ClientID:** A unique identifier for the client. MAY be implemented as the ClientID passed to the GetAuthorizationCookie (section 2.2.2.1) method.<26>
- **TargetGroupName:** A reference to the target group to which this client belongs.

Deployment Table: A collection of administrator-defined specifications that state that specific revisions SHOULD be made available to specific clients. Each entry is uniquely identified by the combination of its RevisionID and TargetGroupName. Each entry MUST include the following elements.

- **RevisionID:** A reference to the deployed revision.
- **TargetGroupName:** A reference to the target group to which the revision should be deployed.
- **Action:** The action that clients in the specified target group should perform on this revision: Install, Uninstall, PreDeploymentCheck (which means do not offer it; just report back on the status), or Evaluate (which means do not offer it and do not report back on the status).
- **LastChangeTime:** The time the deployment was last modified.
- **DownloadPriority:** Specifies whether the client should download the revisions content as high-, medium-, or low-priority relative to other content needed by the client.
- **Deadline:** An optionally specified time by which clients SHOULD perform the deployment action.

3.1.1.1 Populating the Data Model

The server implementation needs to extract information for the data model from the update metadata. Except as specified below, the update metadata does not need to be interpreted by the server. Because the metadata is well-formed XML, the properties specified below can all be extracted using XPATH queries, as specified in [XPATH].

Revision Table

There MUST be one entry in the revision table for each metadata revision in the server. The RevisionID element MUST be populated with a unique server-assigned value. The remaining elements MUST be populated from the revision metadata using the following unqualified XPATHs.

Property	XPATH
UpdateID	/Update/UpdateIdentity/@UpdateID
RevisionNumber	/Update/UpdateIdentity/@RevisionNumber
UpdateType	/Update/Properties/@UpdateType

Driver Table

There MUST be at least one entry in the driver table for each entry in the revision table with UpdateType=Driver. The RevisionID element MUST be populated with a reference to the associated entry in the revision table. The remaining elements MUST be populated from the revision metadata using the following unqualified XPATHs.

Property	XPATH
HardwareID	/Update/ApplicabilityRules/Metadata/WindowsDriverMetaData/@HardwareID
DriverVerDate	/Update/ApplicabilityRules/Metadata/WindowsDriverMetaData/@DriverVerDate
DriverVerVersion	/Update/ApplicabilityRules/Metadata/WindowsDriverMetaData/@DriverVerVersion
Class	/Update/ApplicabilityRules/Metadata/WindowsDriverMetaData/@Class
Manufacturer	/Update/ApplicabilityRules/Metadata/WindowsDriverMetaData/@Manufacturer
Provider	/Update/ApplicabilityRules/Metadata/WindowsDriverMetaData/@Provider
Model	/Update/ApplicabilityRules/Metadata/WindowsDriverMetaData/@Model
WhqlDriverID	/Update/ApplicabilityRules/Metadata/WindowsDriverMetaData/@WhqlDriverID

Prerequisite Table

There MUST be one entry in the prerequisite table for each prerequisite relationship declared in the revision metadata. Each entry MUST have the RevisionID element populated with a reference to entry in the revision table corresponding to the revision metadata declaring the prerequisite relationship. The PrerequisiteUpdateID MUST be populated by using one of the following unqualified XPATHs.

Property	XPATH
PrerequisiteUpdateID	/Update/Relationships/Prerequisites/AtLeastOne/UpdateIdentity/@UpdateID
PrerequisiteUpdateID	/Update/Relationships/Prerequisites/UpdateIdentity/@UpdateID

The server-assigned ClauseID element MUST be the same between different entries in the table if and only if:

- The entries have the same RevisionID element, and

- The entries were populated from the same AtLeastOne XML element from the revision metadata as defined by the following XPATH: /Update/Relationships/Prerequisites/AtLeastOne/UpdateIdentity/@UpdateID.

Bundle Table

There MUST be one entry in the bundle table for each bundle relationship declared in the revision metadata. Each entry MUST have the RevisionID element populated with a reference to the entry in the revision table corresponding to the revision metadata that declares the prerequisite relationship.

There MUST be one entry in the table for each node returned from the following XPATH query:

/Update/Relationships/BundledUpdates/AtLeastOne/UpdateIdentity

The entry MUST have:

RevisionID: Set to the RevisionID of the metadata declaring the bundle relationship.

BundledRevisionID: Set to the RevisionID of the entry in the revision table whose UpdateID and RevisionNumber match the values in the XPATH query above.

ClauseID: This server-assigned element must be the same between different entries in the table if and only if:

- The entries have the same RevisionID element, and
- The entries were populated from the same AtLeastOne XML element from the revision metadata as defined by the following XPATH: /Update/Relationships/ BundledUpdates/AtLeastOne/UpdateIdentity.

Metadata Table

The metadata table is populated with metadata fragments that MUST be generated from the original metadata. For each revision, the server MUST generate the following entries in the metadata table.

Core: There MUST be exactly one "Core" entry created from the revisions metadata.

- **RevisionID:** References the entry in the revision table for the revisions metadata.
- **FragmentType:** Core.
- **Locale:** NULL.
- **XmlFragment:** MUST be derived from the original metadata by:
  - Collecting the following XmlNodes:
    - The XmlNode identified by XPATH /Update/UpdateIdentity
    - The XmlNode identified by XPATH /Update/Properties, all attributes removed except: UpdateType, ExplicitlyDeployable, AutoSelectOnWebSites, and EulaID
    - The XmlNode identified by XPATH /Update/Relationships
    - The XmlNode identified by XPATH /Update/ApplicabilityRules
  - Then, for each XmlNode, it MUST transform element names as follows:

        - Append "b." to any element in namespace http://schemas.microsoft.com/msus/2002/12/BaseApplicabilityRules
        - Append "m." to any element in namespace http://schemas.microsoft.com/msus/2002/12/MsiApplicabilityRules
        - Append "d." to any element in namespace http://schemas.microsoft.com/msus/2002/12/UpdateHandlers/WindowsDriver
    - Next, strip all namespace definitions.
    - Finally, concatenate the XmlNodes to form the XmlFragment.

      Note The resulting fragment is not well-formed XML.
- **Extended:** There MUST be exactly one "Extended" entry created from the revisions metadata.
    - **RevisionID:** References the entry in the revision table for the revisions metadata.
    - **FragmentType:** Extended.
    - **Locale:** NULL.
    - **XmlFragment:** MUST be derived from the original metadata by concatenating the following strings together after removing all XML namespace definitions from each string (the result of which is not well-formed XML):
        - The XmlNode identified by XPATH /Update/Properties, with the following attributes removed: UpdateType, ExplicitlyDeployable, AutoSelectOnWebSites, EulaID, PublicationState, PublisherID, CreationDate, IsPublic, LegacyName, DetectoidType.
        - The XmlNode identified by XPATH /Update/Files.
        - The XmlNode identified by XPATH /Update/HandlerSpecificData.
- **LocalizedProperties:** There are one or more "LocalizedProperties" entries that MUST be created from the revisions metadata: one for each XmlNode, N, with unqualified XPATH /Update/LocalizedPropertiesCollection/LocalizedProperties.
    - RevisionID: References the entry in the revision table for the revisions metadata.
    - FragmentType: LocalizedProperties.
    - XmlFragment: Strip namespaces from the XmlNode.
    - Locale: The "Language" sub-element of the XmlNode.
- **Eula:** There MUST be zero or more "Eula" entries created from the revisions metadata: one for each XmlNode, N, with unqualified XPATH /Update/LocalizedPropertiesCollection/EulaFile.
    - **RevisionID:** References the entry in the revision table for the revisions metadata.
    - **FragmentType:** Eula.
    - **XmlFragment:** Strip namespaces from the XmlNode.
    - **Locale:** The "Language" attribute of the XmlNode.

Note Clients don't use the EULA fragments obtained from a Windows Server Update Services (WSUS) server (this fragment is only used by clients talking to the Windows Update service). Instead, a Windows Server Update Services (WSUS) server administrator must accept EULAs on behalf of clients.

Client Table

This protocol does not mandate a specific mechanism for populating this table. Implementations may do so via administrative configuration or at runtime, or via other means.<27>

Target Group Table

This protocol does not mandate a specific mechanism for populating this table. Implementations may do so via administrative configuration, or via other means.<28>

Deployment Table

This protocol does not mandate a specific mechanism for populating this table. Implementations may do so via administrative configuration, or via other means.<29>

3.1.2 Timers

None. All protocol requests are initiated by the client.

3.1.3 Initialization

The following initialization steps MUST be performed.

1. All tables in the abstract data model contain persistent data that MUST be retrieved from persistent storage at initialization time.
2. Each Web service within the server MUST begin listening for requests at the respective URL addresses given in the message transport (as specified in section 2.1).
3. The self-update tree and content directory MUST be made available at the URL address given in the message transport (as specified in section 2.1).

3.1.4 Higher-Layer Triggered Events

There are no higher-layer triggered events. All protocol requests are initiated by the client.

3.1.5 Message Processing and Sequencing Rules

The following high-level sequence diagram illustrates the operation of the protocol.

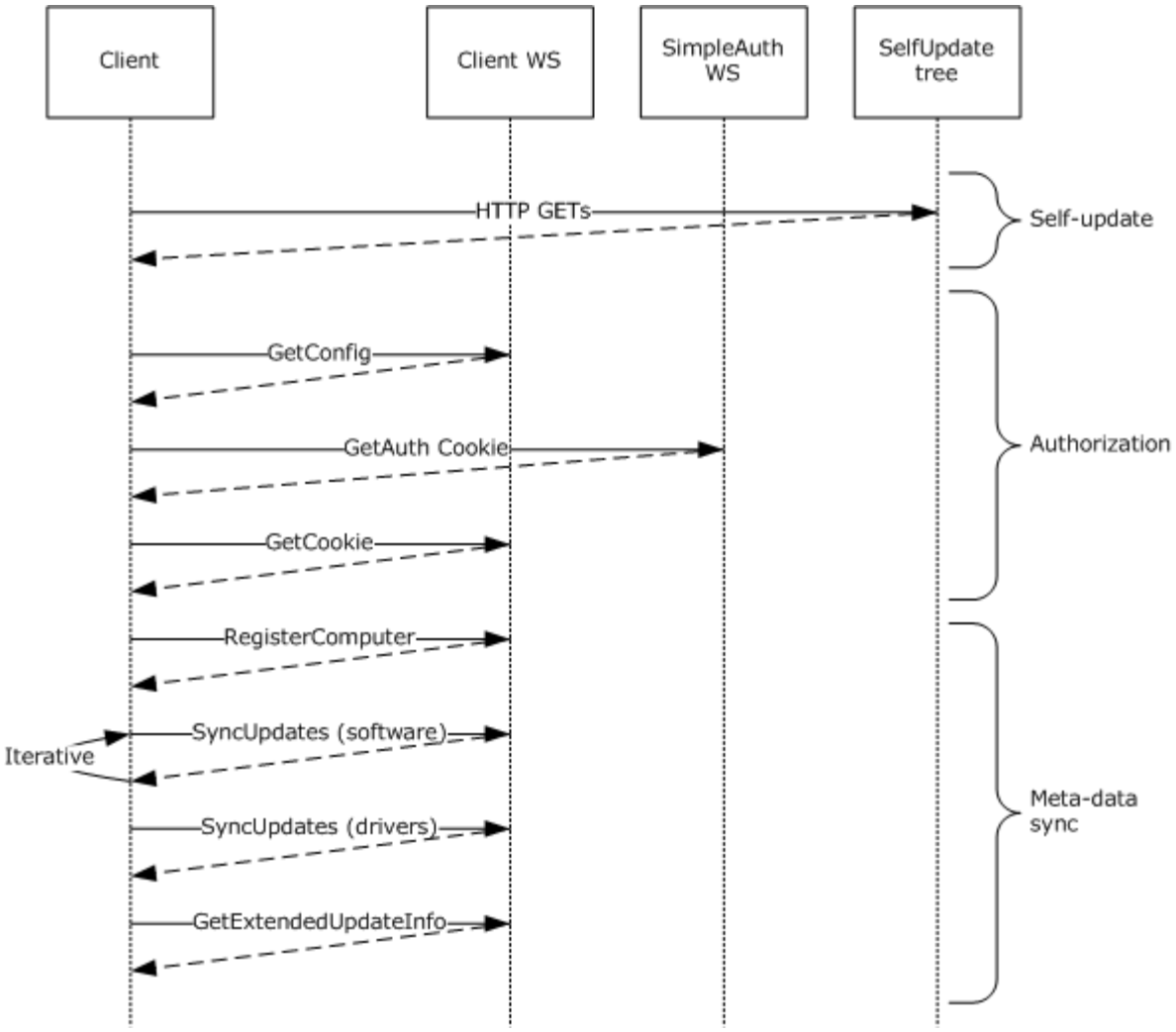

Figure 3: Message processing sequence of the Windows Update Services: Client-Server Protocol

Self-update, authorization, and metadata sync MUST always be performed in the sequence illustrated in this diagram, although specific steps in the sequence MAY be omitted as an optimization. Content download and event reporting SHOULD be performed asynchronously from other operations. Each of these operations is specified in more detail in the following sections.

3.1.5.1 Self-Update

At the start of the protocol, the client MUST check if it needs to self-update. Client/server communications fail if the server does not expose the self-update tree, as specified in this section.

As specified in section 2, the server implementation MUST expose the self-update tree as a virtual directory. The client issues HTTP GET requests (as specified in [RFC2616] section 9.3) to obtain files from the self-update tree; therefore, the server MUST support HTTP requests on this virtual directory.

3.1.5.2 GetConfig

Synopsis:

This method returns configuration information on the server. It MUST be the initial method called by the client when connecting to the server. The client SHOULD cache the results of this call and only call it again if the server throws a SoapFault with one of the following ErrorCodes: InvalidCookie, ServerChanged, or ConfigChanged.<30>

Request Validation:

Parameter	Validation conditions	ErrorCode
protocolVersion	MUST be a non-NULL two-part version string (for example, "1.0").	InvalidParameters

Results:

If no faults occur during the operation, the server MUST return a GetConfigResponse message in response to the client.

3.1.5.3 GetAuthorizationCookie

Synopsis:

This method provides a mechanism for the server to authenticate and authorize client access to the client Web service.

The client SHOULD only call this method if it needs to renew its cookie, either because the cookie expired or because the server has thrown a SoapFault with one of the following ErrorCodes: InvalidCookie, ServerChanged, ConfigChanged, or CookieExpired.<31>

Request Validation:

Parameter	Validation conditions	Error code
clientID	MUST be a non-NULL string.	InvalidParameters
dnsName	MUST be a non-NULL string consistent with DNS naming requirements.	InvalidParameters

Data processing:

The server MAY persist information about the client as a result of this request.<32>

Results:

If no faults occur during the operation, the server MUST return a GetAuthorizationCookieResponse message to the client.<33>

3.1.5.4 GetCookie

Synopsis:

This method is called by the client during the initial dialog between the client and the server. It MUST be called after GetConfig (section 3.1.5.2) and GetAuthorizationCookie (section 3.1.5.3), but before calling any other operation. The cookie returned by the method MUST be passed to all

subsequent operations in the client Web service. The client SHOULD cache the cookie and call this method only if it needs to renew the cached cookie, either because the cookie expired, or because the server has thrown a SoapFault with one of the following ErrorCodes: InvalidCookie, ServerChanged, ConfigChanged, or CookieExpired.<34>

Request Validation:

Parameter	Validation conditions	Error code
authCookies	MUST contain exactly one Authorization Cookie—the one returned from GetAuthorizationCookie (section 3.1.5.3).	InvalidAuthorizationCookie
oldCookie	MUST either be NULL or contain a valid cookie issued by this server (typically an expired cookie).	InvalidCookie or ServerChanged
lastChange	MUST be the same as the last time the configuration changed on the server (although this check can be done in SyncUpdates (section 3.1.5.6) instead).	ConfigChanged
protocolVersion	SHOULD be a non-NULL two-part version string (for example, "1.0").	InvalidParameters

Data Processing:

The server SHOULD store client state information in the returned cookie.<35>

The server MAY copy state information from the oldCookie element to the new cookie it returns from this method.<36>

Results:

If no faults occur during the operation, the server MUST return a GetCookieResponse message to the client.<37>

3.1.5.5 RegisterComputer

Synopsis:

The client calls this method to pass information about itself to the server. The client MUST only call this method if the server has returned IsRegistrationRequired=true from the client's most recent preceding call to GetConfig (section 3.1.5.2). The client SHOULD only call this method again if any of its registration information has changed, or if the server has requested it be called again by throwing a RegistrationRequired ErrorCode while processing a subsequent method invocation.<38>

Request Validation:

Parameter	Validation conditions	Error code
cookie	MUST be a valid cookie, issued by this server, that has not expired.	InvalidCookie, ServerChanged, or CookieExpired
computerInfo	MAY perform validation of these fields.	InvalidParameters

Note Validation of the computerInfo data is optional, because this data is purely informational and is not used anywhere else in the client/server protocol.

Data Processing:

The server MAY store client registration info on the server.<39>

Results:

If no faults occur during the operation, the server MUST return a RegisterComputerResponse message to the client.

3.1.5.6 SyncUpdates

Synopsis:

This is the main operation that supports the synchronization of update metadata to client computers. It is invoked to perform both software and driver metadata synchronization. Software update synchronization MUST be performed first, using a sequence of calls to this method. Driver synchronization MUST then be performed using a single call to the method.

Request Validation:

Parameter	Validation conditions	Error code
cookie	MUST be a valid cookie, issued by this server, that has not expired.	InvalidCookie, ServerChanged, or CookieExpired
Parameters	MUST be specified.	InvalidParameters

Additional checks the server MUST perform:

- If Parameters.SystemSpec is present and Parameters.SkipSoftwareSync is FALSE, throw an InvalidParameters ErrorCode.

Data Processing:

The data processing specified in this section references most of the elements of the abstract data model, as specified in section 3.1.1.

The server MUST check whether the configuration data returned from GetConfig (section 3.1.5.2) has changed since the last time the client synchronized and, if so, throw a ConfigChanged ErrorCode fault.

The server SHOULD check whether client registration is required but the client is not yet registered. If so, it should throw a RegistrationRequired ErrorCode.<40>

The following rules for driver matching MUST be implemented by the server. Given any two drivers in the driver table as specified in the Abstract Data Model, the server MUST conclude that one of the two is a "better" match for a device listed in the system specification if:

1. It has a better HardwareID match: the one driver HardwareID matches on a device HardwareID or CompatibleID that is listed earlier in the device XML, or
2. Both HardwareIDs are equal matches, but the one driver has a more recent DriverVerDate, or
3. Both HardwareIDs are equal matches and both have the same DriverVerDate, but the one driver has a higher DriverVerVersion (after converting the four-part version string to a 64-bit integer), or
4. Both HardwareIDs are equal matches, both have the same DriverVerDate, and both have the same DriverVerVersion, but the one driver has a higher WhqlDriverID.

Given a collection of drivers and a particular device listed in the system spec, the "best" driver for that device is determined by applying the above rules repeatedly to each pair-wise combination of drivers, discarding any driver not deemed "better" in a given pair-wise combination. The last remaining driver is the "best" match.

Given the above rules, the next step is for the server to compute the NeededRevisions list for the client. The server MUST do so as follows:

1. Restrict the set of revisions to those that are deployed to the client computer's target group, combined with any dependencies (prerequisite or bundle) of such updates.

2. Restrict the resulting set further to those revisions whose prerequisites are satisfied by the updates whose Revision IDs are specified in Parameters.InstalledNonLeafUpdateIDs.

3. Restrict the resulting set further to either:

    - If performing software update synchronization (SkipSoftwareSync = false): revisions with UpdateType = Software.

    - If performing driver synchronization (SkipSoftwareSync = true): revisions for which all the following conditions hold:

        - UpdateType = Driver, and

        - The revision has an entry in the driver table that MUST be the "best" match for one of the devices in the system spec, and

        - If there is already a driver installed on the device:

            - The revision has an entry in the driver table that MUST be a "better" match than the installed driver, and

            - If the installed driver is a printer (Class='Printer'), then the revision MUST have an entry in the driver table which matches the Provider and Manufacturer for the installed driver.

Next, the server MUST generate the list of CachedRevisions for the client as follows:

- If performing software synchronization (SkipSoftwareSync = false), take the union of revisions in Parameters.InstalledNonLeafUpdateIDs and Parameters.OtherCachedUpdateIDs.

- If performing driver synchronization (SkipSoftwareSync = true), take the revisions listed in Parameters.CachedDriverIDs.

Results:

If no faults occur during the operation, the server MUST return a SyncUpdatesResponse message to the client. It MUST generate the response as follows:

- **SyncUpdatesResponse.NewUpdates:** Populated with entries for revision in the NeededRevisions list that are not in the CachedRevisions list:

    - **ID:** The Revision ID.

    - **Deployment:**

      Information about the deployment to this revision. If this revision was not itself explicitly deployed to the client by an administrator (for example, it was included in the NeededRevisions list because it was a dependency of an explicitly deployed revision), the

DeploymentAction MUST be set to "Evaluate". For driver updates (UpdateType = driver), when the client reports a protocolVersion of "1.6" or higher in the GetCookie call, the server SHOULD include all the HardwareIDs associated with this revision from the driver table that are selected as "best" matches.<41>

- **IsLeaf:** Specifies whether the revision is a leaf on the prerequisite graph or not. In particular, that there are no entries in the abstract data model Prerequisite table (as specified in section 3.1.1) that have this revisions UpdateID specified as a PrerequisiteUpdateID.
- **Xml:** The revision's associated "core" metadata (FragmentType = "Core").

- **SyncUpdatesResponse.OutOfScopeRevisionIDs:** Populated with the IDs of revision that are in the CachedRevisions list that are not in the NeededRevisions list.
- **SyncUpdatesResponse.ChangedUpdates:** Populated with entries for revisions in the NeededRevisions list that are also in the CachedRevisions list, but for which Deployment or IsLeaf data has changed since the last time the client synchronized with the server. The fields of these entries are populated according to the server's abstract data model (as specified in section 3.1.1) as follows:
  - **Deployment**: The entry in the deployment table that specifies how the revision is deployed to the client's target group.
  - **IsLeaf:** The entry in the Revision table that specifies whether the revision is a leaf in the prerequisite graph.
  - **SyncUpdatesResponse.Truncated:** The server MAY choose to return a subset of the updates that would normally be returned in the NewUpdates collection, in order to reduce the processing overhead incurred by a single call to the server. In such cases, the server MUST set Truncated = true.<42>
  - **SyncUpdatesResponse.NewCookie:** The server MUST return a new cookie for the client to use on subsequent SyncUpdates calls.<43>

3.1.5.7 RefreshCache

Synopsis:

This method is called by the client to refresh the Revision IDs it uses to reference its cache revisions when it initiates synchronization with a new server.

The Windows Update Services: Client-Server Protocol uses revision IDs to identify revisions. Because revision IDs are generated by each server, the client MUST invoked this method to update the revision ID identifiers for each revision in its cache whenever it changes servers. This method MUST only be called by the client in response to the server throwing a ServerChanged or InvalidCookie ErrorCode.

Request Validation:

Parameter	Validation conditions	Error code
cookie	MUST be a valid cookie, issued by this server, that has not expired.	InvalidCookie, ServerChanged, or CookieExpired

Results:

If no faults occur during the operation, the server MUST return a RefreshCacheResponse message to the client.

The response consists of an array of RefreshCacheResult elements. There MUST be one element for each *GlobalID* parameter for which an entry exists in the deployment table from that update to the client's target group. Each such element MUST contain:

RevisionID: The revision ID referenced by the entry in the deployment.

GlobalID: The *GlobalID* parameter.

IsLeaf: The IsLeaf element in the revision table referenced by the revision ID.

Deployment: The entry in the deployment table that specifies the revision ID and the client's TargetGroupName.

3.1.5.8 GetExtendedUpdateInfo

Synopsis:

To optimize client/server communications, the SyncUpdates (section 3.1.5.6) method only downloads the "Core" metadata associated with an update revision. Additional metadata and the URLs of the revision content MUST be obtained by using this method.

Request Validation:

Parameter	Validation conditions	Error code
cookie	MUST be a valid cookie, issued by this server, that has not expired.	InvalidCookie, ServerChanged, or CookieExpired
revisionIDs	MUST be smaller than the maximum request value that the server specified in the return value of GetConfig().	InvalidParameter

Response:

Upon successful completion, the server MUST return a GetExtendedUpdateInfo message to the client. The message MUST be composed as follows:

- **OutOfScopeRevisionIDs:** The IDs of requested revisions that are not deployed to the client.
- **Updates:** Metadata fragments associated with the requested revision that match one of the requested infoTypes.
- **FileLocations:** For each requested revision that is deployed to the client, the URLs (in the content directory) of the updates-associated content.<44>

3.1.5.9 GetFileLocations

Synopsis:

This method is called when the client needs to update its cached FileLocation information. A client SHOULD only invoke this method in response to a FileLocationsChanged ErrorCode.<45>

Request Validation:

Parameter	Validation conditions	Error code
cookie	MUST be a valid cookie, issued by this server, that has not expired.	InvalidCookie, ServerChanged, or CookieExpired

Response:

Upon successful completion, the server MUST return a GetFileLocationsResponse message to the client. The message MUST include file locations for content matching the specified SHA1 hashes, although the file location itself is implementation-specific.<46>

3.1.5.10 ReportEventBatch

Synopsis:

This Web method is invoked by clients to report events to the server.

Request Validation:

Parameter	Validation conditions	Error code
cookie	MUST be a valid cookie, issued by this server, that has not expired.	InvalidCookie, ServerChanged, or CookieExpired

Data Processing:

All processing of reported events is implementation-specific.

Response:

If no faults occur during the operation, the server MUST return a ReportEventBatchResponse message to the client.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

This section describes the logical structure, components, and event handlers of this protocol.

3.2.1 Abstract Data Model

This section describes a conceptual model and possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as the external behavior is consistent with that described in this document.

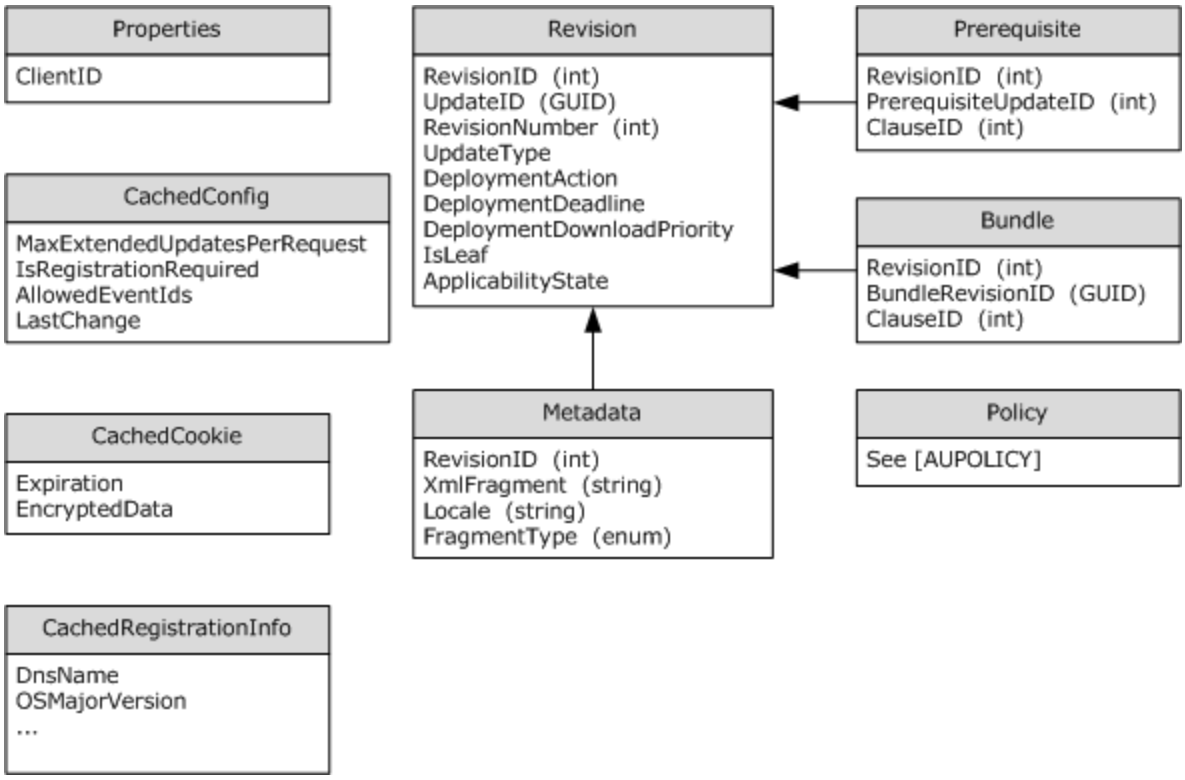

Figure 4: Server abstract data model

Prerequisite table: Has the same definition as specified in section 3.1.1.

Bundle table: Has the same definition as specified in section 3.1.1.

Metadata table: Has the same definition as specified in section 3.1.1.

Deployment table: Has the same definition as specified in section 3.1.1, but adds the following additional elements that MUST be present:

- **DeploymentAction:** Has the same definition as the Deployment table's Action element as specified in section 3.1.1.
- **DeploymentDeadline:** Has the same definition as the Deployment table's Deadline element as specified in section 3.1.1.
- **DeploymentDownloadPriority:** Has the same definition as the Deployment table's DownloadPriority element as specified in section 3.1.1.
- **IsLeaf:** Specifies whether this revision is a leaf in the server's prerequisite graph. This value is returned to the client in the SyncUpdates (section 3.1.5.6) method.
- **ApplicabilityState:** The client evaluates whether the revision is needed, installed, or not applicable during metadata synchronization. The client accomplishes this by evaluating the applicability rules in the "core" metadata fragment (although the format of the applicability rules is opaque to the client-server protocol). The client obtains the core revision metadata from the server's SyncUpdates (section 3.1.5.6) method. The client re-evaluates the applicability state of

all revisions before each sync. The applicability state of the revisions affects how the client passes the parameters to the SyncUpdates (section 3.1.5.6) method.

Revision Table: A collection of entries corresponding to the revisions available on the server. Each entry is uniquely identified by an update ID and revisions number, and is also independently uniquely identified by a revision ID. Each entry MUST include the following elements.

- **RevisionID:** A server-assigned revision identifier. Because it is more compact than the globally unique (UpdateID, RevisionNumber) pair, the RevisionID is used as the revision identifier during client-server communications in order to minimize network traffic.
- **UpdateID:** A globally unique identifier that is the same for all revisions of an update.
- **RevisionNumber:** Used in conjunction with the UpdateID to uniquely identify the revision.
- **UpdateType:** Specifies whether the update is a Driver, Software, Category, or Detectoid. The SyncUpdates (section 3.1.5.6) method treats drivers and non-drivers differently as specified in section 3.1.5.6.

Properties Table: Persistently stored properties used by the client. There is exactly one entry, which MUST include the following element.

- **ClientID:** A globally unique string that the client generates to uniquely identify itself to the server.

CachedConfig table: The client caches the return value from GetConfig (section 3.1.5.2) so that it can avoid calling this method on every sync.

CachedCookie table: The cookie returned from GetCookie (section 3.1.5.4), SyncUpdates (section 3.1.5.6), or GetFileLocations (section 3.1.5.9). There is at most one entry, which MUST include the following elements.

- **Expiration:** A clear-text copy of the time the cookie expires.
- **CookieData:** An opaque sequence of one or more bytes containing server-implementation–specific authorization, authentication, and protocol state information.

CachedRegistrationInfo Table: The last set of parameters passed to the server's RegisterComputer (section 3.1.5.5) method. There is at most one entry, which MUST include one element for each parameter passed to the RegisterComputer method.

Policy Table: The client may support additional configuration properties that control its behavior.<47>

3.2.2 Timers

There are no timers required by the Windows Update Services: Client-Server Protocol.<48>

3.2.3 Initialization

On its first initialization, each client MUST assign itself a globally-unique ClientID string. The resulting ClientID string MUST be stored in a persistent storage location for use in subsequent protocol operations. The algorithm used by the client to create the ClientID is implementation-specific.<49>

3.2.4 Higher-Layer Triggered Events

The only higher-layer triggered events required by the protocol are the generation of the reporting events as specified in the EventID table of ReportEventBatch (section 2.2.4.1). The client SHOULD send these events up after a small random delay in order to allow events to be batched together, which improves network, client, and server performance.<50>

Other higher-layer triggers are up to the implementation.<51>

3.2.5 Message Processing and Sequencing Rules

The "metadata sync" portion of this protocol conforms to the figure as specified in section 3.1.5.

Self-update, authorization, and metadata sync MUST be performed in the sequence shown in the figure as specified in section 3.1.5, although, as an optimization, certain steps MAY be omitted under the conditions specified below. In particular, the client MAY perform the following optimizations.

- The result of GetConfig (section 3.1.5.2) SHOULD be cached on the client. Subsequent synchronizations SHOULD omit this method invocation unless the server throws a ConfigChanged ErrorCode.
- The cookie returned from GetCookie (section 3.1.5.4) or SyncUpdates (section 3.1.5.6) MUST be cached on the client. Subsequent synchronizations SHOULD omit the call to GetAuthorizationCookie (section 3.1.5.3) and GetCookie unless the server throws a CookieExpired ErrorCode, or unless the client determines that the clear-text copy of the cookie expiration time shows the cookie to be expired.
- The registration information sent to the server SHOULD be cached on the client. Subsequent synchronizations SHOULD omit the call to

  RegisterComputer (section 3.1.5.5), unless the registration information has changed on the client, or unless the server throws a RegistrationRequired exception.<52>

In addition to the standard sync path specified above, the client MUST also perform special recovery steps when the server returns certain application-level faults, as specified in section 2.2.5.

In addition to metadata sync, the client also initiates content download for applicable updates, installation of those updates after the download completes, and generation of events to report to the server. It is implementation-specific as to when these should be performed.

3.2.6 Timer Events

There are no timer events required by the Windows Update Services: Client-Server Protocol.<53>

3.2.7 Other Local Events

If, at the start of metadata synchronization, the client determines that the registration information it last sent to the server has changed, it SHOULD call the RegisterComputer (section 3.1.5.5) method during metadata synchronization, as specified in section 3.1.5.5.

4 Protocol Examples

```
<!-- S A M P L E   C L I E N T - S E R V E R  C O N V E R S A T I O N -->

<!-- AUTHORIZATION PHASE: GetConfig() Request  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<GetConfig xmlns="http://www.microsoft.com/SoftwareDistribution/
      Server/ClientWebService">
<protocolVersion>1.0</protocolVersion>
</GetConfig>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: GetConfig() Response - Success  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetConfigResponse xmlns="http://www.microsoft.com/SoftwareDistribut
ion/Server/ClientWebService">
      <GetConfigResult>
        <LastChange>2007-01-31T04:16:35.24Z</LastChange>
        <IsRegistrationRequired>true</IsRegistrationRequired>
        <AuthInfo>
          <AuthPlugInInfo>
            <PlugInID>SimpleTargeting</PlugInID>

<ServiceUrl>SimpleAuthWebService/SimpleAuth.asmx</ServiceUrl>
            <Parameter />
          </AuthPlugInInfo>
        </AuthInfo>
        <Properties>
          <ConfigurationProperty>
            <Name>MaxExtendedUpdatesPerRequest</Name>
            <Value>50</Value>
          </ConfigurationProperty>
          <ConfigurationProperty>
            <Name>PackageServerShare</Name>
            <Value>\\microsof-
cd0710.redmond.corp.microsoft.com\UpdateServicesPackages</Value>
          </ConfigurationProperty>
          <ConfigurationProperty>
            <Name>ProtocolVersion</Name>
            <Value>3.0</Value>
          </ConfigurationProperty>
          <ConfigurationProperty>
            <Name>IsInventoryRequired</Name>
            <Value>0</Value>
          </ConfigurationProperty>
          <ConfigurationProperty>
            <Name>ClientReportingLevel</Name>
            <Value>2</Value>
          </ConfigurationProperty>
        </Properties>
      </GetConfigResult>
    </GetConfigResponse>
  </soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: GetAuthorizationCookie() Request  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
```

```
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<GetAuthorizationCookie
 xmlns="http://www.microsoft.com/SoftwareDistribution/Server/SimpleAuthWebService">
<clientId>5c7f4f80-3896-4d10-8a38-469286a0febc</clientId>
<targetGroupName />
<dnsName>microsof-cd0710.redmond.corp.microsoft.com</dnsName>
</GetAuthorizationCookie>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: GetAuthorizationCookie() Response - Success  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetAuthorizationCookieResponse xmlns="http://www.microsoft.com/SoftwareDistribut
     ion/Server/SimpleAuthWebService">
<GetAuthorizationCookieResult>
<PlugInId>SimpleTargeting</PlugInId>
<CookieData>cXwHhVfNGxd/aJZ9ZqE4zwDbISp2BEDW1uTJ9CUX5N5adk8+If+RLCCC9hq335lESmH
    aHAmAkAgTr53kDAHesvDrgOtT39tzQjbqwBE+d+WI1bCisLMcxIzXyvZIONiAYcEnb0xNkximY6b4Cs
    rVkXqd6cFdMvWlEg70++f2CEgck23j3jW+ak5LE4Yjf/WnbwUFuYkZvUF3xedDVclrSj2xorvdcPMnX
    ieg0zMnCT1hzWACmSNDYR1kgDu4t9/ScE/Y8AwIBZS9d/+OzNUA/Ae0LeQtuqSQMHB+XEHu5bN255tt
    f2Lwib8qE0DkD9gVDqSIYeChI5i/17zIc+9ZDsITO+evN30wR3d48yLZkj5PkubRh5K0Ni5ugehov2e
    FSGO24t5o5miYBiLC6HR/Urmr9m4EuJvyCRfT95voBXnWS4JtbKz/Dcn614SpMemZF8KMlFgRwDe+kG
    rSmnrHqoKvxyR/km/HRt1FN2NuLOO+VxwxSQ0Ion9hQ9E346qR4dAU9TcdOxBlaOJw5y44o+0q+WRlI
    if7fzGXsvy7ibvW9Rnn4LmLpMZT4haAPxM7qPdFQU+AMJK9IniOVHTm+26WaTSd6Ezhe6WcJPU6ymEA
    8T10FYNkuULLllTL2HD44TbhMzLsXTsjtl2zVcTk7eEOnN/o+0kZX+oCnkHxtS8k6kqqQPBYJu34uVw
    pjuW6</CookieData>
</GetAuthorizationCookieResult>
</GetAuthorizationCookieResponse>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: GetCookie() Request  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org
 /2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<GetCookie xmlns="http://www.microsoft.com/SoftwareDistribution/Server/ClientWebS
    ervice">
<authCookies xmlns:q1="http://www.microsoft.com/SoftwareDistribution/Server/Clie
     ntWebService" soapenc:arrayType="q1:AuthorizationCookie[1]">
<AuthorizationCookie>
<PlugInId>SimpleTargeting</PlugInId>
<CookieData>cXwHhVfNGxd/aJZ9ZqE4zwDbISp2BEDW1uTJ9CUX5N5adk8+If+RLCCC9hq335lESm
     HaHAmAkAgT r53kDAHesvDrgOtT39tzQjbqwBE+d+WI1bCisLMcxIzXyvZIONiAYcEnb0xNkximY6b
     4CsrVkXqd 6cFdMvWlEg70++f2CEgck23j3jW+ak5LE4Yjf/WnbwUFuYkZvUF3xedDVclrSj2xorvd
     cPMnXieg 0zMnCT1hzWACmSNDYR1kgDu4t9/ScE/Y8AwIBZS9d/+OzNUA/Ae0LeQtuqSQMHB+XEHu5
     bN255tt f2Lwib8qE0DkD9gVDqSIYeChI5i/17zIc+9ZDsITO+evN30wR3d48yLZkj5PkubRh5K0Ni
     5ugeho v2eFSGO24t5o5miYBiLC6HR/Urmr9m4EuJvyCRfT95voBXnWS4JtbKz/Dcn614SpMemZF8K
     MlFgR wDe+kGrSmnrHqoKvxyR/km/HRt1FN2NuLOO+VxwxSQ0Ion9hQ9E346qR4dAU9TcdOxBlaOJw
     5y44 o+0q+WRlIif7fzGXsvy7ibvW9Rnn4LmLpMZT4haAPxM7qPdFQU+AMJK9IniOVHTm+26WaTSd6
     Ezh e6WcJPU6ymEA8T10FYNkuULLllTL2HD44TbhMzLsXTsjtl2zVcTk7eEOnN/o+0kZX+oCnkHxtS
     8k 6kqqQPBYJu34uVwpjuW6</CookieData>
</AuthorizationCookie>
</authCookies>
<oldCookie>
<Expiration>2006-05-16T18:54:28.85Z</Expiration>
<EncryptedData xsi:nil="1" />
</oldCookie>
<lastChange>2006-05-16T18:54:28.85Z</lastChange>
```

```
<currentTime>2006-05-17T16:16:38Z</currentTime>
<protocolVersion>1.0</protocolVersion>
</GetCookie>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: GetCookie() Response - Success  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http
://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetCookieResponse xmlns="http://www.microsoft.com/SoftwareDistribution/Server/Clie
    ntWebService">
<GetCookieResult>
<Expiration>2006-05-26T16:58:01.8749472Z</Expiration>
<EncryptedData>Y41m+GJEYODokhVx8U56ON1R7zAaKLtLMykG5g5pOveyMPUJ4XKTUm/XPSrLz/OxDe
     AH2/ZVi2HMzl/dclPbemJzsa3NymBvcBW1XBAJNCmMIwt80Bo2prJoCG1tj2k1XMFf0kXWBueYaBmJdd
     09PQ/upwDElfGSaqVA0zfxpZ0nmLPfkq199+HP8l2eOjtY0N3aYlOUbJg09UecBkKhH0TpiAT50m4Jey
     ChR2DVdxGFgsYW0M7ToiElMkeh4lbErkrYBTczYiHKyapXZbk5pg==</EncryptedData>
</GetCookieResult>
</GetCookieResponse>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: GetCookie() Response - SOAP Fault - ConfigChanged  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/X
 MLSchema">
<soap:Body>
<soap:Fault>
<faultcode>soap:Client</faultcode>
<faultstring>System.Web.Services.Protocols.SoapException: Fault occurred at M
    icrosoft.UpdateServices.Internal.SoapUtilities.ThrowException(ErrorCode erro
    rCode, String message) at Microsoft.UpdateServices.Internal.ClientImplementa
    tion.GetCookie(AuthorizationCookie[] authCookies, Cookie oldCookie, DateTime
     lastChange, DateTime currentClientTime, String protocolVersion) at Microsof
    t.UpdateServices.Internal.Client.GetCookie(AuthorizationCookie[] authCookies
    , Cookie oldCookie, DateTime lastChange, DateTime currentTime, String protoc
    olVersion)</faultstring>
<faultactor>http://hemantn-srv:8530/ClientWebService/client.asmx</faultactor>
<detail>
<ErrorCode>ConfigChanged</ErrorCode>
<Message />
<ID>2587c8c1-4f54-4033-9f1a-3a195bf0495d</ID>
<Method>"http://www.microsoft.com/SoftwareDistribution/Server/
        ClientWebService/GetCookie"</Method>
</detail>
</soap:Fault>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: GetCookie() Response - SOAP Fault - InvalidCookie  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XM
LSchema">
<soap:Body>
<soap:Fault>
<faultcode>soap:Client</faultcode>
<faultstring>System.Web.Services.Protocols.SoapException: Fault occurred at M
    icrosoft.UpdateServices.Internal.SoapUtilities.ThrowException(ErrorCode erro
    rCode, String message) at Microsoft.UpdateServices.Internal.Authorization.Au
    thorizationManager.DecryptOldCookie(Cookie oldCookie) at Microsoft.UpdateSer
    vices.Internal.Authorization.AuthorizationManager.GetCookie(AuthorizationCoo
    kie[] authCookies, Cookie oldCookie, DateTime lastChange, DateTime currentCl
    ientTime, String clientProtocolVersion) at Microsoft.UpdateServices.Internal
    .ClientImplementation.GetCookie(AuthorizationCookie[] authCookies, Cookie ol
```

```
    dCookie, DateTime lastChange, DateTime currentClientTime, String protocolVer
    sion) at Microsoft.UpdateServices.Internal.Client.GetCookie(AuthorizationCoo
    kie[] authCookies, Cookie oldCookie, DateTime lastChange, DateTime currentTi
    me, String protocolVersion)</faultstring>
<faultactor>http://MyUpdateServer:8530/ClientWebService/
      client.asmx</faultactor>
<detail>
<ErrorCode>InvalidCookie</ErrorCode>
<Message />
<ID>34f57c46-0f67-45ed-b3d5-90bd2e8a1e87</ID>
<Method>"http://www.microsoft.com/SoftwareDistribution/Server/ClientWebServi
     ce/GetCookie"</Method>
</detail>
</soap:Fault>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: RegisterComputer() Request  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="
 http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XML
 Schema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <RegisterComputer
xmlns="http://www.microsoft.com/SoftwareDistribution/Server/ClientWebService">
      <cookie>
        <Expiration>2007-02-06T03:08:47.9843753Z</Expiration>

<EncryptedData>QNmzcgoarx4NjlUDjxZmtwZBqxczAszEMbYFOI3Ud/zeFNc1RZ2L/HXS
H7VdVYAgo0+svC7MwMplSsU14c7nkbStGb91OV5fuLWMdPGATzRNO2Z6P8wXwVpX129YSpg
0n9kiRG17RL+S+uUdgByFCKT+TUvxCbUSvsDwym3+wcHoips2CPnqRDJLXPjDbGzDvFGIRo
27f4HETOVHGuHNErOQj0miMYCRGDX42mQklvNgDiYaJSC1tikOFfHTnG6mVweI4A+cmgmg0
cer8yasZA==</EncryptedData>
      </cookie>
      <computerInfo>
        <DnsName>microsof-cd0710.redmond.corp.microsoft.com</DnsName>
        <OSMajorVersion>5</OSMajorVersion>
        <OSMinorVersion>2</OSMinorVersion>
        <OSBuildNumber>3790</OSBuildNumber>
        <OSServicePackMajorNumber>1</OSServicePackMajorNumber>
        <OSServicePackMinorNumber>0</OSServicePackMinorNumber>
        <OSLocale>en-US</OSLocale>
        <ComputerManufacturer>Microsoft
Corporation</ComputerManufacturer>
        <ComputerModel>Virtual Machine</ComputerModel>
        <BiosVersion>080002</BiosVersion>
        <BiosName>BIOS Date: 08/14/03 19:41:02 Ver: 08.00.02</BiosName>
        <BiosReleaseDate>2003-08-14T00:00:00Z</BiosReleaseDate>
        <ProcessorArchitecture>x86</ProcessorArchitecture>
        <SuiteMask>272</SuiteMask>
        <OldProductType>3</OldProductType>
        <NewProductType>0</NewProductType>
        <SystemMetrics>0</SystemMetrics>
        <ClientVersionMajorNumber>7</ClientVersionMajorNumber>
        <ClientVersionMinorNumber>0</ClientVersionMinorNumber>
        <ClientVersionBuildNumber>6000</ClientVersionBuildNumber>
        <ClientVersionQfeNumber>317</ClientVersionQfeNumber>
      </computerInfo>
    </RegisterComputer>
  </soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: RegisterComputer() Response  -->


<!-- AUTHORIZATION PHASE: RefreshCache() Request ==> After getting an InvalidCoo
```

```
 kie SOAP Fault on GetCookie, new auth phase followed by this  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XM
LSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<RefreshCache xmlns="http://www.microsoft.com/SoftwareDistribution/Server/Clie
   ntWebService">
<cookie>
<Expiration>2006-05-23T03:18:56Z</Expiration>
<EncryptedData>JTaRKKkZAjLUqC8SaVvfYrrNigA8ICtNu1po4umDm6aXpgEo91QPg5wa8u9+q
     ecS3SkW42bZnI6n Sw9j/r4SEeA4nD4IyZbpqSNTeBEbdQbfkS8ZfbqkzPzVcWZ16QY4Mq2JBhJ
     exvqAv711twO5bZjF yb57rG7eSnFIas2J06tvCnQ9l1x+nJWtlFO+gU70BuZNfaviHvCj3ByaD
     9M33eLPaq0cZzKNYt/F mvAW9ddl5L6a9GTL4+eV1raIY/sb7ABevwIX4jOsnXSd9Mt8cw==</E
     ncryptedData>
</cookie>
<globalIDs xmlns:q1="http://www.microsoft.com/SoftwareDistribution/Server/Cli
    entWebService" soapenc:arrayType="q1:UpdateIdentity[7]">
<UpdateIdentity>
<UpdateID>0a4c6c73-8887-4d7f-9cbe-d08fa8fa9d1e</UpdateID>
<RevisionNumber>50</RevisionNumber>
</UpdateIdentity>
<UpdateIdentity>
<UpdateID>352f9494-d516-4b40-a21a-cd2416098982</UpdateID>
<RevisionNumber>51</RevisionNumber>
</UpdateIdentity>
<UpdateIdentity>
<UpdateID>5c9376ab-8ce6-464a-b136-22113dd69801</UpdateID>
<RevisionNumber>1</RevisionNumber>
</UpdateIdentity>
<UpdateIdentity>
<UpdateID>7c40e8c2-01ae-47f5-9af2-6e75a0582518</UpdateID>
<RevisionNumber>1</RevisionNumber>
</UpdateIdentity>
<UpdateIdentity>
<UpdateID>a4bedb1d-a809-4f63-9b49-3fe31967b6d0</UpdateID>
<RevisionNumber>100</RevisionNumber>
</UpdateIdentity>
<UpdateIdentity>
<UpdateID>7145181b-9556-4b11-b659-0162fa9df11f</UpdateID>
<RevisionNumber>50</RevisionNumber>
</UpdateIdentity>
<UpdateIdentity>
<UpdateID>5cc25303-143f-40f3-a2ff-803a1db69955</UpdateID>
<RevisionNumber>1</RevisionNumber>
</UpdateIdentity>
</globalIDs>
</RefreshCache>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: RefreshCache() Response  -->


<!-- AUTHORIZATION PHASE: GetFileLocations() Request  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="
 http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XML
 Schema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<GetFileLocations xmlns="http://www.microsoft.com/SoftwareDistribution/Server/C
   lientWebService">
<cookie>
<Expiration>2006-05-23T03:18:56Z</Expiration>
<EncryptedData>JTaRKKkZAjLUqC8SaVvfYrrNigA8ICtNu1po4umDm6aXpgEo91QPg5wa8u9+qe
     cS3SkW42bZnI6n Sw9j/r4SEeA4nD4IyZbpqSNTeBEbdQbfkS8ZfbqkzPzVcWZ16QY4Mq2JBhJex
     vqAv711twO5bZjF yb57rG7eSnFIas2J06tvCnQ9l1x+nJWtlFO+gU70BuZNfaviHvCj3ByaD9M3
```

```
     3eLPaq0cZzKNYt/F mvAW9ddl5L6a9GTL4+eV1raIY/sb7ABevwIX4jOsnXSd9Mt8cw==</Encry
     ptedData>
</cookie>
<fileDigests soapenc:arrayType="xsd:base64Binary[1]">
<base64Binary>AAAAAAAAAAAAAAAAAAAAAAAAAAA=</base64Binary>
</fileDigests>
</GetFileLocations>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: GetFileLocations() Response  -->


<!-- AUTHORIZATION PHASE: SyncUpdates() Request - Very First from WinXP  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="
 http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XML
 Schema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<SyncUpdates xmlns="http://www.microsoft.com/SoftwareDistribution/Server/Client
   WebService">
<cookie>
<Expiration>2006-05-23T04:06:32.9843746</Expiration>
<EncryptedData>vS39KAEKKzRInaR4GVchjeaj4kJ3GEFqsmQE6fQGdxntnGxs+PPNTGQqqnBrJT
     JTgSlQl3aZsax5 I/bpT+u017LSMUnuR68LyMQXmk9EikRsYbZto2NEVowAXMRe5B8bmlTXFKj4v
     gVYqhyq7Yas57FP WjTOAjtLMUVB9vdoWeqwq7PDwwZJNqhhvqmYU3z0aSnK6G1rQ4zv765bmqb5
     Mti9Dp/16ODVZEic 4Q8IzX0V2Rb7BOk7PA7RMdXMbBDbZ6ZTU0eMebi6tzVp+Qu0eA==</Encry
     ptedData>
</cookie>
<parameters>
<ExpressQuery>false</ExpressQuery>
<InstalledNonLeafUpdateIDs xsi:nil="1" />
<OtherCachedUpdateIDs xsi:nil="1" />
<SystemSpec xsi:nil="1" />
<CachedDriverIDs xsi:nil="1" />
<SkipSoftwareSync>false</SkipSoftwareSync>
</parameters>
</SyncUpdates>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: SyncUpdates() Request - Second Request from WinXP after R
 esponse from the previous one  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="ht
 tp://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSche
 ma" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<SyncUpdates xmlns="http://www.microsoft.com/SoftwareDistribution/Server/ClientWe
   bService">
<cookie>
<Expiration>2006-05-23T04:06:32.9843746Z</Expiration>
<EncryptedData>aOEIdlPlPHBDVMJgwkbRuDeInZcJr/v272AcY/kDqTghmncGveoB80UQI9jzAQY6
     N9Hio13440Bb UAtx640sUkvw8lFh+0rupjSFg/KWq4CSpUoESWlfKItW+RnYS5bc7T/5ORSr+7BOf
     8Xy6SP3EkuR qDm2Y/AXphfLoJhSXW1AifcyLuzQf5VOBoyYjIL7elkKt1JKMHCfyAeiHEy3Si6a0r
     gu/2endOWn 5CLxgmlDlfsmmewhk/omaKr2Lk8rnBMf5Qiid76jvMwnbO7PeA==</EncryptedData>
</cookie>
<parameters>
<ExpressQuery>false</ExpressQuery>
<InstalledNonLeafUpdateIDs soapenc:arrayType="xsd:int[1]">
<int>109</int>
</InstalledNonLeafUpdateIDs>
<OtherCachedUpdateIDs soapenc:arrayType="xsd:int[29]">
<int>101</int>
<int>102</int>
<int>103</int>
<int>104</int>
```

```
<int>105</int>
<int>106</int>
<int>107</int>
<int>108</int>
<int>110</int>
<int>111</int>
<int>112</int>
<int>113</int>
<int>114</int>
<int>115</int>
<int>378</int>
<int>379</int>
<int>380</int>
<int>381</int>
<int>382</int>
<int>383</int>
<int>384</int>
<int>385</int>
<int>386</int>
<int>387</int>
<int>388</int>
<int>389</int>
<int>390</int>
<int>391</int>
<int>392</int>
</OtherCachedUpdateIDs>
<SystemSpec xsi:nil="1" />
<CachedDriverIDs xsi:nil="1" />
<SkipSoftwareSync>false</SkipSoftwareSync>
</parameters>
</SyncUpdates>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: SyncUpdates() Request - Third Request from WinXP after R
 esponse from the previous one  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="h
 ttp://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSc
 hema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<SyncUpdates xmlns="http://www.microsoft.com/SoftwareDistribution/Server/ClientW
   ebService">
<cookie>
<Expiration>2006-05-23T04:06:32.9843746Z</Expiration>
<EncryptedData>ae5VXUP5lEJygdVCP7NnJFmZVzmsbo8agdMfdqYe03xJfNw/P2SpXL/RtaBlQzc
     TTLn8SUarYVne kc1XU/c5MP+7MNdW1b9xERCy+Vm/CXNFluJ64veTPuVGgJ8wtlIAcsEHVmCNbhw
     UXnb1wS/YOnUC 3OmcXdwQcVRWAASxepjBaF/Y8em5KGn37XW75D0cSCVdmUts+uGMgQ2QYze5AuC
     WDAqpHf0x2jSb 8QfL2Ygljs44gLGOIlN40E72tLpQuIe3yn9C4lAg0VKdbAsieA==</Encrypted
     Data>
</cookie>
<parameters>
<ExpressQuery>false</ExpressQuery>
<InstalledNonLeafUpdateIDs soapenc:arrayType="xsd:int[2]">
<int>109</int>
<int>191</int>
</InstalledNonLeafUpdateIDs>
<OtherCachedUpdateIDs soapenc:arrayType="xsd:int[58]">
<int>39</int>
<int>40</int>
<int>41</int>
<int>42</int>
<int>43</int>
<int>44</int>
<int>45</int>
<int>46</int>
```

```
<int>47</int>
<int>48</int>
<int>49</int>
<int>50</int>
<int>51</int>
<int>52</int>
<int>53</int>
<int>54</int>
<int>55</int>
<int>56</int>
<int>57</int>
<int>58</int>
<int>97</int>
<int>99</int>
<int>100</int>
<int>101</int>
<int>102</int>
<int>103</int>
<int>104</int>
<int>105</int>
<int>106</int>
<int>107</int>
<int>108</int>
<int>110</int>
<int>111</int>
<int>112</int>
<int>113</int>
<int>114</int>
<int>115</int>
<int>179</int>
<int>181</int>
<int>196</int>
<int>197</int>
<int>198</int>
<int>210</int>
<int>378</int>
<int>379</int>
<int>380</int>
<int>381</int>
<int>382</int>
<int>383</int>
<int>384</int>
<int>385</int>
<int>386</int>
<int>387</int>
<int>388</int>
<int>389</int>
<int>390</int>
<int>391</int>
<int>392</int>
</OtherCachedUpdateIDs>
<SystemSpec xsi:nil="1" />
<CachedDriverIDs xsi:nil="1" />
<SkipSoftwareSync>false</SkipSoftwareSync>
</parameters>
</SyncUpdates>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: SyncUpdates() Request - Fourth Request from WinXP after R
 esponse from the previous one  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="ht
 tp://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSche
 ma" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
```

```
<SyncUpdates xmlns="http://www.microsoft.com/SoftwareDistribution/Server/ClientWe
   bService">
<cookie>
<Expiration>2006-05-23T04:06:32.9843746Z</Expiration>
<EncryptedData>ae5VXUP5lEJygdVCP7NnJFmZVzmsbo8agdMfdqYe03xJfNw/P2SpXL/RtaBlQzcT
     TLn8SUarYVne kc1XU/c5MP+7MNdW1b9xERCy+Vm/CXNFluJ64veTPuVGgJ8wtlIAcsEHVmCNbhwUX
     nb1wS/YOnUC 3OmcXdwQcVRWAASxepjBaF/Y8em5KGn37XW75D0cSCVdmUts+uGMgQ2QYze5AuCWDA
     qpHf0x2jSb 8QfL2Ygljs44gLGOIlN40E72tLpQuIe3yn9C4lAg0VKdbAsieA==</EncryptedData>
</cookie>
<parameters>
<ExpressQuery>false</ExpressQuery>
<InstalledNonLeafUpdateIDs soapenc:arrayType="xsd:int[2]">
<int>109</int>
<int>191</int>
</InstalledNonLeafUpdateIDs>
<OtherCachedUpdateIDs soapenc:arrayType="xsd:int[58]">
<int>39</int>
<int>40</int>
<int>41</int>
<int>42</int>
<int>43</int>
<int>44</int>
<int>45</int>
<int>46</int>
<int>47</int>
<int>48</int>
<int>49</int>
<int>50</int>
<int>51</int>
<int>52</int>
<int>53</int>
<int>54</int>
<int>55</int>
<int>56</int>
<int>57</int>
<int>58</int>
<int>97</int>
<int>99</int>
<int>100</int>
<int>101</int>
<int>102</int>
<int>103</int>
<int>104</int>
<int>105</int>
<int>106</int>
<int>107</int>
<int>108</int>
<int>110</int>
<int>111</int>
<int>112</int>
<int>113</int>
<int>114</int>
<int>115</int>
<int>179</int>
<int>181</int>
<int>196</int>
<int>197</int>
<int>198</int>
<int>210</int>
<int>378</int>
<int>379</int>
<int>380</int>
<int>381</int>
<int>382</int>
<int>383</int>
```

```
<int>384</int>
<int>385</int>
<int>386</int>
<int>387</int>
<int>388</int>
<int>389</int>
<int>390</int>
<int>391</int>
<int>392</int>
</OtherCachedUpdateIDs>
<SystemSpec xsi:nil="1" />
<CachedDriverIDs xsi:nil="1" />
<SkipSoftwareSync>false</SkipSoftwareSync>
</parameters>
</SyncUpdates>
</soap:Body>
</soap:Envelope><!-- AUTHORIZATION PHASE: SyncUpdates() Response  -->


<!-- AUTHORIZATION PHASE: GetCookie() Request  -->


<!-- AUTHORIZATION PHASE: GetCookie() Response  -->


<!-- AUTHORIZATION PHASE: GetCookie() Request  -->


<!-- AUTHORIZATION PHASE: GetCookie() Response  -->


<!-- AUTHORIZATION PHASE: ReportEventBatch() Request - EventID=148  -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="ht
 tp://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSche
 ma" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<ReportEventBatch xmlns="http://www.microsoft.com/SoftwareDistribution">
<cookie>
<Expiration>2006-05-17T17:16:37.9843748Z</Expiration>
<EncryptedData>QNmzcgoarx4NjlUDjxZmtwZBqxczAszEMbYFOI3Ud/zeFNc1RZ2L/HXSH7VdVYAg
     o0+svC7MwMpl SsU14c7nkbStGb91OV5fuLWMdPGATzRNO2Z6P8wXwVpX129YSpg0n9kiRG17RL+S+
     uUdgByFCKT+ TUvxCbUSvsDwym3+wcHoips2CPnqRDJLXPjDbGzDvFGIRo27f4HETOVHGuHNErOQj0
     miMYCRGDX4 2mQklvNgDiYaJSC1tikOFfHTnG6mVweI4A+cmgmg0cer8yasZA==</EncryptedData>
</cookie>
<clientTime>2006-05-17T16:16:38.015</clientTime>
<eventBatch xmlns:q1="http://www.microsoft.com/SoftwareDistribution" soapenc:arr
    ayType="q1:ReportingEvent[2]">
<ReportingEvent>
<BasicData>
<TargetID>
<Sid>5c7f4f80-3896-4d10-8a38-469286a0febc</Sid>
</TargetID>
<SequenceNumber>0</SequenceNumber>
<TimeAtTarget>2006-05-17T16:13:29.734</TimeAtTarget>
<EventInstanceID>E6D82915-627F-418B-A5CC-B9FCD400455B</EventInstanceID>
<NamespaceID>1</NamespaceID>
<EventID>148</EventID>
<SourceID>101</SourceID>
<UpdateID>
<UpdateID>D67661EB-2423-451D-BF5D-13199E37DF28</UpdateID>
<RevisionNumber>0</RevisionNumber>
</UpdateID>
<Win32HResult>-2145107943</Win32HResult>
<AppName>SelfUpdate</AppName>
</BasicData>
```

```
<ExtendedData>
<ReplacementStrings soapenc:arrayType="xsd:string[1]">
<string>0x80244019</string>
</ReplacementStrings>
<MiscData soapenc:arrayType="xsd:string[5]">
<string>Q=1</string>
<string>G=7.0.5378.45</string>
<string>J=703</string>
<string>K=EPP runtime BIOS - Version 1.1</string>
<string>L=2005-11-22T00:00:00</string>
</MiscData>
<ComputerBrand>Hewlett-Packard</ComputerBrand>
<ComputerModel>HP Compaq nx6125 (EM493UC#ABA)</ComputerModel>
<BiosRevision>68DTT Ver. F.0D</BiosRevision>
<ProcessorArchitecture>X86Compatible</ProcessorArchitecture>
<OSVersion>
<Major>5</Major>
<Minor>1</Minor>
<Build>2600</Build>
<Revision>65792</Revision>
<ServicePackMajor>2</ServicePackMajor>
<ServicePackMinor>0</ServicePackMinor>
</OSVersion>
<OSLocaleID>1033</OSLocaleID>
<DeviceID />
</ExtendedData>
<PrivateData>
<ComputerDnsName />
<UserAccountName />
</PrivateData>
</ReportingEvent>
<ReportingEvent>
<BasicData>
<TargetID>
<Sid>5c7f4f80-3896-4d10-8a38-469286a0febc</Sid>
</TargetID>
<SequenceNumber>0</SequenceNumber>
<TimeAtTarget>2006-05-17T16:15:11.171</TimeAtTarget>
<EventInstanceID>3F5E26A3-4BF8-4E25-9D3F-9D9C420E3D43</EventInstanceID>
<NamespaceID>1</NamespaceID>
<EventID>148</EventID>
<SourceID>101</SourceID>
<UpdateID>
<UpdateID>D67661EB-2423-451D-BF5D-13199E37DF28</UpdateID>
<RevisionNumber>0</RevisionNumber>
</UpdateID>
<Win32HResult>-2145107943</Win32HResult>
<AppName>SelfUpdate</AppName>
</BasicData>
<ExtendedData>
<ReplacementStrings soapenc:arrayType="xsd:string[1]">
<string>0x80244019</string>
</ReplacementStrings>
<MiscData soapenc:arrayType="xsd:string[5]">
<string>Q=1</string>
<string>G=7.0.5378.45</string>
<string>J=703</string>
<string>K=EPP runtime BIOS - Version 1.1</string>
<string>L=2005-11-22T00:00:00</string>
</MiscData>
<ComputerBrand>Hewlett-Packard</ComputerBrand>
<ComputerModel>HP Compaq nx6125 (EM493UC#ABA)</ComputerModel>
<BiosRevision>68DTT Ver. F.0D</BiosRevision>
<ProcessorArchitecture>X86Compatible</ProcessorArchitecture>
<OSVersion>
```

```
<Major>5</Major>
<Minor>1</Minor>
<Build>2600</Build>
<Revision>65792</Revision>
<ServicePackMajor>2</ServicePackMajor>
<ServicePackMinor>0</ServicePackMinor>
</OSVersion>
<OSLocaleID>1033</OSLocaleID>
<DeviceID />
</ExtendedData>
<PrivateData>
<ComputerDnsName />
<UserAccountName />
</PrivateData>
</ReportingEvent>
</eventBatch>
</ReportEventBatch>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: ReportEventBatch() Request - EventID=147, 156 -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="ht
 tp://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSche
 ma" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<ReportEventBatch xmlns="http://www.microsoft.com/SoftwareDistribution">
<cookie>
<Expiration>2006-05-23T04:06:32.9843746Z</Expiration>
<EncryptedData>pEJ+Mq0mpyyWk8c4LHBlmHO1vCdZqcwgO+v4lvmAdPnUce/+FZ3Zm/Z2a7vJ6Q3
     kFq8wiygisS6r EqD9kD14aUtN2fPPhstSDI0f7IEKPrmoKpZ9Ub62olCDp/QaUWTcHsDeEgzA5+D
     iDotyfXs0bNkc plTR44TgJo52m+s+IKj+OhoEyxlPY4q8rzAQ1/9PIwQ2IDKykZRR97etYlrG0JK
     OnJOsoo0b6UyR 29SdCo5UWBkk4ddKFvPYzc1hW/DuOH2x8n4Erw0XrQVqZB1shQ==</Encrypted
     Data>
</cookie>
<clientTime>2006-05-23T03:12:14.859</clientTime>
<eventBatch xmlns:q1="http://www.microsoft.com/SoftwareDistribution" soapenc:ar
    rayType="q1:ReportingEvent[2]">
<ReportingEvent>
<BasicData>
<TargetID>
<Sid>5c7f4f80-3896-4d10-8a38-469286a0febc</Sid>
</TargetID>
<SequenceNumber>0</SequenceNumber>
<TimeAtTarget>2006-05-23T03:09:45.828</TimeAtTarget>
<EventInstanceID>D61E5EE1-968B-4162-88BE-BCEA05C5992F</EventInstanceID>
<NamespaceID>1</NamespaceID>
<EventID>147</EventID>
<SourceID>101</SourceID>
<UpdateID>
<UpdateID>00000000-0000-0000-0000-000000000000</UpdateID>
<RevisionNumber>0</RevisionNumber>
</UpdateID>
<Win32HResult>0</Win32HResult>
<AppName>AutomaticUpdates</AppName>
</BasicData>
<ExtendedData>
<ReplacementStrings soapenc:arrayType="xsd:string[1]">
<string>0</string>
</ReplacementStrings>
<MiscData soapenc:arrayType="xsd:string[6]">
<string>D=0</string>
<string>Q=1</string>
<string>G=7.0.5378.51</string>
<string>J=703</string>
<string>K=EPP runtime BIOS - Version 1.1</string>
```

```
<string>L=2005-11-22T00:00:00</string>
</MiscData>
<ComputerBrand>Hewlett-Packard</ComputerBrand>
<ComputerModel>HP Compaq nx6125 (EM493UC#ABA)</ComputerModel>
<BiosRevision>68DTT Ver. F.0D</BiosRevision>
<ProcessorArchitecture>X86Compatible</ProcessorArchitecture>
<OSVersion>
<Major>5</Major>
<Minor>1</Minor>
<Build>2600</Build>
<Revision>65792</Revision>
<ServicePackMajor>2</ServicePackMajor>
<ServicePackMinor>0</ServicePackMinor>
</OSVersion>
<OSLocaleID>1033</OSLocaleID>
<DeviceID />
</ExtendedData>
<PrivateData>
<ComputerDnsName />
<UserAccountName />
</PrivateData>
</ReportingEvent>
<ReportingEvent>
<BasicData>
<TargetID>
<Sid>5c7f4f80-3896-4d10-8a38-469286a0febc</Sid>
</TargetID>
<SequenceNumber>0</SequenceNumber>
<TimeAtTarget>2006-05-23T03:09:45.828</TimeAtTarget>
<EventInstanceID>07B6BD18-BC34-4458-8FDA-D517E3500272</EventInstanceID>
<NamespaceID>1</NamespaceID>
<EventID>156</EventID>
<SourceID>101</SourceID>
<UpdateID>
<UpdateID>00000000-0000-0000-0000-000000000000</UpdateID>
<RevisionNumber>0</RevisionNumber>
</UpdateID>
<Win32HResult>0</Win32HResult>
<AppName>AutomaticUpdates</AppName>
</BasicData>
<ExtendedData>
<ReplacementStrings xsi:nil="1" />
<MiscData soapenc:arrayType="xsd:string[6]">
<string>V=E45B26C2-278F-48F3-97D1-AA0FD57423D5;6917A042-BA50-4723-A0C3-3783
        A8DDC28B;BBE805EE-2CF7-4110-9BE8-4FE32B144E5A;9A127D52-45FA-44BA-ACC5-5EA9
        A637FE4E;BB758AAA-8024-44D0-8434-73CEFE8CA80B;4B689082-CA9C-46CC-B2FC-720F
        EDE21299;3A7EFDB0-D88D-42A6-8439-C7D0B1BCAC82;B20C7AC9-F972-480C-9601-DE04
        86342506;1FD3B3AC-9E83-4286-8369-D122669C24F6;C4B891EE-36A1-DDB6-2F9F-C7B4
        E58990A3;B166A6A9-398B-4C81-B8DB-2043CF0672A1;4CDB577B-FDD1-49D4-87CC-3808
        AD20ACBC;9306CDFC-C4A1-4A22-9996-848CB67EDDC3;5BC6E116-9964-4452-AE5A-C54B
        D709BC53;1E59F963-12F5-4BFD-A73D-366921779C28;D3AC165E-D7C4-4BDF-83F0-E249
        ECBE873B;DD167F04-69E8-46A0-83DA-D4A5143EDBF1;AADB419D-3C09-408D-9D64-20A4
        51CBD5BC;D80D91CF-6361-4FDF-B35D-F05077902973;2FCDEFD7-B98F-4A25-8DF4-F806
        0AFC0942;1AFB1AFE-9AAF-4DC4-9DAC-C75303A84655;70E702CB-5D44-494D-8594-476A
        C774CD51;06F7520C-2CF3-46AB-A457-402327016632;66216830-74FA-4A91-BEA8-212E
        6A5A43DD;3AD5B256-D0BB-4391-8E3C-3EE605D1A7F1;E0A7FC7E-FF6E-4559-990B-4B4A
        01F0AD39;EE49A276-D4AE-4C9E-829E-7901D10055E0;9197DADA-DE1C-420F-B558-E701
        8602E960;B00FE76D-8C87-48A2-9D5D-FD2D1336698C;AE76FA20-B794-4E9C-8554-7A00
        C8153E27;6D5B814E-201C-4F9B-965A-6672F85CEB9A;923E52D0-B0B7-48A5-9C4D-C074
        1AD4B53F;88ECB294-6967-4250-BCB7-8C37B83915A8;B5503E2C-ADAF-494B-B3F7-15E8
        93F84273;5E7DD70B-B694-4D0C-9146-14C08F4BAE3B;0E23ED43-08B0-4EB6-8539-1449
        C29FB131;41A25FE0-F119-4AA4-AFC6-86A1CCA0F195;FF8EC90C-6D23-48EB-8158-BAE4
        696563DC;9B390F2B-845A-4C1C-9964-4C49D1788D00;58313F8C-4976-48AA-9537-79E4
        C1BFEBE0;3B936005-0C22-4DD7-90BB-51793D6D2E57;9C0CFA0F-28F6-4A26-AD6C-61FB
        34130873;015984CC-2265-47B6-B255-1818F9937F20;F4D3F81B-6CB8-4501-A316-9A62
```

```
        16001FEC;C94437C9-5C05-47C9-9891-424A9C01D1FF;8C6B7628-95CF-4450-8F60-136A
        70B930C5;9608001E-2D54-4C54-B795-ACBCAC1A9930</string>
<string>Q=1</string>
<string>G=7.0.5378.51</string>
<string>J=703</string>
<string>K=EPP runtime BIOS - Version 1.1</string>
<string>L=2005-11-22T00:00:00</string>
</MiscData>
<ComputerBrand>Hewlett-Packard</ComputerBrand>
<ComputerModel>HP Compaq nx6125 (EM493UC#ABA)</ComputerModel>
<BiosRevision>68DTT Ver. F.0D</BiosRevision>
<ProcessorArchitecture>X86Compatible</ProcessorArchitecture>
<OSVersion>
<Major>5</Major>
<Minor>1</Minor>
<Build>2600</Build>
<Revision>65792</Revision>
<ServicePackMajor>2</ServicePackMajor>
<ServicePackMinor>0</ServicePackMinor>
</OSVersion>
<OSLocaleID>1033</OSLocaleID>
<DeviceID />
</ExtendedData>
<PrivateData>
<ComputerDnsName />
<UserAccountName />
</PrivateData>
</ReportingEvent>
</eventBatch>
</ReportEventBatch>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: ReportEventBatch() Request - EventID=183, 202, 147, 156 -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http:
 //www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xm
 lns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<ReportEventBatch xmlns="http://www.microsoft.com/SoftwareDistribution">
<cookie>
<Expiration>2006-05-23T07:10:57Z</Expiration>
<EncryptedData>tWB+wl7EII8S0t3g14U/ZzYZBZMMNs799VoKHe3nvDSOjTHL0b1PT9T1ODoD4fSKkwq
     6SLwNL35c FprXzFyZVYJ62wX3cSjGtTHkn1OlSHx0hLBXuUxpt6EQAFNRWorX2niWD7KDK8l9tUOXMCD
     0k59j BjjWnUTKWl+U4s4r/ZjoSMrdxXLW+9zSOv1YY2oCM4WpCL/3KapdjWkPDG3gwDzF8hNECd3dUBA
     6 FaUzdClEo0hPGvX3Un0SWDLl+C9sOraRb5ogogOy351TgrQQyA==</EncryptedData>
</cookie>
<clientTime>2006-05-23T06:17:10.839</clientTime>
<eventBatch xmlns:q1="http://www.microsoft.com/SoftwareDistribution" soapenc:arrayT
    ype="q1:ReportingEvent[4]">
<ReportingEvent>
<BasicData>
<TargetID>
<Sid>0f6d43f3-8a2e-4313-99a6-71558f67f436</Sid>
</TargetID>
<SequenceNumber>0</SequenceNumber>
<TimeAtTarget>2006-05-23T06:10:58.306</TimeAtTarget>
<EventInstanceID>83626623-594A-4B8F-B60D-FCE0618EEA30</EventInstanceID>
<NamespaceID>1</NamespaceID>
<EventID>183</EventID>
<SourceID>101</SourceID>
<UpdateID>
<UpdateID>D67661EB-2423-451D-BF5D-13199E37DF28</UpdateID>
<RevisionNumber>0</RevisionNumber>
</UpdateID>
<Win32HResult>0</Win32HResult>
```

```
<AppName>SelfUpdate</AppName>
</BasicData>
<ExtendedData>
<ReplacementStrings soapenc:arrayType="xsd:string[1]">
<string>Automatic Updates</string>
</ReplacementStrings>
<MiscData soapenc:arrayType="xsd:string[6]">
<string>Q=1</string>
<string>G=7.0.5378.51</string>
<string>A=1830</string>
<string>J=465</string>
<string>K=Phoenix ROM BIOS PLUS Version 1.10 A06</string>
<string>L=2006-02-20T00:00:00</string>
</MiscData>
<ComputerBrand>Dell Inc.</ComputerBrand>
<ComputerModel>OptiPlex GX620</ComputerModel>
<BiosRevision>A06</BiosRevision>
<ProcessorArchitecture>X86Compatible</ProcessorArchitecture>
<OSVersion>
<Major>5</Major>
<Minor>2</Minor>
<Build>3790</Build>
<Revision>196882</Revision>
<ServicePackMajor>1</ServicePackMajor>
<ServicePackMinor>0</ServicePackMinor>
</OSVersion>
<OSLocaleID>1033</OSLocaleID>
<DeviceID />
</ExtendedData>
<PrivateData>
<ComputerDnsName />
<UserAccountName />
</PrivateData>
</ReportingEvent>
<ReportingEvent>
<BasicData>
<TargetID>
<Sid>0f6d43f3-8a2e-4313-99a6-71558f67f436</Sid>
</TargetID>
<SequenceNumber>0</SequenceNumber>
<TimeAtTarget>2006-05-23T06:11:43.290</TimeAtTarget>
<EventInstanceID>AEFA4BAA-BAB6-4696-8991-07B2BC766009</EventInstanceID>
<NamespaceID>1</NamespaceID>
<EventID>202</EventID>
<SourceID>102</SourceID>
<UpdateID>
<UpdateID>00000000-0000-0000-0000-000000000000</UpdateID>
<RevisionNumber>0</RevisionNumber>
</UpdateID>
<Win32HResult>0</Win32HResult>
<AppName>AutomaticUpdates</AppName>
</BasicData>
<ExtendedData>
<ReplacementStrings xsi:nil="1" />
<MiscData soapenc:arrayType="xsd:string[6]">
<string>Q=1</string>
<string>G=7.0.5378.51</string>
<string>A=1830</string>
<string>J=465</string>
<string>K=Phoenix ROM BIOS PLUS Version 1.10 A06</string>
<string>L=2006-02-20T00:00:00</string>
</MiscData>
<ComputerBrand>Dell Inc.</ComputerBrand>
<ComputerModel>OptiPlex GX620</ComputerModel>
<BiosRevision>A06</BiosRevision>
```

```
<ProcessorArchitecture>X86Compatible</ProcessorArchitecture>
<OSVersion>
<Major>5</Major>
<Minor>2</Minor>
<Build>3790</Build>
<Revision>196882</Revision>
<ServicePackMajor>1</ServicePackMajor>
<ServicePackMinor>0</ServicePackMinor>
</OSVersion>
<OSLocaleID>1033</OSLocaleID>
<DeviceID />
</ExtendedData>
<PrivateData>
<ComputerDnsName />
<UserAccountName />
</PrivateData>
</ReportingEvent>
<ReportingEvent>
<BasicData>
<TargetID>
<Sid>0f6d43f3-8a2e-4313-99a6-71558f67f436</Sid>
</TargetID>
<SequenceNumber>0</SequenceNumber>
<TimeAtTarget>2006-05-23T06:11:50.525</TimeAtTarget>
<EventInstanceID>640E4DD8-1717-466B-8D78-3E0547DC11F6</EventInstanceID>
<NamespaceID>1</NamespaceID>
<EventID>147</EventID>
<SourceID>101</SourceID>
<UpdateID>
<UpdateID>00000000-0000-0000-0000-000000000000</UpdateID>
<RevisionNumber>0</RevisionNumber>
</UpdateID>
<Win32HResult>0</Win32HResult>
<AppName>AutomaticUpdates</AppName>
</BasicData>
<ExtendedData>
<ReplacementStrings soapenc:arrayType="xsd:string[1]">
<string>0</string>
</ReplacementStrings>
<MiscData soapenc:arrayType="xsd:string[7]">
<string>D=0</string>
<string>Q=1</string>
<string>G=7.0.5378.51</string>
<string>A=1830</string>
<string>J=465</string>
<string>K=Phoenix ROM BIOS PLUS Version 1.10 A06</string>
<string>L=2006-02-20T00:00:00</string>
</MiscData>
<ComputerBrand>Dell Inc.</ComputerBrand>
<ComputerModel>OptiPlex GX620</ComputerModel>
<BiosRevision>A06</BiosRevision>
<ProcessorArchitecture>X86Compatible</ProcessorArchitecture>
<OSVersion>
<Major>5</Major>
<Minor>2</Minor>
<Build>3790</Build>
<Revision>196882</Revision>
<ServicePackMajor>1</ServicePackMajor>
<ServicePackMinor>0</ServicePackMinor>
</OSVersion>
<OSLocaleID>1033</OSLocaleID>
<DeviceID />
</ExtendedData>
<PrivateData>
<ComputerDnsName />
```

```
<UserAccountName />
</PrivateData>
</ReportingEvent>
<ReportingEvent>
<BasicData>
<TargetID>
<Sid>0f6d43f3-8a2e-4313-99a6-71558f67f436</Sid>
</TargetID>
<SequenceNumber>0</SequenceNumber>
<TimeAtTarget>2006-05-23T06:11:50.525</TimeAtTarget>
<EventInstanceID>76484064-8BB7-44C2-86EC-8DB03489B5D1</EventInstanceID>
<NamespaceID>1</NamespaceID>
<EventID>156</EventID>
<SourceID>101</SourceID>
<UpdateID>
<UpdateID>00000000-0000-0000-0000-000000000000</UpdateID>
<RevisionNumber>0</RevisionNumber>
</UpdateID>
<Win32HResult>0</Win32HResult>
<AppName>AutomaticUpdates</AppName>
</BasicData>
<ExtendedData>
<ReplacementStrings xsi:nil="1" />
<MiscData soapenc:arrayType="xsd:string[7]">
<string>V=2D6C4040-E112-4803-89D7-E5C7FB0560D3;9FE4C2B1-5280-4291-838E-D9B6
        3B7143DA;745BEB74-FCB0-4323-ADF3-470A9AF3C615;A9711EF4-689B-4833-975B-0FBF
        38D7236A;5B4AF580-4613-4C09-A7C8-25EA125B9DD5;2FC54BAF-DCA9-41EC-91F0-298E
        26DA6207;0B8645AF-7D19-4687-B103-3390169C50F6;BA825FF2-7B5B-411A-B066-B021
        2DC66425;CDEA1B11-FE38-4967-8402-C0F4E6293842;9766EE68-0156-4898-9BB8-D1BC
        08E0F4B1;56C460FB-3A01-4867-958B-2F4959CB899E;F39ABD66-507F-419A-BB26-9F2D
        AC21EDE9;2542DF13-B081-4F4A-908B-11A404EE251C;5716D3F3-0550-4294-B8E6-64CA
        C43DC876;1FFF7AF8-E5CF-40CE-BB8C-E156F1788EF0;E423E6DE-9555-4580-A922-12A6
        2292C242;58305952-B8B4-4983-8685-09995FBC05E6;BF8C38FB-75B6-4A5B-87C4-A73C
        2169F62D;0E7075AE-C8E0-47B4-AE21-1FD753A7F9A6;B362BBF8-37AD-46F7-A29D-10B2
        337636D7;4B8C1BD1-F2E6-45B5-9211-F6AF0EEDB5A0;3309C6C8-9B2F-4191-A300-98D9
        202D48B1;58313F8C-4976-48AA-9537-79E4C1BFEBE0;A3F4DE35-7701-4579-8500-9172
        07724D85;4AC190F1-6581-46AE-8FFC-56186988D355;7F6BEDB4-9710-4170-A1D4-6D80
        9EF8BAD0;C36EE53C-FE94-417F-BEB4-B997775A040A;D0C919F2-15A0-4738-B0D1-B2FF
        6E908E21</string>
<string>Q=1</string>
<string>G=7.0.5378.51</string>
<string>A=1830</string>
<string>J=465</string>
<string>K=Phoenix ROM BIOS PLUS Version 1.10 A06</string>
<string>L=2006-02-20T00:00:00</string>
</MiscData>
<ComputerBrand>Dell Inc.</ComputerBrand>
<ComputerModel>OptiPlex GX620</ComputerModel>
<BiosRevision>A06</BiosRevision>
<ProcessorArchitecture>X86Compatible</ProcessorArchitecture>
<OSVersion>
<Major>5</Major>
<Minor>2</Minor>
<Build>3790</Build>
<Revision>196882</Revision>
<ServicePackMajor>1</ServicePackMajor>
<ServicePackMinor>0</ServicePackMinor>
</OSVersion>
<OSLocaleID>1033</OSLocaleID>
<DeviceID />
</ExtendedData>
<PrivateData>
<ComputerDnsName />
<UserAccountName />
</PrivateData>
```

```
</ReportingEvent>
</eventBatch>
</ReportEventBatch>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: ReportEventBatch() Response - Success -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="
 http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XML
 Schema">
<soap:Body>
<ReportEventBatchResponse xmlns="http://www.microsoft.com/SoftwareDistribution">
<ReportEventBatchResult>true</ReportEventBatchResult>
</ReportEventBatchResponse>
</soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: ReportEventBatch() Response - SOAP Fault - ConfigChange
 d -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="
 http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XML
 Schema">
<soap:Body>
<soap:Fault>
<faultcode>soap:Client</faultcode>
<faultstring>System.Web.Services.Protocols.SoapException: Fault occurred at Mi
    crosoft.UpdateServices.Internal.SoapUtilities.ThrowException(ErrorCode errorC
    ode, String message) at Microsoft.UpdateServices.Internal.Authorization.Autho
    rizationManager.CrackCookie(Cookie cookie) at Microsoft.UpdateServices.Intern
    al.Reporting.WebService.ReportEventBatch(Cookie cookie, DateTime clientTime,
    ReportingEvent[] eventBatch)</faultstring>
<faultactor>http://hemantn-srv:8530/ReportingWebService/ReportingWebService.as
    mx</faultactor>
<detail>
<ErrorCode>ConfigChanged</ErrorCode>
<Message />
<ID>f83d0ddb-2937-4d0a-a564-2d9a5cb906a4</ID>
<Method>"http://www.microsoft.com/SoftwareDistribution/ReportEventBatch"</Met
     hod>
</detail>
</soap:Fault>
</soap:Body>
</soap:Envelope>
```

5 Security Considerations

Because the server can tell the client to install binaries, care must be taken to prevent a **man-in-the-middle** or other forms of a spoof server telling the client to install binaries that will compromise the client computer. For this reason, the client SHOULD perform several checks:

- It SHOULD only accept content signed by trusted certificates.
- It SHOULD only accept content whose SHA1 hash matches the SHA1 hash specified in the metadata.<54>

As a result, it is strongly recommended that the server be configured so that all metadata communication is done over a Secure Sockets Layer (SSL) port. Using SSL ensures that the client is communicating with the real server and so prevents a spoof server from sending the client harmful requests (for example, to uninstall patches).

Because the WSUS server distributes publicly available patches (from Microsoft Update), client authentication is not a particularly important security consideration. In fact, supporting unauthenticated clients is probably the best approach because in most environments, it is more important to keep all machines patched than it is to deny access to unauthenticated clients.

There are two strategies one can use to reduce the impact of denial-of-service (DOS) attacks against the server:

- Turn on authentication and deny access to unauthenticated clients. This will allow one to quickly disable access to rogue client machines. The downside of this approach, discussed in the section above, is that it means new clients might not get patched by default.
- Make sure no single operation takes too much processing time on the server. That will ensure that any attacker must keep up a steady stream of requests to deny access to the server, and so a simple network trace will allow one to identify the offending machine and shut it down. This applies to requests sent by "spoof clients" (for example, a virus emulating a client, which might try to pass an unbounded set of parameters to various methods).

If the server implementation stores and displays any data passed to it from clients (for example, DnsName or BiosName), care must be taken to ensure that the data is not malformed—especially if it is displayed in the context of a scripting language (for example, from JScript from within a Web page).

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows 2000 SP3

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 1.3: Windows Server Update Services 2.0 (WSUS) is a component of the Windows Server 2003 and Windows 2000 Server operating systems that may be installed to enable a server to operate as an update server.

The Windows Update Agent (WUA) is a component of the Windows Server 2003, Windows XP, Windows 2000 SP4, and Windows 2000 SP3 operating systems that allows these operating systems to act as a client of the update server. The WUA will continue to be a component of the Windows Server 2008 and Windows Vista operating systems.

<2> Section 1.5: Windows Update Agent (WUA) obtains its configuration using Group Policy, as specified in [MS-GPOL]. More information is as specified in [AUPOLICY].

<3> Section 2.1: The Windows Update Services: Client-Server Protocol supports HTTPS for securing its ports, although SSL is not configured by default when the Windows Update Services: Server-Server Protocol (as specified in [MS-WSUSSS]) is installed.

<4> Section 2.1: The Windows Update Services: Client-Server Protocol does not require HTTPS.

<5> Section 2.1: WUA obtains its configuration by using Group Policy, as specified in [MS-GPOL]. For more information, see [AUPOLICY].

<6> Section 2.1: The Windows Update Services: Client-Server Protocol exposes the self-update tree on port 80. It has a setup-time configuration option to expose a non-SSL contentPort on either port 80 or port 8530. For non-SSL configurations, the Windows Update Services: Client-Server Protocol uses the contentPort as the commonPort. For SSL configurations, the Windows Update Services: Client-Server Protocol uses port 443 for the commonPort if the content port is 80, or port N+1 for the commonPort if the content port is any other port. So if the contentPort is 8530, the commonPort can be either 8530 (non-SSL) or 8531 (SSL). See [MS-WSUSSS].

<7> Section 2.1: The Windows Update Services: Client-Server Protocol server exposes the self-update tree on port 80 to support the WUA on Windows 2000 SP3, which uses protocol version 0.9. The versions of WUA that come with Windows Server 2003 and Windows XP can self-update over a configurable commonPort.

<8> Section 2.1: When invoking Client and SimpleAuth Web service methods, the (previously self-updated) WUA always adds xpress to the HTTP "Accept-Encoding" request-header to request the encoding in "Win 2K3 Compression Algorithm," as specified in [MS-DRSR] (although the request will occur on all platforms). The Windows Server Update Services (WSUS) server compresses the reply in the requested encoding (although xpress compression may be disabled in IIS).

<9> Section 2.2.1.3: All Windows implementations store the client's ID, target groups the client belongs to, and the cookie expiration time. The resulting data is serialized and encrypted by using symmetric encryption so that the client cannot interpret or modify it.

<10> Section 2.2.1.4: All Windows implementations store the client's ID, target groups the client belongs to, cookie expiration time, client protocol version, last time the client synchronized

software, drivers and printer catalog, and the server's identity. The resulting data is serialized and encrypted by using symmetric encryption so that the client cannot interpret or modify it.

<11> Section 2.2.2.1: WUA generates this unique string at the time of installation as a globally unique identifier.

<12> Section 2.2.2.1: WUA obtains its configuration by using Group Policy, as specified in [MS-GPOL]. For more information, see [AUPOLICY].

<13> Section 2.2.2.2: WUA never invokes this method. The Windows Update Services: Client-Server Protocol defines this method to support health monitor checks on the server from external tools.

<14> Section 2.2.3.1: WUA currently passes "1.6" as the version string. Because WUA self-updates before invoking this method, the agent uses the 1.6 protocol on all platforms. Older versions of WUA passed "1.0" as the version string.

<15> Section 2.2.3.1: The WSUS server on all Windows versions includes all the listed ConfigurationProperties in the 3.0 protocol. It uses the value 2 for ClientReportingLevel.

<16> Section 2.2.3.2: WUA currently passes "1.6" as the version string. Because WUA self-updates before invoking this method, the agent uses the 1.6 version string on all platforms. WSUS 3.0 Service Pack 1 uses this string to implement version-specific behavior. Older versions of WSUS ignore this string.

<17> Section 2.2.3.3: WUA currently includes these elements in the 3.0 protocol. The WSUS server assumes that the name of the client's operating system is the one specified in the first column, provided that the lvalues in the other columns match. All string comparisons are case-insensitive. For example, the following are all treated the same: AMD64, amd64, Amd64. If multiple rows of the table match, the WSUS server uses the name in the row that appears first.

<18> Section 2.2.3.3: WUA currently includes the elements ClientVersionMajorNumber, ClientVersionMinorNumber, ClientVersionBuildNumber, and ClientVersionQfeNumber in the 3.0 protocol.

<19> Section 2.2.3.4: WSUS 3.0 Service Pack 1 includes this element when the Deployment is associated with a driver update and the protocolVersion reported by the client in the GetCookie call is "1.4". Older versions of WSUS do not include this in the response.

<20> Section 2.2.3.9: WUA never invokes this method. Windows Server Update Services 2.0 (WSUS) defines this method to support health monitor checks on the server from external tools.

<21> Section 2.2.4.1: WSUS supports events with value 1 (client) or 2 (server).

<22> Section 2.2.4.1: WUA sets this to one of the EventID numbers specified in the EventID table in section 2.2.4.1.

<23> Section 2.2.4.1: For the tags used by WUA, see the MiscData table in section 2.2.4.1.

<24> Section 2.2.4.2: WUA never invokes this method. Windows Server Update Services (WSUS) defines this method to support health monitor checks on the server from external tools. For more information, see [WSUS].

<25> Section 2.2.5: The WSUS server supports the logging of faults to a server log-file. The GUID returned to the client can be used to troubleshoot errors by using it as a search string in the server log file to discover events that occurred on the server when the fault was generated.

<26> Section 3.1.1: The WSUS server uses the ClientID to identify clients.

<27> Section 3.1.1.1: Windows Server Update Services 2.0 (WSUS) populates table entries when the clients call the GetAuthorizationCookie (section 2.2.2.1) method.

<28> Section 3.1.1.1: WSUS populates table entries based on administrative configuration. The entries are never populated automatically by the server.

<29> Section 3.1.1.1: WSUS populates table entries based on administrative configuration. For more information, see [WSUS].

<30> Section 3.1.5.2: WUA caches the result of this call and only calls it again if the WSUS server throws a SoapFault with one of the specified ErrorCodes.

<31> Section 3.1.5.3: WUA calls this method only if the cookie has expired or the WSUS throws a SoapFault with one of the specified ErrorCodes.

<32> Section 3.1.5.3: WSUS stores data about each computer that contacts the server, including its dnsName, clientID, and requested target group.

<33> Section 3.1.5.3: WSUS stores client target group membership in the binary portion of the cookie. When the server is configured to run in "client-specified targeting mode", the client is placed in the requested target group (if it exists on the server), or a built-in "unassigned computers" target group (otherwise). When the server is configured to run in "server-specified targeting mode", the client's requested target group is ignored.

<34> Section 3.1.5.4: Windows Update Services: Client-Server Protocol caches the cookie of this call and calls it again only if the cookie has expired, or if WSUS throws a SoapFault with one of the specified ErrorCodes.

<35> Section 3.1.5.4: WSUS places the following information in the returned cookie:

- Client identity and target group membership (as extracted from the cookie returned by the SimpleAuth Web service).
- Cookie expiration time (enabling the server to determine whether the cookie has expired).
- Server identity (enabling the server to verify that it issued the cookie).
- Client protocol version (enabling the server to implement version-specific behavior).
- Configuration "lastChange" time (enabling the server to determine if it should force the client to call GetConfig (section 3.1.5.2) again).
- The client's last synchronization time.

The resulting data is encrypted using symmetric encryption so that the client cannot interpret or modify it.

<36> Section 3.1.5.4: WSUS copies state information from the oldCookie to the newCookie, provided the old cookie was obtained from this server.

<37> Section 3.1.5.4: WSUS stores the state information described above in the binary portion of the cookie. It defaults to a one-hour cookie expiration time.

<38> Section 3.1.5.5: WSUS requires registration information from each client. The registration information is informational only; it is stored on the server so the administrator can view the type of

client computers being managed, but it has no effect on the rest of the client-server communications protocol.

<39> Section 3.1.5.5: WSUS stores computer registration information on the server so that administrators can see the types of computers that are getting updates from the server.

<40> Section 3.1.5.6: WSUS performs this validation.

<41> Section 3.1.5.6: WSUS 3.0 Service Pack 1 includes the HardwareIDs when the Deployment is associated with a driver update and the protocolVersion reported by the client in the GetCookie call is "1.6".

<42> Section 3.1.5.6: WSUS truncates responses at 200 NewUpdates.

<43> Section 3.1.5.6: WSUS updates the cookie with the highest LastChangeTime stored in the deployment table. This makes it easier for the server to determine, on future calls to SyncUpdates, if a revision that stays in scope for the client needs to have its deployment returned in the ChangedUpdates list. (It does so only if the current deployments LastChangeTime is greater than the value stored in the cookie.)

<44> Section 3.1.5.8: WSUS stores all content in a relative path on the content directory based on the SHA1 hash of the content. The SHA1 hash of the content can be parsed from the metadata as specified in section 3.1.1.

<45> Section 3.1.5.9: WUA invokes this method only in response to a FileLocationsChanged ErrorCode.

<46> Section 3.1.5.9: WSUS always stores files in a subdirectory of the content directory that is defined by the SHA1 hash of the content. In particular, the SHA1 hash is converted to a Binhex string; the last two bytes of that string are used to specify the subdirectory; and the full string is used as the location of the file in the subdirectory.

<47> Section 3.2.1: WUA uses Group Policy. For more information about configuring additional client behavior, see [AUPOLICY].

<48> Section 3.2.2: The Windows Update Agent (WUA) obtains its configuration by using Group Policy, as specified in [MS-GPOL]. For more information, see [AUPOLICY].

- Sync Timer: Specifies the rate at which the client initiates metadata synchronization from the server. The default rate is every 22 hours (plus or minus a 20-percent randomization).
- Scheduled Install: Specifies if the client should initiate an install of downloaded software on a specified schedule. This option is enabled in the default configuration of the client on Windows XP SP2, but is disabled by default on earlier versions of Windows.

<49> Section 3.2.3: WUA uses a self-generated GUID as the client ID. It stores the GUID in the registry in HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate/SusClientId. It regenerates the GUID whenever the client joins an Active Directory domain or changes its domain membership.

<50> Section 3.2.4: The WUA uses a random delay of between 1 and 10 minutes before reporting an event batch.

<51> Section 3.2.4: WUA obtains its configuration by using Group Policy, as specified in [MS-GPOL], to drive the generation of the following higher-layer triggered events. For more information, see [AUPOLICY].

- Install at Shutdown. Specifies that all downloaded updates should be installed when the computer is shut down.
- Automatic Updates. Specifies whether the client should automatically download needed updates immediately after metadata sync, and (if so) if the client should perform a scheduled install of the downloaded updates at a particular time of day or week.
- Allow Immediate Installation. Specifies that updates whose metadata says they can be installed without interrupting any application or service should be applied immediately after the download completes.
- Delayed Restart. Specifies the amount of time for Automatic Updates to wait before proceeding with a scheduled restart (meaning a reboot triggered by the installation of an update that requires a reboot on a scheduled install).
- No auto restart. Specifies that to complete a scheduled installation, automatic update will wait for the computer to be restarted by any user who is logged on, instead of causing the computer to restart automatically.

<52> Section 3.2.5: WUA performs all optimizations identified above.

<53> Section 3.2.6: WUA implements the following timer events.

- Reporting. When a reporting event is generated on the client, it is not sent to the server right away. Rather, it is added to an event queue and a timer is started. In a random time interval from 0-5 minutes, the client uploads all events in the queue in a batch. This allows the client to send events up in batches, which optimizes network bandwidth compared to sending all the events individually.
- Retry failed sync. If metadata synchronization fails for some reason, the client will retry in five hours.

<54> Section 5: WUA only accepts install binaries signed by Microsoft certificates and validates the content's SHA1 hash with what is specified in the update metadata.

7 Index