[MS-WSUSSS]:
Windows Update Services:
Server-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	1.3	Minor	Added informative reference; minor language changes.
08/17/2007	1.3.1	Editorial	Revised and edited the technical content.
09/21/2007	1.4	Minor	Updated the technical content.
10/26/2007	1.4.1	Editorial	Revised and edited the technical content.
01/25/2008	1.4.2	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

This document specifies the Windows Update Services: Server-Server Protocol that enables a hierarchically organized collection of servers to synchronize metadata and content associated with software updates over the Internet by using the Simple Object Access Protocol (SOAP) and HTTP protocols. This protocol is implemented by Windows Server 2003 and Windows Server 2008.

1.1 Glossary

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

Globally Unique Identifier (GUID)

The following terms are specific to this document:

Anchor: An opaque data element generated by an **update server** to identify the occurrence of a software **update**-related event in a manner that distinguishes temporally separate occurrences of the event.

Autonomous DSS: A **downstream server (DSS)** that obtains **updates** from its **upstream server (USS)**, but manages the **deployments** of the **updates** to its **client computers** independently from its **upstream server (USS)**.

Category: A logical grouping of **updates** identified by a **globally unique identifier (GUID)** and described by **metadata**. A category can be treated as an **update** with no associated **content**.

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

Content: A package consisting of all files associated with an **update** for installation on a **client computer**.

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

Detectoid: A logical condition that is evaluated on a **client computer** to detect the presence of software, drivers, or their **updates**. A **detectoid** is identified by a **globally unique identifier (GUID)** and described by **metadata**. It is represented as an **update** with no associated **content**.

Downstream Server (DSS): An **update server** that synchronizes its **updates** from another **update server**.

DSS Authorization Web Service: A **Web service** on the **upstream server (USS)** used to authorize the release of **updates** to **downstream servers (DSS)**.

End User License Agreement (EULA): A textual description of the terms that a user or administrator must accept before an **update** is installed. Each **EULA** is identified by a **GUID**, and each update **revision** may be associated with a **EULA**.

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

Microsoft Update: A Microsoft-hosted Web site located at http://update.microsoft.com.

Patch Storage Format (PSF): A version of a **content** file that includes only changes in binary content from a previous version of a software/driver binary.

Replica DSS: A **downstream servers (DSS)** that obtains both **updates** and update **deployments** from its **upstream server (USS)**.

Revision: A specific version of an **update** identified by a combination of an Update ID and a 32-bit revision number.

Server Sync Web Service: A **Web service** on the **upstream server (USS)** that provides updates **metadata** and **deployments** information to the **downstream servers (DSS)**.

Simple Object Access Protocol (SOAP): An XML-based protocol for exchanging information in distributed systems, as specified in [SOAP1.1], [SOAP1.2/1], and [SOAP1.2/2].

SOAP Fault: A **Simple Object Access Protocol (SOAP)** XML element that contains fault information generated by a **SOAP** node.

SOAP Message: A discrete element of communication composed of XML elements exchanged between **Simple Object Access Protocol (SOAP)** nodes.

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

Update: The combination of **metadata** and its associated **content**. An **update** is identified by a **globally unique identifier (GUID)**.

Update Classification: A scheme to classify **updates** such as Critical, Security, Service Pack, and so on. An **update classification** is identified by a **globally unique identifier (GUID)** and described by **metadata**. It can be treated as an **update** with no associated **content**.

Update Server: A machine that implements the Windows Update Services: Server-Server Protocol for providing **updates** to **client computers** and other **update servers**.

Upstream Server (USS): An **update server** that provides **updates** to other **update servers**.

Web Method: A discrete operation exposed by a **Web service**, invoked using a single **SOAP message**.

Web Service: A software entity that responds to **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 later releases that may be installed to enable a machine to operate as an **update server**.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described 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-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

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

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

[RFC1035] Mockapetris, R., "Domain Names - Implementation and Specification", RFC 1035, November 1987, http://www.ietf.org/rfc/rfc1035.txt

[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

[XML Namespaces] Bray, T., Hollander, D., and Layman, A., "Namespaces in XML", W3C Recommendation, January 1999, http://www.w3.org/TR/1999/REC-xml-names-19990114/

[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

[MSDN-BITS] Microsoft Corporation, "Background Intelligent Transfer Service", http://msdn2.microsoft.com/en-us/library/aa362827.aspx

[MSDN-CAB] Microsoft Corporation, "Microsoft Cabinet SDK", March 1997, http://msdn2.microsoft.com/en-us/library/ms974336.aspx

1.3 Protocol Overview (Synopsis)

The **Windows Server Update Services (WSUS)** family of protocols provides support for central publication and distribution of software components and 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: Server-Server Protocol, which enables synchronization of updates within a hierarchical topology of **update servers**.

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

The following figure shows a typical hierarchical topology of update servers and **client computers**. An **upstream server (USS)** in a hierarchy provides information on software and drivers to **downstream servers (DSSs)**. Any update server in the hierarchy can serve simultaneously as a **DSS** with respect to its upstream server and as a **USS** with respect to its downstream servers. For example, in the following figure, update server C acts as a **DSS** when communicating with its upstream server A and acts as a **USS** when communicating with its downstream server D or E.

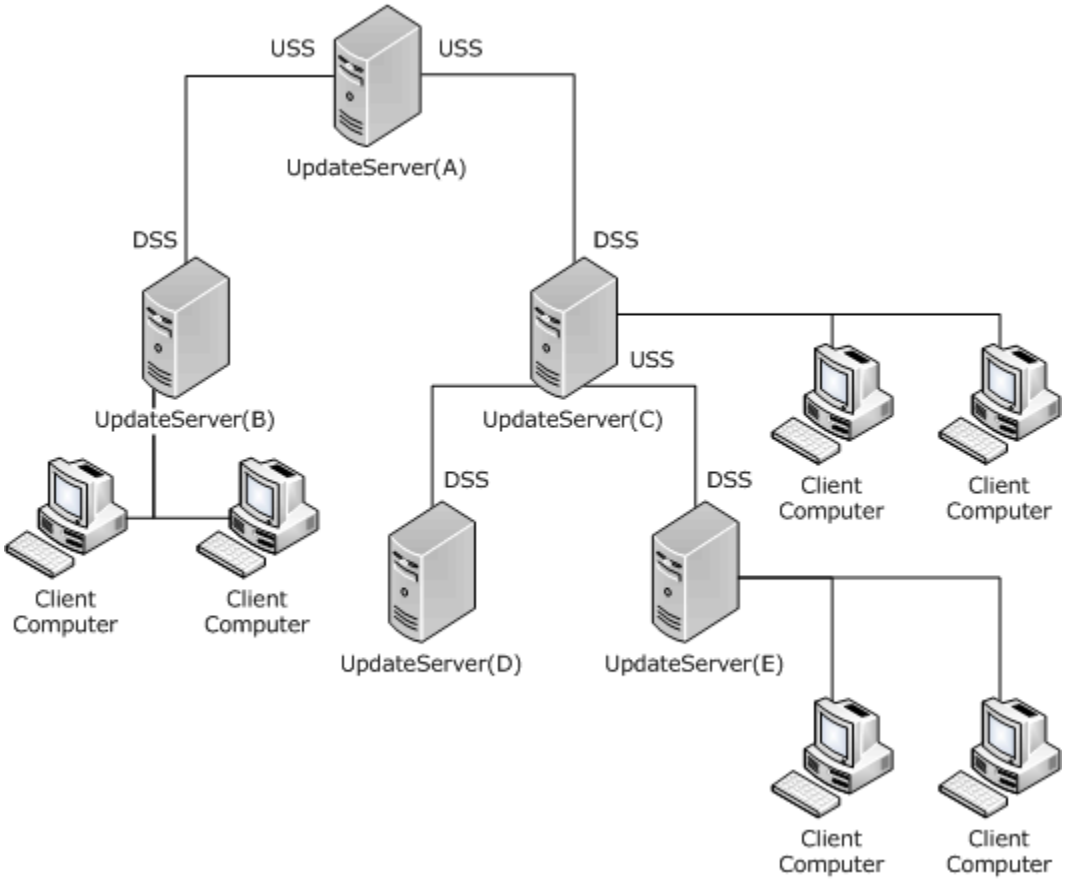

Figure 1: Typical hierarchical topology of update servers and client computers

An update server groups its client computers into **target groups**. An update server can be configured to deploy the updates to its client computers by assigning the updates to the target groups for **deployment** and, optionally, by specifying an installation or removal deadline. This mapping of the individual update revisions to target groups is known as a **deployment**.

There are two types of **DSS** defined by this protocol: autonomous and replica. Either type of **DSS** gets its updates from the **USS**. However, only a **replica DSS** gets the deployments from its **USS**. The **autonomous DSS** manages its deployments independently from the **USS**.

This specification describes the communication between two adjacent update servers in such a hierarchy.

This specification does not cover the format of self-update binaries, update metadata, and update content.

<1>

1.4 Relationship to Other Protocols

The Web services application protocol uses SOAP over HTTP or HTTPS, as specified in [SOAP1.1], for communication. The content files are downloaded using HTTP 1.1, as specified in [RFC2616].

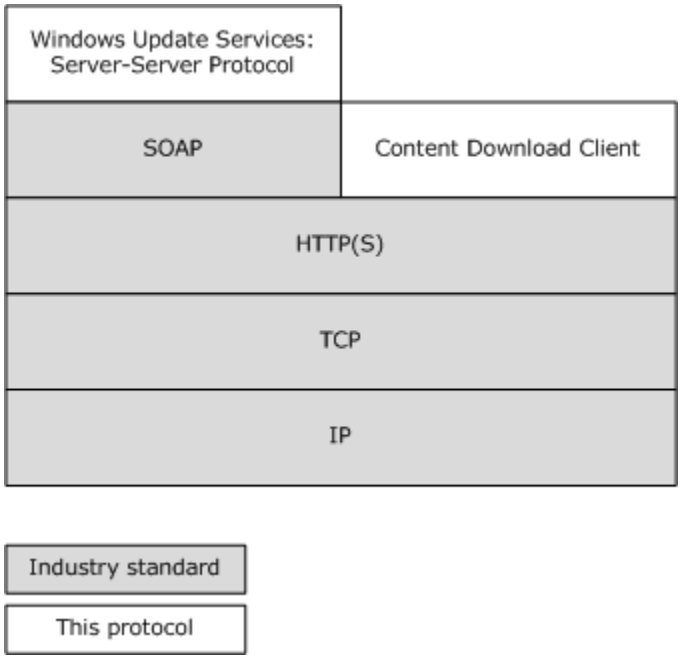

Figure 2: Windows Update Services: Server-Server Protocol relationship to other protocols

Content download is accomplished using HTTP 1.1 GET Byte Range requests, as specified in [RFC2616] section 14.35.

This Windows Update Services Server-Server Protocol is closely related to the Windows Update Services: Client-Server Protocol, as specified in [MS-WUSP], which is used for communication between update servers and client computers.

1.5 Prerequisites/Preconditions

The Windows Update Services: Server-Server Protocol imposes the following requirement on update server implementations.

- This document specifies how the binaries and metadata are distributed using the server-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 compatible with the XPATH queries, as specified in section 3.1.1.1, for populating the update server data model. In all other respects, the binaries and metadata are treated as opaque by update servers.

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

- A DSS MUST have some implementation-specific way of learning the DNS name or IP address and the TCP/IP port of the USS.
- This protocol does not mandate the use of HTTPS. However, when a USS is configured to require HTTPS, the DSS MUST have some implementation-specific way of learning the root X509 certificate that must be used for verifying the server.
- The protocol does not require the authentication of a DSS. However, when a USS is configured to require authentication, the DSS MUST have some implementation-specific way of learning the authentication scheme to be used and the identity information that needs to be passed to USS for authenticating the DSS.

1.6 Applicability Statement

The Windows Update Services: Server-Server Protocol is applicable in environments in which there is a need to synchronize software updates within a hierarchical topology of update servers.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas.

Supported Transports: The Windows Update Services: Server-Server Protocol is implemented on top of HTTP and HTTPS, as specified in section 2.1. The protocol supports the use of either SOAP 1.2 ([SOAP1.2/1] and [SOAP1.2/2]) or SOAP 1.1 ([SOAP1.1]). This configuration option MUST match on both ends of the communication, and is not negotiated within the protocol.

Protocol Versions: There are three versions of this protocol, as shown in the following table.

Protocol version	WSUS version implementing this protocol version	Number of update languages supported (see section 2.2.4.4)	Perform reporting data synchronization (see section 3.2.5)
1.1	WSUS 2.0	63	No
1.2	WSUS 2.0 SP1	511	No
1.3	WSUS 3.0	511	Yes

1.8 Vendor-Extensible Fields

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

1.9 Standards Assignments

The following standard XML namespaces (as specified in [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/

In addition, Microsoft defines the following new XML namespaces for use within this protocol.

- http://microsoft.com/wsdl/types/
- http://www.microsoft.com/SoftwareDistribution/
- http://www.microsoft.com/SoftwareDistribution/Server/DssAuthWebService

2 Messages

The Windows Update Services: Server-Server Protocol consists of a set of SOAP-based Web services and a content download service that enables content to be downloaded from the USS using HTTP, as specified in [RFC2616] section 5.

2.1 Transport

The Windows Updated Services: Server-Server Protocol operates over the following transports.

- Web Services: SOAP 1.1 ([SOAP1.1]) or SOAP 1.2 ([SOAP1.2/1] and [SOAP1.2/2]) over HTTP or HTTPS over TCP/IP ([RFC2616]).
- USS Content Download: HTTP over TCP/IP ([RFC2616]).

The Web services MUST operate on the following URI endpoints.

Web service	Location
Server Sync Web Service	http://<server>:<server port>/ServerSyncWebService/ServerSyncWebService.asmx
DSS Authorization Web Service	http://<server>:<server port>/ dssauthWebService/dssauthWebService.asmx
Reporting Web Service	http://<server>:<server port>/ReportingWebService/ReportingWebService.asmx

Each Web service MUST be bound to the HTTP transport, as specified in the following Web Service Definition Language (WSDL) element.

```
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
```

Each Web service MUST support SOAP over HTTP, as specified in [RFC2616]. Each Web service SHOULD support HTTPS for securing its communication with clients.

<2>

To optimize network bandwidth, the DSS 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 USS SHOULD compress the reply in the requested format.

<3>

USS Content Download

The USS Content Download MUST operate on the following URI endpoints.

http://<server>:<server port>/Content/<folder name>/<content file name>

where:

- <server> is the IP address or DNS name of the USS.
- <server port> is the TCP/IP port on which the USS supports this protocol.

- <folder name> is the last two hexadecimal digits of the SHA-1 hash for the file to be downloaded. The SHA-1 hash is sent down in the **GetUpdateData (section 3.1.4.6)** call.
- <content file name> is the name of the content file.

The following rules MUST be applied to the configuration of the USS Content Download transport based on the configuration of the USS Web Services transport:

- If USS Web Services uses HTTP, USS Content Download uses HTTP over TCP port 80.
- If USS Web Services uses HTTPS over TCP port 443, USS Content Download uses HTTP over port 80.
- If USS Web Services uses HTTPS over TCP port N, where N!=443, USS Content Download uses HTTP over port –1.

<4>

A USS Content Download SHOULD support HTTP 1.1 Byte Range requests, as specified in [RFC2616] section 14.35. The downloaded content files are binary.

<5>

2.2 Message Syntax

This section specifies the syntax of SOAP messages exchanged by the protocol. The following rules apply to all SOAP messages in the protocol.

- The <soap:header> element ([SOAP1.1] section 4.2 and [SOAP1.2/1] section 5.2) MUST NOT be used.
- The <soap:binding> element of the 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 this protocol. In the WSDL file, the **minOccurs** and **maxOccurs** attributes are used to specify options and cardinality of all elements, except where otherwise specified.

For certain 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 described immediately after the WSDL is given.

2.2.1 Common Data Types

The following table shows the standard XML namespaces (as specified in [XML Namespaces]) used within this 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/
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 this 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 DSS Authorization Web Service WSDL, it is http://www.microsoft.com/SoftwareDistribution/Server/DssAuthWebService. For Server Sync Web Service WSDL, and Reporting Web Service WSDL, it is http://www.microsoft.com/SoftwareDistribution.

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

2.2.1.1 GUID

A **globally unique identifier (GUID)** of an object or entity within the protocol. For example, each update has a unique ID that is a **GUID**.

Defined in the namespace: http://www.microsoft.com/wsdl/types.

```
<s:schema elementFormDefault="qualified"

targetNamespace=http://microsoft.com/wsdl/types/">
 <s:simpleType name="guid">
  <s:restriction base="s:string">
   <s:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-
9a-fA-F]{4}-[0-9a-fA-F]{12}" />
  </s:restriction>

  </s:simpleType>
</s:schema>
```

2.2.1.2 ArrayOfInt

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

Defined in the 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>
```

2.2.1.3 AuthorizationCookie

An object returned by the USS on successful completion of the **GetAuthorizationCookie** operation.

Defined in the 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 bytes constituting the data for the authorization cookie. The USS creates the data and interprets its content. The DSS does not interpret the content of this element. The format of the data is implementation dependent; however, the USS that generated this data MUST be able to obtain the following information from the data.

- Globally unique identifier identifying the DSS.
- List of target groups to which the DSS belongs.
- Cookie expiration time.

<6>

2.2.1.4 ArrayOfAuthorizationCookie

An array of **AuthorizationCookie** objects.

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

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

2.2.1.5 Cookie

An object generated by the USS to opaquely capture USS protocol state for submission by the DSS in subsequent protocol operations.

Defined in the 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: UTC date/time when the cookie expires.

EncryptedData: An opaque sequence of bytes constituting the data for the cookie. The USS creates the data and interprets its content. The DSS does not interpret the content of this element. The format of the data is implementation dependent; however, the USS that generated this data MUST be able to obtain the following information from the data.

- Globally unique identifier identifying the DSS.
- List of target groups to which the DSS belongs.
- Cookie expiration time.
- Protocol version.
- Globally unique identifier identifying the USS.

<7>

2.2.1.6 ArrayOfUpdateIdentity

An array of **UpdateIdentity** objects used in messages within the protocol.

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

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

2.2.1.7 UpdateIdentity

A GUID for a specific revision of an update.

Defined in the 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 uniquely identifies an update.

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

2.2.1.8 ArrayOfBase64Binary

An array of binary values encoded in Base 64.

Defined in the 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>
```

2.2.1.9 ArrayOfGuid

An array of **GUIDs** used in messages within this protocol.

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

```
<s:complexType name="ArrayOfGuid">
  <s:sequence>
   <s:element minOccurs="0" maxOccurs="unbounded" name="guid"
   type="s2:guid" />
  </s:sequence>
</s:complexType>
```

2.2.1.10 ArrayOfString

An array of strings used in messages within this protocol.

Defined in the 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>
```

2.2.2 SOAP Faults

This protocol allows a USS to notify a DSS of application-level faults by generating SOAP faults. Not all SOAP faults returned by the USS contain application-level fault information. Details are specified in sections 3.1.4 and 3.2.4.

In the case that a SOAP fault contains application-level fault information, the format is specified in the following sections. The format varies depending on the SOAP version that is being used.

2.2.2.1 Format for SOAP 1.1

When using SOAP 1.1, the "detail" sub-element under the "fault" element MUST contain the following child elements.

- <ErrorCode>: A case-sensitive string that identifies the type of exception being thrown. The valid values and their descriptions are given in the table in section 2.2.2.3.
- <Message>: A human-readable message describing the error.
- <ID>: A GUID that identifies this instance of the fault.

These element names MUST NOT be qualified by a namespace.

The other sub-elements of the "fault" element MUST be set as specified in [SOAP1.1].

2.2.2.2 Format for SOAP 1.2

When using SOAP 1.2, the Fault element MUST contain a child element named "detail", with no namespace name. This "detail" element MUST contain <ErrorCode>, <Message>, and <ID> sub-elements, as specified in section 2.2.2.1. This "detail" element is distinct from the Detail element specified in [SOAP1.2/1].

The other sub-elements of the Fault element MUST be set as specified in [SOAP1.2/1].

2.2.2.3 ErrorCode Values

When present, the <ErrorCode> element MUST contain one of the following values. If a DSS receives a SOAP fault with one of the following ErrorCode values, it MUST react to the fault as defined in the following table.

ErrorCode	Description
InvalidParameters	Parameters passed to a Web method are not valid. The "message" part of the exception MUST contain the parameter name. The DSS MAY retry the operation with a new set of valid parameters, if available. Otherwise, the DSS MUST abort the protocol.<8>
InvalidCookie	The **EncryptedData** field of the cookie has a syntax, formatting, or other error. The DSS MUST restart the protocol from the beginning.
InternalServerError	An internal error occurred on the server. The DSS MUST abort the protocol.
IncompatibleProtocolVersion	The version of the protocol used by DSS is incompatible with the version used by USS. The DSS MUST abort the protocol.
InvalidAuthorizationCookie	The authorization cookie submitted by the DSS is invalid. The DSS MUST restart the protocol from the beginning.

ErrorCode	Description
FileDigestsMissing	Some or all of the requested **FileDigest** values are not known to the USS. This may be due to deletion of some unneeded updates on the USS. The DSS MUST restart the protocol from the beginning.
ServerChanged	The USS has changed. The DSS MUST restart the protocol from the beginning.
ServerBusy	The USS is too busy to handle this request. The DSS MAY back off and try again later. Otherwise, the DSS MUST abort the protocol.<9>

3 Protocol Details

This protocol operates between two update servers acting in the following roles.

- DSS: This update server initiates all communication with the USS.
- USS: This update server responds to requests received from the DSS.

The protocol incorporates support for the following capabilities.

- Discovering and synchronizing software and driver updates metadata.
- Downloading content for the updates synchronized by the DSS from the USS.
- Discovering and synchronizing target groups when the DSS is configured as a replica of the USS.
- Discovering and synchronizing deployments of the updates to target groups when the DSS is configured as a replica of the USS.
- Reporting the status of updates on client computers by the DSS to the USS.

The protocol is intended to be stateless wherever possible. However, due to the requirements of authorization and the inter-relationships between deployments, updates, and target groups, the DSS must execute the steps of this protocol in a specific order. A high-level sequence figure, as specified in section 3.2.4, summarizes the working of the protocol. Further details are provided in the following sections. This section first specifies details that are common between both roles. It then provides additional USS-specific and DSS-specific behavior, respectively.

Information on updates, target groups, and deployments is maintained on each USS, and each DSS obtains this information through a series of SOAP message exchanges that comprise the USS Web Services.

Similarly, information on descendent DSSs, client computers, and the status of updates on those client computers is maintained on each DSS, and the DSS sends this information to its USS through a series of SOAP message exchanges with USS.

A USS may also be configured to store the content associated with updates. In such a configuration, each DSS obtains content from its USS through a series of HTTP operations that comprise the USS Content Download.

If the content is not available from the USS, the DSS obtains content from an Internet Web site through a series of HTTP operations using the file location URLs obtained from the USS.

3.1 USS Details

An update server in the USS role provides a downstream update server (DSS) with the capability to synchronize update metadata, update content, target groups, and deployments. The USS provides Web services for DSS to communicate using SOAP message exchange, it may also be configured to provide update content via HTTP download requests. This section provides the processing details of the USS role.

3.1.1 Abstract Data Model

This section describes a conceptual model of 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 their external behavior is consistent with that described in this document.

Both the DSS and the USS maintain the following information regarding updates, target computers, target groups, and deployments.

Server Configuration: A set of configuration elements specified administratively for the server.

- **ServerID**: A self-generated GUID that identifies the update server.
- Replication Mode: Autonomous or replica, as specified in section 1.1.
- CatalogOnlySync?: A flag that indicates if the server stores the content files locally and exposes them as the USS Content Download service. When set to FALSE, the server stores the content files locally and supports the USS Content Download service. When set to TRUE, it specifies that the USS only supports the USS Web Services and does not support the USS Content Download service.
- LazySync?: A flag that indicates when the content files are downloaded. This flag is ignored if **CatalogOnlySync** is set to TRUE. It can be one of the following:
  - TRUE: Content files are downloaded when the update is approved for install.
  - FALSE: Content files are downloaded when the update metadata is synchronized from a parent USS.
- ServerHostsPsfFiles?: A flag that indicates if the server will store the PSF versions of content files if available for an update.
- **MaxNumberOfUpdatesPerRequest**: The maximum number of revision IDs that the DSS is allowed to submit in the **GetUpdateData (section 3.1.4.6)** operation.
- Languages Support: The list of languages known to the update server and a flag for each language indicating whether or not the update server supports the language.
  - All Languages: A flag indicating that the update server supports all languages. When All Languages is set to TRUE, the specific language settings MUST be ignored. If set to FALSE, the specific language settings define what languages are supported by the update server.
  - Specific Language Settings: Each supported language MUST include the following elements:
    - Language ID
    - Short name
    - Long name
    - Enabled state (TRUE/FALSE)
- **DoDetailedRollup**: A flag that indicates if this update server allows DSSs to report detailed information about client computers that get updates from them.
  - TRUE: This update server allows DSSs to report detailed information about client computers that get updates from them.
  - FALSE: This update server does not allow DSSs to report detailed information about client computers that get updates from them. The update server MUST reject any calls to the **RollupComputers**, **GetOutOfSyncComputers**, and **RollupComputerStatus** methods.

Server State: A set of information on this server's runtime state.

- **ConfigAnchor**: An anchor that identifies the last change in the configuration for the server.

Synchronization History Table: A table that tracks the completion time stamp of each successful synchronization with the USS. If this update server is configured as a replica server, a new entry MUST be added following the successful completion of the Deployments Synchronization (section 3.2.4.3) step. If this update server is configured as an Autonomous Server, a new entry MUST be added following the successful completion of the Metadata Synchronization (section 3.2.4.2) step. Existing entries in the table MAY be removed at any time. However, entries SHOULD NOT be removed until they are at least 15 days old.

<10>

- SynchronizationTime: A time stamp that identifies the time when this update server successfully completed the Deployments Synchronization (section 3.2.4.3) step if this server is a replica server, or completed the Metadata Synchronization (section 3.2.4.2) step if this server is an Autonomous Server.

Parent USS Configuration: Each update server has at most one parent USS. The following information on the parent USS is configured by an administrator and maintained by the update server.

- Server DNS name or IP address where the USS Web Services is provided.
- Server port on which the USS Web Services is provided.

Parent USS State: A set of information regarding the state of this server's communication with its USS. This information is received from the USS and is maintained for use within the protocol.

- Last AuthorizationCookie received from the parent USS in a **GetAuthorizationCookie (section 3.1.4.2)** response.
- Last Cookie received from the parent USS in a **GetCookie (section 3.1.4.3)** response.
- Last **ConfigAnchor** received from the parent in a **GetConfigData (section 3.1.4.4)** response.
- Last Sync Anchor received from the parent USS in a **GetRevisionIdList (section 3.1.4.5)** response.
- Last Deployment Anchor received from the USS in a **GetDeployments (section 3.1.4.8)** response.
- **MaxNumberOfUpdatesPerRequest** received from the parent USS in a **GetConfigData** (section 3.1.4.4) response.
- List of languages supported by the USS. This is received from the parent USS in a **GetConfigData** (section 3.1.4.4) response.
- CatalogOnlySync?: A flag that indicates if the USS stores the content files locally. This is received from the parent USS in a **GetConfigData** (section 3.1.4.4).
- LazySync?: A flag that indicates when the content files are downloaded. This is received from the parent USS in a **GetConfigData** (section 3.1.4.4). This flag is ignored if **CatalogOnlySync** is set to TRUE. It can be one of the following:
  - TRUE: Download content when the update is approved for install.

- FALSE: Download content immediately.

- ServerHostsPsfFiles?: Set to TRUE if the server stores the PSF versions of content files when available; otherwise, set to FALSE. This is received from the parent USS in a **GetConfigData** (section 3.1.4.4).

Target Groups Table: A table of entries that correspond to each target group that is configured on the update server. The entries in this table are administratively configured if the update server is an autonomous DSS of its parent USS. Otherwise, the entries in this table are received from the parent USS in response to invocations of the **GetDeployments** (section 3.1.4.8) method. Each entry is indexed by its **TargetGroupID** and MUST include the following elements.

- TargetGroupID: A GUID that identifies this target group.

- Name: The name of the target group given as a string value.

- IsBuiltin?: A flag indicating whether this target group is provided by the implementation of the server or is created by a user.

DSS Table: A table that stores the information on each DSS and descendant DSSs that synchronize with this update of this update server.

A new entry is added to this table when a **GetAuthorizationCookie** message is received for the first time from a DSS. New entries are also added to this table when the USS receives a **RollupDownstreamServers** message that contains information on new DSSs. Existing entries MAY be modified or removed at any time. When an entry is deleted from this table, all entries in the client computers table with a **ParentServerID** value matching the **ServerID** value of the deleted entry MUST be deleted.

<11>

Each entry is keyed by the **ServerID** and MUST include the following elements.

- **ServerID**: The GUID that identifies this DSS.

- Fully qualified DNS name of the DSS.

- **ParentServerID**: The **ServerID** of the USS that this DSS synchronizes with.

- **LastRollupTime**: A time stamp that indicates the time when this DSS last reported information on itself to its USS by calling the **RollupDownstreamServers** method. The value MUST be initialized to NULL.

- **LastSyncTime**: A time stamp that identifies when this DSS last synchronized with its parent USS. The value MUST be initialized to NULL.

- **Version**: The version number of the software on the DSS that gives it the capability to synchronize with the USS.

  <12>

- IsReplica?: A flag indicating whether this DSS is configured to be an Autonomous DSS or a Replica DSS of its parent USS.

- **UpdateCount**: The number of updates known to the DSS.

- **DeclinedUpdateCount**: The number updates that have been marked as hidden on the DSS.

- **ApprovedUpdateCount**: The number of updates that have at least one Deployment with an Action of Install or Uninstall on the DSS.
- **NotApprovedUpdateCount**: The number of updates that are not marked as hidden and have no Deployments with an Action of Install or Uninstall on the DSS.
- **UpdatesWithStaleUpdateApprovalsCount**: The number of updates that have at least one Deployment with an Action of Install or Uninstall on the DSS where the deployment is associated with a revision of the update other than the latest revision.
- **CriticalOrSecurityUpdatesNotApprovedForInstallCount**: The number of updates that are related to the security of the client computers or are otherwise considered critical that have no Deployments with an Action of Install on the DSS.
- **WsusInfrastructureUpdatesNotApprovedForInstallCount**: The number of updates that the client computers must install to enable them to continue to get updates from the DSS, which have no Deployments with an Action of Install on the DSS.
- **UpdatesWithClientErrorsCount**: The number of updates on the DSS that at least one client computer has attempted and failed to install.
- **UpdatesWithServerErrorsCount**: The number of updates on the DSS for which the DSS has attempted to download content but was unable to complete the download due to an error.
- **UpdatesNeedingFilesCount**: The number of updates on the DSS for which the DSS must download content but has not completed the download.
- **UpdatesNeededByComputersCount**: The number of updates on the DSS that have at least one client computer that it is applicable to but not yet installed on.
- **UpdatesUpToDateCount**: The number updates on the DSS that are known to be installed on all client computers that it is applicable to.
- **CustomComputerTargetGroupCount**: The number of target groups on the DSS that have been created administratively on the DSS or have been received from its USS.
- **ComputerTargetCount**: The number of client computers that get updates from this DSS.
- **ComputerTargetsNeedingUpdatesCount**: The number of client computers that get updates from this DSS, on which at least one update is known to be applicable, but not yet installed.
- **ComputerTargetsWithUpdateErrorsCount**: The number of client computers that get updates from this DSS that has tried and failed to install at least one update.
- **ComputersUpToDateCount**: The number of client computers that get updates from this DSS that are known to have successfully installed all updates that are applicable to it.

Client Computers Table: A table that stores the information on each client computer that gets updates from this update server or from a descendant DSS. New entries are added to this table on the USS when a DSS reports information on new client computers to the USS using the **RollupComputers** method. New entries MAY be added to this table when new client computers are discovered through other implementation-specific means. Existing entries MAY be modified or removed at any time.

<13>

Each entry is indexed by the ComputerID and MUST include the following elements.

- **ComputerID**: A globally unique string that identifies this client computer.
- **ParentServerID**: The **ServerID** of the update server that the client computer gets updates from.
- **LastSyncTime**: A timestamp that identifies when this client computer last contacted the update server to get updates. The value MUST be initialized to NULL.
- **LastSyncResult**: The result of the last attempt by this client computer to get updates from the update server. The valid values are the following:
  - Unknown/not applicable: The result is unknown, or the client computer has never attempted to get updates from the update server.
  - Succeeded: The client computer successfully retrieved updates from the update server.
  - Failed: The client computer failed to retrieve updates from the update server.
- LastSentStatusRollupNumber: The **RollupNumber** value that was used when this update server last sent information about this client computer to the USS using the **RollupComputerStatus** method.
- **LastReceivedStatusRollupNumber**: The **RollupNumber** value that was used when this update server last received information about this client computer from a DSS.
- **LastReportedRebootTime**: A timestamp that identifies when this client computer notified the update server that it has rebooted. The value MUST be initialized to NULL.
- **LastInventoryTime**: A timestamp indicating the last time this client computer reported software and hardware inventory information to the update server. The value MUST be initialized to NULL.
- **RequestedTargetGroupNames**: A list of strings listing the target groups of which the client computer has been configured to be a member.
- **IPAddress**: The IP address of the client computer.
- **FullDomainName**: The fully qualified DNS name of the client computer.
- **OSMajorVersion**, **OSMinorVersion**, **OSBuildNumber**, **OSServicePackMajorVersionNumber**, **OSServicePackMinorVersionNumber**, **OSLocale**, **SuiteMask**, **NewProductType**, **OldProductType**, and **SystemMetrics**: These values are used to identify the operating system used by the client computer.

  This protocol provides a mechanism for transporting these values between update servers, but does specify how these values are initialized or how they are consumed. An implementation that consumes these values MUST have an out of band mechanism for determining the meaning of these values.

  <14>
- Computer manufacturer and model names.
- BIOS name, version, and release date.
- Version number of the client software that gives the client computer the ability to get updates from the update server.

- **TargetGroupIDList**: The list of GUIDs identifying the target groups that this client computer belongs to. Each GUID in the list corresponds to the **TargetGroupID** value from an entry in the Target Groups Table.
- HasDetailsChanged?: A flag that indicates if the value of the **RequestedTargetGroupNames**, **IPAddress**, **FullDomainName**, OS version number, Computer manufacturer/model, BIOS name/version/release date, or the **TargetGroupID** element has changed since the last time information about this client computer was sent to the USS in a **RollupComputers** request. This value MUST be set to TRUE when the value of one or more of these other elements is changed.
- LastStatusRollupTime: A timestamp indicating when the status of updates on this client computer was last reported to the USS. The value MUST be initialized to NULL.
- EffectiveLastDetectionTime: A timestamp indicating the time when the newest update that the client computer has reported status for was made available on the update server. This value MUST be initialized to NULL.

  This value is updated on the USS as part of the Reporting Data Synchronization step. If the implementation has the ability to receive messages from client computers describing the state of all updates on the client computer, then this field SHOULD be updated at the same time these messages are processed.

  <15>

Update Status Table: A table that tracks the status of each update on each client computer. Entries are added to, and removed from, this table on the USS when a DSS reports information regarding the status of updates on its client computers using the **RollupComputerStatus** method.

Entries MAY be added or removed through additional implementation specific means.

<16>

Each entry is indexed by the pair (ComputerID, UpdateID), and MUST include the following elements.

- ComputerID: A globally unique string that identifies this client computer.
- UpdateID: A GUID that identifies the update. This corresponds to the GUID portion of the UpdateIdentifier element in the Revisions Table.
- Status: The status of the update on the client computer. The valid values are shown in the following table.

Value	Meaning
0	The status of the update on the client computer is unknown.
1	The update is not applicable to the client computer (the software that the update is intended to update is not present).
2	The update is applicable to the client computer, but it has not yet been downloaded or installed.
3	The update is applicable to the client computer, and the client computer has downloaded all files required to install the update, but it has not installed the update yet.
4	The update is applicable to the client computer, and the update has been installed.

Value	Meaning
5	The update is applicable to the client computer, but the client computer has either failed to download the files required to install the update, or has completed the download but has failed to install the update.
6	The update is applicable to the client computer, and the update has been installed, but the client computer is required to reboot before the update can take effect.

- LastChangeTime: A timestamp indicating when the status of this update last changed on this client computer.

Client Computer Activity Summary Table: A table that tracks the number of times an update has been installed, successfully and unsuccessfully, by client computers. The client computers are grouped based on the update server that they get updates from, and the operating system used by the client computer. Entries are added, modified, and removed as part of the Reporting Data Synchronization step of this protocol.

Update server implementations MUST have the capability to receive notifications from client computers on which they have installed an update (or have failed to do so). The mechanism for receiving such notifications is implementation dependent; however, when such notifications are received, this table MUST be modified at that time as follows:

1. Use the Client Computers table to determine the OS version number of the client computer and the ServerID of the update server it gets updates from.
2. Find the entry in this table with the corresponding OS version number and ServerID. If no such entry exists, insert one.
3. If the client computer has successfully installed an update, increment the InstallSuccessCount value of this entry. Otherwise, if the client computer attempted to install the update but failed to successfully complete the install, increment the InstallFailureCount value.

<17>

Each entry is indexed by the UpdateID, **ServerID**, and the OS version elements, and MUST include the following elements.

- UpdateID: A GUID that identifies the update. This corresponds to the GUID portion of the UpdateIdentifier element in the Revisions Table.
- **ServerID**: The **ServerID** of the update server that the client computers get updates from.
- OS version: Consists of the **OSMajorVersion**, **OSMinorVersion**, **OSBuildNumber**, **OSServicePackMajorVersionNumber**, **OSServicePackMinorVersionNumber**, **OSLocale**, **SuiteMask**, **NewProductType**, **OldProductType**, and **SystemMetrics** values, corresponding elements in the client computers table.
- **InstallSuccessCount**: The number of times that client computers, running the operating system described by the OS version elements, have successfully installed the update specified by the UpdateID element.
- **InstallFailureCount**: The number of times that client computers, running the operating system described by the OS version elements, have attempted unsuccessfully to install the update specified by the UpdateID element.

The **InstallSuccessCount** and **InstallFailureCount** values MUST be initialized to 0, and MUST be incremented accordingly whenever a client computer (using an operating system with version numbers matching the OS version fields) notifies the update server that it has successfully or unsuccessfully installed this update. The mechanism used by client computers to notify the update server is implementation dependent.<18>

The value is also modified, on both the USS and DSS, as part of the Reporting Data Synchronization (section 3.2.4.5) step of this protocol.

Categories Table: A table of entries corresponding to each category with which an update may be associated. Each entry is indexed by the CategoryIdentity and MUST include the following elements.

- CategoryIdentity: This consists of a GUID that identifies the category and a revision number for each version of the category.
- XMLMetaData: This provides all information about the category in XML format.
- LastChangedAnchor: An anchor that identifies when this entry was changed.

Update Classifications Table: A table of entries corresponding to each update classification that may be assigned to an update. The entries in this table are generated at runtime when this update server synchronizes its updates from its parent USS. Each entry is indexed/identified by the ClassificationIdentity, and MUST include the following elements.

- ClassificationIdentity: This consists of a GUID that identifies the update classification and a revision number for each version of the update classification.
- XMLMetaData: This provides all information about an update classification in XML format.
- LastChangedAnchor: An anchor that identifies when this entry was changed.

Detectoids Table: A table of entries corresponding to each detectoid. The entries in this table are generated at runtime when this update server synchronizes its updates from its parent USS. Each entry is indexed/identified by DetectoidIdentity, and MUST include the following elements.

- DetectoidIdentity: This consists of a GUID that identifies the detectoid and a revision number for each version of the detectoid.
- XMLMetaData: This provides all information on the detectoid in XML format.
- LastChangedAnchor: An anchor that identifies when this entry was changed.

Revisions Table: A table of entries corresponding to the revisions that exist for each update. The entries in this table are generated at runtime when this update server synchronizes its updates from its parent USS. Each entry is indexed/identified by UpdateIdentity and MUST include the following elements.

- UpdateIdentity: This consists of a GUID that identifies the update and a revision number for each revision of the update.
- XMLMetaData: This provides all information about an update in XML format.
- Hidden?: A flag indicating if the administrator has chosen to hide this update. Hidden updates are assumed to be declined and will not be offered for deployment. When configured as a replica, a DSS gets information on hidden updated from its parent USS in the response to **GetDeployments** (section 3.1.4.8).
- LastChangedAnchor: An anchor that identifies when this entry was changed.

- Classification: An index into the Update Classification Table identifying the update classification for this update revision.
- For each content file associated with this revision:
  - **FileDigest**: An SHA-1 hash of the content file.
  - **MUUrl**: A HTTP URL specifying the location of this file on the Microsoft Update HTTP service. This field is NULL for updates that do not originate from the Microsoft Update Web site.

EULAs Table: A table of entries corresponding to the EULA for the software and driver update revisions. The entries in this table are populated at run time from the XMLMetaData associated with this revision using the **EulaID** property, as specified in section 3.1.1.1. The Accepted flag of each entry may be updated by administrative action. Each entry is uniquely identified by its **EulaID** and MUST include the following elements.

- EulaID: This is a GUID that identifies the EULA.
- Accepted?: A flag indicating if this EULA was accepted by the administrator of this update server or its parent or ancestor update server.

Deployments Table: A table of entries corresponding to each deployment that is administratively approved for distribution to client computers and DSSs. The entries in this table are administratively configured if the update server is an autonomous DSS of its parent USS. Otherwise, the entries in this table are received from the parent USS using **GetDeployments** (section 3.1.4.8). Each entry is uniquely identified by DeploymentID and MUST include the following elements.

- DeploymentID: This is a GUID that identifies the deployment.
- UpdateIdentity: This consists of a GUID that identifies the update and a revision number for the update revision that is deployed by this deployment.
- TargetGroupID: An index into the target groups table identifying the group of client computers to be targeted by this deployment.
- Action: This field specifies the specific action defined by this deployment. The following are valid values for Action.

Value	Meaning
0	Install the update revision.
1	Uninstall the update revision.
2	Scan for the presence of the update revision.
3	Block.

- LastChangedAnchor: An anchor that identifies when this entry was changed.
- All other deployment options as given in **ServerSyncDeployment** structure, as specified in **GetDeployments** (section 3.1.4.8).

Content Store: A data store where content files are stored. The names of the files are as specified by the parent USS. The Content Store is empty if the **CatalogOnlySync** flag is set to FALSE for this update server. Each file is uniquely identified by the SHA-1 hash value. The Content Store stores the

file by organizing them into folders with two character names that match the last two characters of the SHA-1 hash for files stored in that folder.

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 in the following table can all be extracted using XPATH queries, as specified in [XPATH].

The following properties are extracted from the metadata using the XPATH queries shown in the following table. The other sections of this document indicate what property is being extracted from the metadata.

Property	XPATH
UpdateType	/Update/Properties/@UpdateType
CategoryType	/Update/HandlerSpecificData/CategoryInformation/@CategoryType
EulaID	/Update/Properties/@EulaID
FileDigest	/Update/Files/File[]/@Digest
PatchingType	/Update/Files/File[]/@PatchingType
FileName	/Update/Files/File[]/@FileName

3.1.2 Timers

None.

3.1.3 Initialization

The following initialization steps MUST be performed.

1. All tables in section 3.1.1 except the Last AuthorizationCookie in the USS Parent State table contain persistent data that MUST be retrieved from persistent storage at initialization time.
2. Each field of the USS Parent State Table MUST be initialized to NULL.
3. Each Web service within the server MUST begin listening for requests at the respective URL addresses specified in section 2.1.
4. If the USS is configured to store content files locally (see the CatalogOnlySync flag as specified in section 3.1.1 in "Server Configuration"), the USS Content Download service MUST be started at the URL address as specified in section 2.1.

3.1.4 Message Processing Events and Sequencing Rules

When a message arrives, the first XML element inside <soap:body> identifies the operation that is requested by the DSS. The message is then handled based on this operation name as specified in the following sections.

3.1.4.1 GetAuthConfig

A DSS calls the **GetAuthConfig** method to get information on the authorization service to use to get the authorization cookie.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/GetAuthConfig"
 style="document" />
```

Request:

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

This type has no fields.

Response:

```
<s:element name="GetAuthConfigResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
      name="GetAuthConfigResult"
      type="s1:ServerAuthConfig" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetAuthConfigResult: On successful execution of this operation, this object MUST be returned. Its format is as follows:

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

LastChange: This field is not used. The DSS MUST ignore this field.

AuthInfo: Contains the Authorization PlugIn information on the USS. On successful execution of this operation, this array MUST contain exactly one element. Its format (ArrayOfAuthPlugInInfo) is shown below.

AllowedEventIds: Unused. It SHOULD NOT be present, and MUST be ignored on receipt, if present.<19>

```
<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 on the Authorization PlugIn available on the USS. 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 set to "DssTargeting".

ServiceUrl: This MUST be set to "DssAuthWebService/DssAuthWebService.asmx". It is a partial URL that can be appended to http://<server>:<server port>/ to form the full URL to be used for DSS Authorization Web Service.

Parameter: Unused. It SHOULD NOT be present, and MUST be ignored upon receipt.<20>

Request validation:

None.

Data processing:

The USS MUST compose a GetAuthConfigResponse message in response, as follows:

- **AuthInfo**: MUST contain exactly one element, and its fields MUST be set as the following:
  - **PlugInID**: Set to "DssTargeting".
  - **ServiceUrl**: Set to "DssAuthWebService/DssAuthWebService.asmx".
  - **Parameter**: This MUST NOT be included.

Response:

If no errors occur during processing, the USS MUST return the response to the DSS.

The following SOAP fault may be returned by this operation.

ErrorCode	Description
InternalServerError	An internal error occurred on the server. The error is implementation specific. The DSS MUST abort the protocol.

3.1.4.2 GetAuthorizationCookie

A DSS calls the **GetAuthorizationCookie** method to authenticate and authorize access to the Server Sync Web Service. The USS sends back an authorization cookie that is used in the **GetCookie** operation.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistributi
on/Server/DssAuthWebService/GetAuthorizationCookie" style="document" />
```

Request:

```
<s:element name="GetAuthorizationCookie">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
      name="accountName" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1"
      name="accountGuid" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

accountName: This field MUST be present. It MUST be set to the fully-qualified DNS name of the DSS.

accountGuid: This field MUST be present. It MUST be a GUID that identifies the DSS.

Response:

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

GetAuthorizationCookieResult: On successful completion of this operation, this field MUST be returned. It MUST be an AuthorizationCookie structure, as defined in section 2.2.1.3, and the **PlugInId** field on the structure MUST be set to "DssTargeting".

Request validation:

The USS validates inputs as specified below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS with the <ErrorCode> set, as shown in the following table.

Parameter	Validation conditions	ErrorCode
accountName	MUST not be null or a zero-length string. The format of this field MUST be consistent with DNS name requirements (that is, it does not contain characters that are invalid for a DNS name, as specified in [RFC1035] section 2.3).	InvalidParameters
accountGuid	MUST not be null or a zero-length string. The format of this field MUST be consistent with the GUID format, as specified in section 2.2.1.1.	InvalidParameters

Data processing:

The USS MUST process this message as follows:

1. Locate the entry identified by the **accountGuid** element in the DSS Table.
2. If not found, initialize and insert a new entry in the DSS Table containing the **accountGuid** and **accountName** elements.
3. The USS MUST compose a GetAuthorizationCookieResponse message in response to the DSS, as follows:
   - Generate an AuthorizationCookie element for the DSS.
   - Initialize the *PlugInId* element to "DssTargeting".
   - Initialize the *CookieData* element to a sequence of bytes, as defined in section 2.2.1.3.

Response:

If no errors occur during processing, the USS MUST return the response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault. The SOAP fault SHOULD contain an <ErrorCode> element, as defined in section 2.2.2. If the SOAP fault contains an <ErrorCode> element, its value MUST be one of the following.

If the DSS receives a SOAP fault containing an <ErrorCode> element, it MUST react to the fault, as described below. If the DSS receives a fault that does not contain an <ErrorCode> element, it MUST abort the protocol.

ErrorCode	Description
InternalServerError	An internal error occurred on the server. The error is implementation specific. The DSS MUST abort the protocol.
InvalidParameters	Parameters passed to a Web method are not valid. The "message" part of the exception will contain the parameter name. The DSS MUST abort the protocol.

3.1.4.3 GetCookie

A DSS invokes the **GetCookie** method to obtain a cookie containing an opaque sequence of bytes that encode implementation-specific authorization, authentication, and runtime information for use by the USS.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/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="0" maxOccurs="1"
      name="protocolVersion" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

authCookies: This field MUST be present. It provides the list of Authorization Cookies received via the **GetAuthorizationCookie (section 3.1.4.2)** method. Because only one Authorization PlugIn is defined within this protocol, the array MUST contain exactly one element.

oldCookie: This field MUST be set to the cookie returned in the last successful **GetCookie** operation response.

protocolVersion: This field MUST be present. It identifies the version of the protocol that the DSS supports, as specified in section 1.7. Its values may be "1.1", "1.2", or "1.3".

Response:

```
<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 field MUST be returned. It MUST be a Cookie structure, as defined in section 2.2.1.5.

Request validation:

The USS validates inputs, as specified below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS with the <ErrorCode> set, as shown in the following table.

Input	Validation conditions	ErrorCode
authCookies	MUST contain exactly one AuthorizationCookie element	InvalidParameters
authCookies	The CookieData field MUST be of the correct format such that the USS can read values out of it, as specified in section 2.2.1.3.	InvalidAuthorizationCookie
protocolVersion	MUST be of the format "x.y", where x is the Major Version and y is the Minor Version number.	InvalidParameters
protocolVersion	Major Version MUST be "1", and Minor Version MUST be "1" or "2".	IncompatibleProtocolVersion

Data processing:

The USS MUST process this message as follows:

1. Parse the *CookieData* in the AuthorizationCookie and extract the ExpirationTime, target groups list, and DSS account GUID.
2. If the cookie has a syntax, formatting, or other error preventing the necessary information from being read out of the EncryptedData field, return SOAP fault with <ErrorCode> set to InvalidAuthorizationCookie.
3. Create a Cookie with the *Expiration* set to an implementation-specific cookie expiration interval sometime in the future.

   <21>
4. Initialize the **EncryptedData** field of the Cookie to a sequence of bytes, as defined in section 2.2.1.5.

Response:

If no errors occur during processing, the USS MUST return the response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault. The SOAP fault SHOULD contain an <ErrorCode> element, as defined in section 2.2.2. If the SOAP fault contains an <ErrorCode> element, its value MUST be one of the following.

If the DSS receives a SOAP fault containing an <ErrorCode> element, it MUST react to the fault, as described below. If the DSS receives a fault that does not contain an <ErrorCode> element, it MUST abort the protocol.

ErrorCode	Description
InvalidParameters	Parameters passed to a Web method are not valid. The "message" part of the exception will contain the parameter name. The DSS MUST abort the

ErrorCode	Description
	protocol.
InternalServerError	An internal error occurred on the server. The DSS MUST abort the protocol.
IncompatibleProtocolVersion	The version of the protocol used by DSS is incompatible with the version used by USS. The DSS MUST abort the protocol.
InvalidAuthorizationCookie	The authorization cookie submitted by the DSS is invalid. The DSS MUST restart the protocol from the beginning.

3.1.4.4 GetConfigData

A DSS calls the **GetConfigData** method to obtain configuration data that governs the metadata and content synchronization phases of the protocol, as specified in section 3.2.4.2.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/GetConfigData" style="document" />
```

Request:

```
<s:element name="GetConfigData">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie"
      type="s1:Cookie" />
      <s:element minOccurs="0" maxOccurs="1"
      name="configAnchor" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: This field MUST be present and set to the **Cookie** returned by the most recent invocation of the **GetCookie** method. The **Cookie** MUST NOT be expired.

configAnchor: If the DSS is calling **GetConfigData** for the first time, this field MUST NOT be present.

If the DSS has successfully called **GetConfigData** in the past, this field MUST contain the NewConfigAnchor value returned by the USS during the last successful call.

Response:

```
<s:element name="GetConfigDataResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
      name="GetConfigDataResult"
```

```
      type="s1:ServerSyncConfigData" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetConfigDataResult: On successful execution of this operation, this object MUST be returned. Its format is as follows:

```
<s:complexType name="ServerSyncConfigData">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1"
    name="CatalogOnlySync" type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1" name="LazySync"
    type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1"
    name="ServerHostsPsfFiles" type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1"
    name="MaxNumberOfUpdatesPerRequest" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1"
    name="NewConfigAnchor" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1"
    name="LanguageUpdateList"
    type="s1:ArrayOfServerSyncLanguageData" />
  </s:sequence>
</s:complexType>
```

CatalogOnlySync: MUST be set to TRUE if the USS supports the Web services, but does not support the USS Content Download for downloading the content files. Otherwise, MUST be set to FALSE.

LazySync: MUST be set to TRUE if the USS defers the download of content for updates until they are needed by its client computers or its DSSs. Otherwise, MUST be set to FALSE.

ServerHostsPsfFiles: Indicates that the USS also hosts the patch storage format (PSF) version of the content files if available for an update revision.

MaxNumberOfUpdatesPerRequest: Specifies the maximum number of revisions that may be requested in the **GetUpdateData (section 3.1.4.6)** operation.

NewConfigAnchor: This field MUST be present. It identifies the point in time that this operation was completed successfully. It is an opaque string that is not interpreted by the DSS. The format of the string is implementation specific.<22>

LanguageUpdateList: This field MUST be present. It identifies the list of languages that are supported by the USS. The array only contains the list of languages whose settings have changed since the time identified by the **configAnchor**.

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

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

ArrayOfServerSyncLanguageData: An array structure, each element of which provides information on a language setting on the USS. The format of each element is as follows:

```
<s:complexType name="ServerSyncLanguageData">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1"
    name="LanguageID" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1"
    name="ShortLanguage" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1"
    name="LongLanguage" type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="Enabled"
    type="s:boolean" />
  </s:sequence>
</s:complexType>
```

LanguageID: Identifies the language corresponding to this entry. This MUST be set to 0 (which is a special value that refers to all languages) or to a language ID referring to a specific language, as specified in [MS-LCID].

ShortLanguage: This field MUST be present. It provides a short name for the language. The value "all" refers to "All languages". All other language short names are as specified in [MS-LCID].

LongLanguage: This field MUST be present. It provides a long name for the language. The value "all" refers to "All languages". All other language long names are as specified in [MS-LCID].

Enabled: MUST be set to TRUE if the USS currently supports updates in the specified language. Otherwise, MUST be set to FALSE.

Request validation:

The USS validates inputs as given below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS with the <ErrorCode> set, as shown in the following table.

Input	Validation conditions	ErrorCode
cookie	MUST be present and not empty.	InvalidParameters
cookie	The EncryptedData MUST be the correct format such that the USS can read values out of it, as specified in section 2.2.1.5.	InvalidCookie
protocolVersion in the cookie **EncryptedData**	MUST be of the format "x.y" where x is the Major Version and y is the Minor Version number.	InvalidParameters
protocolVersion in the cookie **EncryptedData**	Major Version MUST be "1" and Minor Version MUST be "1" or "2".	IncompatibleProtocolVersion

Data processing:

The USS MUST compose a GetConfigDataResponse message in response, as follows:

1. Copy the following Server Configuration Table entries into the **ServerSyncConfigData** structure of the response:

   - **CatalogOnlySync**
   - **LazySync**
   - **ServerHostsPsfFiles**
   - **MaxNumberOfUpdatesPerRequest**

2. Copy the following Server State Table entries into the **ServerSyncConfigData** structure of the response:

   - **ConfigAnchor**

3. Add an entry to the **LanguageUpdateList** element of the **ServerSyncConfigData** structure of the response:

   - **LanguageID** = 0.
   - **ShortLanguage** = "all".
   - **LongLanguage** = "all".
   - **Enabled** ="All Languages" flag value from the Languages Support element in the Server Configuration Table.

4. Copy the Specific Language Settings from the Server Configuration Table entries into the **LanguageUpdateList** element of the **ServerSyncConfigData** structure.

Response:

If no errors occur during processing, the USS MUST return the response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault. The SOAP fault SHOULD contain an <ErrorCode> element, as described in section 2.2.2. If the SOAP fault contains an <ErrorCode> element, its value MUST be one of the following.

If the DSS receives a SOAP fault containing an <ErrorCode> element, it MUST react to the fault, as described below. If the DSS receives a fault that does not contain an <ErrorCode> element, it MUST abort the protocol.

ErrorCode	Description
InvalidParameters	Parameters passed to a Web method are not valid. The "message" part of the exception will contain the parameter name. The DSS MUST abort the protocol.
InternalServerError	An internal error occurred on the server. The DSS MUST abort the protocol.
InvalidCookie	The cookie has a syntax, formatting, or other error. The DSS MUST restart the protocol from the beginning.
IncompatibleProtocolVersion	The version of the protocol used by DSS is incompatible with the version used by USS. The DSS MUST abort the protocol.

3.1.4.5 GetRevisionIdList

A DSS calls the **GetRevisionIdList** method to get the revision IDs for new updates. The DSS provides filters to be used to prune the list of revisions.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/GetRevisionIdList" style="document" />
```

Request:

```
<s:element name="GetRevisionIdList">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie"
      type="s1:Cookie" />
      <s:element minOccurs="0" maxOccurs="1" name="filter"
      type="s1:ServerSyncFilter" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: This field MUST be present and set to the **Cookie** returned by the **GetCookie** operation. The **Cookie** MUST NOT be expired.

filter: This field MUST be present. It describes the revisions for which the DSS is searching. Its format is as follows:

```
<s:complexType name="ServerSyncFilter">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Anchor"
    type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="GetConfig"
    type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1"
    name="Get63LanguageOnly" type="s:boolean" />
    <s:element minOccurs="0" maxOccurs="1" name="Categories"
    type="s1:ArrayOfIdAndDelta" />
    <s:element minOccurs="0" maxOccurs="1"
    name="Classifications" type="s1:ArrayOfIdAndDelta" />
    <s:element minOccurs="0" maxOccurs="1" name="Languages"
    type="s1:ArrayOfLanguageAndDelta" />
  </s:sequence>
</s:complexType>
```

Anchor: This field MUST not be present for the first call to this operation. It SHOULD be set to the Anchor returned in the last successful response from this operation. It identifies the point in time that the last **GetRevisionIdList** operation was completed successfully. This field is an opaque string that is not interpreted by the DSS.<23>

GetConfig: MUST be set to TRUE if the DSS is requesting categories, update classifications, and detectoids that have changed since the time denoted by the Anchor. MUST be set to FALSE if the DSS is requesting software updates.

Get63LanguageOnly: This field MUST be present if a DSS supports version 1.2 or 1.3 of the protocol. This element is not defined in the ServerSyncFilter type for protocol version 1.1. It MUST NOT be present if the DSS supports version 1.1 of the protocol. MUST be set to TRUE if the DSS is requesting up to 63 languages, as specified in section 1.7. A USS implementing version 1.1 of the protocol MUST silently ignore this field.

Categories: This field is reserved for future use. It SHOULD NOT be present, and MUST be ignored on receipt.<24>

Classifications: This field is reserved for future use. It SHOULD NOT be present, and MUST be ignored on receipt.<25>

Languages: This field is reserved for future use. It SHOULD NOT be present, and MUST be ignored on receipt.<26>

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

ArrayOfIdAndDelta: An array structure, each element of which identifies one category for which update revisions are requested. It also specifies if only newer update revisions are requested. The format of each element is as follows:

```
<s:complexType name="IdAndDelta">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="Id"
    type="s2:guid" />
    <s:element minOccurs="1" maxOccurs="1" name="Delta"
    type="s:boolean" />
  </s:sequence>
</s:complexType>
```

Id: Identifies the category for which update revisions are requested.

Delta: Specifies if only update revisions newer than the time denoted by the Anchor are requested.

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

ArrayOfLanguageAndDelta: An array structure, each element of which identifies one language for which update revisions are requested. It also specifies if only newer update revisions are requested. The format of each element is as follows:

```
<s:complexType name="LanguageAndDelta">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="Id"
    type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="Delta"
    type="s:boolean" />
  </s:sequence>
</s:complexType>
```

Id: Identifies the language for which update revisions are requested, as specified in [MS-LCID].

Delta: Set TRUE to indicate that only update revisions newer than the time denoted by the Anchor are requested. Set FALSE to indicate that all update revisions are requested.

Response:

```
<s:element name="GetRevisionIdListResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
      name="GetRevisionIdListResult"
      type="s1:RevisionIdList" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetRevisionIdListResult: On successful execution of this operation, this object MUST be returned. Its format is as follows:

```
<s:complexType name="RevisionIdList">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Anchor"
    type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="NewRevisions"
    type="s1:ArrayOfUpdateIdentity" />
  </s:sequence>
</s:complexType>
```

Anchor: This field MUST be returned. It identifies the point in time that this operation was completed successfully.

NewRevisions: This field MUST be present. It identifies the list of update revisions that match the filter specified in the request and are new since the Anchor specified in the filter. The list MUST be empty if there are no new revisions.

Request validation:

The USS validates inputs as specified below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS with the <ErrorCode> set, as shown in the following table.

Input	Validation conditions	ErrorCode
cookie	MUST be present and not empty.	InvalidParameters
cookie	The EncryptedData MUST be the correct format such that the USS can read values out of it, as specified in section 2.2.1.5.	InvalidCookie
protocolVersion in the cookie **EncryptedData**	MUST be of the format x.y where x is the Major Version and y is the Minor Version number.	InvalidParameters
protocolVersion in the cookie **EncryptedData**	Major Version MUST be 1, and Minor Version MUST be 1 or 2.	IncompatibleProtocolVersion
filter:Anchor	MUST be valid format or NULL or empty string. The format of the Anchor is implementation specific and not defined by the protocol.<27>	InvalidParameters

Data processing:

The USS MUST compose a GetRevisionIdListResponse message in response, as follows:

1. If the DSS protocol version is 1.1, and the USS protocol version is 1.2, force-set the filter's **Get63LanguageOnly** field to TRUE.
2. If **GetConfig** is set to TRUE in the filter, create the **NewRevisions** list in the response as follows:
    1. Select entries from the Categories, Update Classifications, and Detectoids Tables using the following rules:
        - If the GUID portion of the ID for multiple entries is the same, select only the entry with the highest Revision Number.
        - The LastChangedAnchor in the entry is newer that the Anchor specified in the **GetRevisionIdList** request.
    2. Add the selected entry to the **NewRevisions** field of the response.
3. If **GetConfig** is set to FALSE in the filter, create the **NewRevisions** list in the response as follows:
    1. Select entries from the Revisions Table using the following rules:
        - If the GUID portion of the ID for multiple entries is the same, select only the entry with the highest Revision Number.
        - The LastChangedAnchor in the entry is newer that the Anchor specified in the **GetRevisionIdList** request.
    2. Add the selected entry to the **NewRevisions** field of the response.
4. Set the **Anchor** field in the response to mark the time this operation was completed.

Response:

If no errors occur during processing, the USS MUST return the success response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault. The SOAP fault SHOULD contain an <ErrorCode> element, as described in section 2.2.2. If the SOAP fault contains an <ErrorCode> element, its value MUST be one of the following.

If the DSS receives a SOAP fault containing an <ErrorCode> element, it MUST react to the fault, as described below. If the DSS receives a fault that does not contain an <ErrorCode> element, it MUST abort the protocol.

ErrorCode	Description
InvalidParameters	Parameters passed to a Web method are not valid. The "message" part of the exception will contain the parameter name. The DSS MUST abort the protocol.
InternalServerError	An internal error occurred on the server. The DSS MUST abort the protocol.
InvalidCookie	The cookie has a syntax, formatting, or other error. The DSS MUST restart the protocol from the beginning.
IncompatibleProtocolVersion	The version of the protocol used by DSS is incompatible with the version used by USS. The DSS MUST abort the protocol.

3.1.4.6 GetUpdateData

A DSS calls the **GetUpdateData** method to get the full update metadata for the list of revision IDs requested.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/GetUpdateData" style="document" />
```

Request:

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

cookie: This field MUST be present and set to the **Cookie** returned by the **GetCookie** operation. The **Cookie** MUST NOT be expired.

updateIds: This field MUST be present. The array MUST contain at least one element. The array MUST contain a maximum number of elements as specified by the **MaxNumberOfUpdatesPerRequest** field returned by the **GetConfigData (section 3.1.4.4)** operation. It contains a list of update identities, each identifying a specific update revision for which update metadata is requested, as specified in section 2.2.1.6.

Response:

```
<s:element name="GetUpdateDataResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
      name="GetUpdateDataResult"
      type="s1:ServerUpdateData" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetUpdateDataResult: On successful execution of this operation, this object MUST be returned. Its format is as follows:

```
<s:complexType name="ServerUpdateData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="updates"
    type="s1:ArrayOfServerSyncUpdateData" />
    <s:element minOccurs="0" maxOccurs="1" name="fileUrls"
    type="s1:ArrayOfServerSyncUrlData" />
  </s:sequence>
</s:complexType>
```

updates: This field MUST be present. Each element in the array contains the update metadata. Its type is **ArrayOfServerSyncUpdateData**, whose format is given below, after the description of the **fileUrls** field.

fileUrls: This field MUST be present. It provides information on the content files referenced by the updates field. Its format is given below.

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

ArrayOfServerSyncUpdateData: An array structure, each element of which provides information on an update revision. The format of each element is as follows:

```
<s:complexType name="ServerSyncUpdateData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Id"
    type="s1:UpdateIdentity" />
    <s:element minOccurs="0" maxOccurs="1"
```

```
      name="XmlUpdateBlob" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1"
      name="FileDigestList"
      type="s1:ArrayOfBase64Binary" />
      <s:element minOccurs="0" maxOccurs="1"
      name="XmlUpdateBlobCompressed"
      type="s:base64Binary" />
    </s:sequence>
  </s:complexType>
```

Id: This field MUST be present. It identifies the update revision that is described in this structure. The details of UpdateIdentity are specified in section 2.2.1.7.

XmlUpdateBlob: This field MUST be present if **XmlUpdateBlobCompressed** is absent. This field MUST be absent if **XmlUpdateBlobCompressed** is present. The field contains the metadata associated with the update.

FileDigestList: Contains an array of file digests, one for each content file associated with the update revision. The file digest is an SHA-1 hash of the content of the file. If the update revision does not have any associated content file, this field MUST be absent.

XmlUpdateBlobCompressed: This field MUST be absent if **XmlUpdateBlob** is present. It contains metadata associated with the update, compressed using a LZX variant of the Lempel-Ziv compression algorithm. For more information, see [MSDN-CAB]. The LZX window size used for compressing this field is 2 MB.

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

ArrayOfServerSyncUrlData: An array structure, each element of which provides information on one content file. The format of each element is as follows:

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

FileDigest: An SHA-1 hash of the content file.

MUUrl: An HTTP URL specifying the location on the Internet from which client computers can download this file. This field MUST NOT be present if no such location is available.<28>

UssUrl: Unused. This field SHOULD NOT be present, and MUST be ignored on receipt.<29>

Request validation:

The USS validates inputs as given below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS with the <ErrorCode> set, as shown in the following table.

Input	Validation conditions	ErrorCode
cookie	MUST be present and non-empty.	InvalidParameters
cookie	The EncryptedData MUST be the correct format such that the USS can read values out of it, as specified in section 2.2.1.5.	InvalidCookie
protocolVersion in the cookie **EncryptedData**	MUST be of the format x.y where x is the Major Version and y is the Minor Version number.	InvalidParameters
protocolVersion in the cookie **EncryptedData**	Major Version MUST be 1, and Minor Version MUST be 1 or 2.	IncompatibleProtocolVersion
updateIds	MUST not be NULL.	InvalidParameters
updateIds	Number of elements in the array MUST NOT be greater than the **MaxNumberOfUpdatesPerRequest** value returned in the **GetConfigData** (section 3.1.4.4) response message.	InvalidParameters

Data processing:

The USS MUST process this message as follows:

1. Search for the requested **updateIds** in the Categories Table, Update Classifications Table, Detectoids Table, and Update Revisions Table.
2. For each entry found in step 1, create an entry in the **ArrayOfServerSyncUpdateData** element of the response. Initialize the entry as follows:
    1. For categories, update classifications, and detectoids, set the UpdateIdentity element to the CategoryIdentity, ClassificationIdentity, and DetectoidIdentity respectively and the XML metadata from the table.
    2. For update Revisions found in the Revision Table, set the UpdateIdentity to the UpdateIdentity from the table.
    3. For each ServerSyncUpdateData entry initialize the **XmlUpdateBlob** or the **XmlUpdateBlobCompressed** element with the XML metadata stored in the table.

        <30>
    4. For each update Revision, initialize the **FileDigestList** element of the ServerSyncUpdateData with the **FileDigest** of the content files associated with the Revision Table entry.

3. For each content file of an update Revision found in the Revision Table in step 2, create an entry in the **ArrayOfServerSyncUrlData** element of the response. Initialize the **FileDigest** and **MUUrl** fields of this entry from the information in the Revision Table.

Response:

If no errors occur during processing, the USS MUST return the success response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault. The SOAP fault SHOULD contain an <ErrorCode> element, as defined in section 2.2.2. If the SOAP fault contains an <ErrorCode> element, its value MUST be one of the following.

If the DSS receives a SOAP fault containing an <ErrorCode> element, it MUST react to the fault, as described below. If the DSS receives a fault that does not contain an <ErrorCode> element, it MUST abort the protocol.

ErrorCode	Description
InvalidParameters	Parameters passed to a Web method are not valid. The "message" part of the exception will contain the parameter name. The DSS MUST abort the protocol.
InternalServerError	An internal error occurred on the server. The DSS MUST abort the protocol.
InvalidCookie	The cookie has a syntax, formatting, or other error. The DSS MUST restart the protocol from the beginning.
IncompatibleProtocolVersion	The version of the protocol used by DSS is incompatible with the version used by USS. The DSS MUST abort the protocol.

3.1.4.7 GetRelatedRevisionsForUpdates

This method is not used in the protocol. The USS SHOULD/MUST ignore it if invoked.

3.1.4.8 GetDeployments

A DSS calls the **GetDeployments** method to get the list of deployments that are new or changed since the last successful **GetDeployments** call made by this DSS. This call MUST NOT be invoked unless the DSS is configured to be a "Replica Server", as specified in section 3.1.1.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/GetDeployments" style="document" />
```

Request:

```
<s:element name="GetDeployments">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie"
```

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

cookie: This field MUST be present and set to the **Cookie** returned by the **GetCookie** operation. The **Cookie** MUST NOT be expired.

deploymentAnchor: This identifies the point in time that the last **GetDeployments** operation was completed successfully. If this is the first time the **GetDeployments** operation is called, this field MUST NOT be present. For subsequent calls to **GetDeployments**, it MUST be set to the Anchor returned in the last **GetDeployments** call.

syncAnchor: This field MUST be present. It MUST be set to the Anchor returned in the last successful **GetRevisionIdList** operation response. It identifies the point in time that the last **GetRevisionIdList** operation was completed successfully.

Response:

```
<s:element name="GetDeploymentsResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
      name="GetDeploymentsResult"
      type="s1:ServerSyncDeploymentResult" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetDeploymentsResult: On successful execution of this operation, this object MUST be returned. Its format is as follows:

```
<s:complexType name="ServerSyncDeploymentResult">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Anchor"
    type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Groups"
    type="s1:ArrayOfServerSyncTargetGroup" />
    <s:element minOccurs="0" maxOccurs="1" name="Deployments"
    type="s1:ArrayOfServerSyncDeployment" />
    <s:element minOccurs="0" maxOccurs="1"
    name="DeadDeployments" type="s1:ArrayOfGuid" />
    <s:element minOccurs="0" maxOccurs="1"
    name="HiddenUpdates" type="s1:ArrayOfGuid" />
    <s:element minOccurs="0" maxOccurs="1"
    name="AcceptedEulas" type="s1:ArrayOfGuid" />
  </s:sequence>
</s:complexType>
```

Anchor: This field MUST be present. It identifies the point in time that the last **GetDeployments** operation was completed successfully.

Groups: This field MUST be present. It provides information on all client computer target groups configured on the USS. The target group information is as specified in the **ServerSyncTargetGroup** data type in this section.

Deployments: This field MUST be present, but it may contain zero elements in the array. It provides information on all the deployments that have been added to the USS between the time identified by the **deploymentAnchor** and the time identified by the **syncAnchor** inputs passed in the request. The deployment information is as specified in the **ServerSyncDeployment** data type in this section.

DeadDeployments: This field MUST be present, but it may contain zero elements in the array. It provides information on the list of deployments that have been removed from the USS between the time identified by the **deploymentAnchor** and the time identified by the **syncAnchor** inputs passed in the request.

HiddenUpdates: This field MUST be present, but it may contain zero elements in the array. It lists all of the updates that have been hidden by the administrator on the USS and are therefore unavailable for deployment on client computers. Unlike deployments and **DeadDeployments**, this is a full list rather than a list of the elements changed since the last time identified by the **deploymentAnchor**.

AcceptedEulas: This field MUST be present, but it may contain zero elements in the array. It lists all the End User License Agreements (EULAs) that have been accepted by the administrator on the USS. Unlike deployments and **DeadDeployments**, this is a full list that does not take the **deploymentAnchor** into account.

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

ArrayOfServerSyncTargetGroup: An array structure, each element of which provides information on one target group. The format of each element is as follows:

```
<s:complexType name="ServerSyncTargetGroup">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1"
    name="TargetGroupID" type="s2:guid" />
    <s:element minOccurs="0" maxOccurs="1" name="Name"
    type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="IsBuiltin"
    type="s:boolean" />
  </s:sequence>
</s:complexType>
```

TargetGroupID: This field uniquely identifies the target group.

Name: This field MUST be present. It specifies a human-readable text name for the target group.

IsBuiltin: This field is set to FALSE for a target group that is created by an administrator. Otherwise, it is set to TRUE.

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

ArrayOfServerSyncDeployment: An array structure, each element of which provides information on one deployment. The format of each element is as follows:

```
<s:complexType name="ServerSyncDeployment">
  <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:element minOccurs="1" maxOccurs="1" name="Action"
    type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="AdminName"
    type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="Deadline"
    type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="IsAssigned"
    type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1" name="GoLiveTime"
    type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1"
    name="DeploymentGuid" type="s2:guid" />
    <s:element minOccurs="1" maxOccurs="1"
    name="TargetGroupId" type="s2:guid" />
    <s:element minOccurs="1" maxOccurs="1"
    name="DownloadPriority" type="s:unsignedByte" />
  </s:sequence>
</s:complexType>
```

UpdateId: This field identifies the update that is referred to in this deployment.

RevisionNumber: This field identifies the specific revision of the update referred to in this deployment.

Action: This field specifies the specific Action defined by this deployment. The value MUST be one of the following:

Value	Action
0	Install the update revision.
1	Uninstall the update revision.
2	Scan for the presence of the update revision.

Value	Action
3	Block.

Deadline: Specifies the date/time by which the update revision referred to in this deployment must be installed on the computers in the target group referred to in this deployment. If there is no deadline associated with this deployment, the deadline value MUST be set to the following.

Date (Year-Month-Date) = 9999-12-31

Time (Hours:Minutes:Seconds)= 23:59:59.9999999

IsAssigned: This field is not used within this protocol and is used when client computers communicate with the update server. The value in this field must be preserved for use with client computers. It specifies if the update revision must be installed by the Automatic Update component of the client, as specified in [MS-WUSP].

GoLiveTime: This field specifies the date/time after which the update revision must be made available for deployment.

DeploymentGuid: This field MUST be set to a GUID that uniquely identifies this deployment.

TargetGroupId: This field identifies the target group for which this deployment action applies.

DownloadPriority: This field specifies a value indicating how the client computer should prioritize its download of content for this deployment relative to other deployments. The value MUST be an integer value between 1 and 3, inclusive, where 1 indicates the lowest priority, and 3 indicates the highest priority.

Request validation:

The USS validates inputs as given below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS with the <ErrorCode> set, as shown in the following table.

Input	Validation conditions	ErrorCode
cookie	MUST be present and not empty.	InvalidParameters
cookie	The EncryptedData MUST be the correct format such that the USS can read values out of it, as specified in section 2.2.1.5.	InvalidCookie
protocolVersion in the cookie **EncryptedData**	MUST be of the format x.y where x is the Major Version and y is the Minor Version number.	InvalidParameters
protocolVersion in the cookie **EncryptedData**	Major Version MUST be 1, and Minor Version MUST be 1 or 2.	IncompatibleProtocolVersion
deploymentAnchor	MUST be a string with the correct format, an empty string, or not present. The format of the Anchor is implementation specific and not defined by the protocol.<31>	InvalidParameters
syncAnchor	MUST be a valid format and not NULL and not an empty string. The format of the Anchor is implementation specific and not	InvalidParameters

Input	Validation conditions	ErrorCode
	defined by the protocol.<32>	

Data processing:

The USS MUST compose a GetDeploymentsResponse message in response, as follows:

1. Search the Target Groups Table for all target groups defined on the USS. For each entry found in the table create a **ServerSyncTargetGroup** entry in the **Groups** field of the response. Initialize the **ServerSyncTargetGroup** entry from the entry in the Targets Group Table.

2. Search the Deployments Table for all deployments that have been added between the time identified by the **deploymentAnchor** and the time identified by the sync Anchor. For each entry found in the Deployments Table create a **ServerSyncDeployment** entry in the **Deployments** field of the response. Initialize the **ServerSyncDeployment** entry from the entry in the Deployments Table.

3. Search the Deployments Table for all deployments that have been deleted between the time identified by the **deploymentAnchor** and the time identified by the **syncAnchor** and add their GUIDs to the **DeadDeployments** array in the response.

4. Search the Revisions Table for all entries whose **Hidden** field is set TRUE and add their GUIDs to the **HiddenUpdates** array in the response.

5. Search the EULAs Table for all entries whose **Accepted** field is set TRUE and add their GUIDs to the **AcceptedEulas** array in the response.

Response:

If no errors occur during processing, the USS MUST return the success response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault. The SOAP fault SHOULD contain an <ErrorCode> element, as described in section 2.2.2. If the SOAP fault contains an <ErrorCode> element, its value MUST be one of the following.

If the DSS receives a SOAP fault containing an <ErrorCode> element, it MUST react to the fault, as described below. If the DSS receives a fault that does not contain an <ErrorCode> element, it MUST abort the protocol.

ErrorCode	Description
InvalidParameters	Parameters passed to a Web method are not valid. The "message" part of the exception will contain the parameter name. The DSS MUST abort the protocol.
InternalServerError	An internal error occurred on the server. The DSS MUST abort the protocol.
InvalidCookie	The cookie has a syntax, formatting, or other error. The DSS MUST restart the protocol from the beginning.
IncompatibleProtocolVersion	The version of the protocol used by DSS is incompatible with the version used by USS. The DSS MUST abort the protocol.

3.1.4.9 DownloadFiles

A DSS invokes the **DownloadFiles** method to initiate the Content Download process on a USS for a set of files from the parent of the USS. This method is typically invoked by a DSS after a Content Download attempt from the USS fails with a "File Not Found" condition.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/DownloadFiles" style="document" />
```

Request:

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

cookie: This field MUST be present and set to the **Cookie** returned by the **GetCookie** operation. The **Cookie** MUST NOT be expired.

fileDigestList: This field MUST be present. The array MUST contain at least one element and, at most, 100 elements. Each element contains the SHA-1 hash value for an update content file. The hash value is one of those returned by the USS in **GetUpdateData** operation.

Response:

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

This type has no fields.

Request validation:

The USS validates inputs as given below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS with the <ErrorCode> set, as shown in the following table.

Input	Validation conditions	ErrorCode
cookie	MUST be present and not empty.	InvalidParameters

Input	Validation conditions	ErrorCode
cookie	The EncryptedData MUST be the correct format such that the USS can read values out of it, as specified in section 2.2.1.5.	InvalidCookie
protocolVersion in the cookie **EncryptedData**	MUST be of the format x.y where x is the Major Version and y is the Minor Version number.	InvalidParameters
protocolVersion in the cookie **EncryptedData**	Major Version MUST be 1, and Minor Version MUST be 1 or 2.	IncompatibleProtocolVersion
fileDigestList	MUST be present, and the number of elements in the array MUST NOT be greater than 100.	InvalidParameters

Data processing:

The USS MUST process this message as follows:

1. Locate the entry corresponding to each requested **FileDigest** in the Revisions Table.
2. For all **FileDigest** values not found in the table, report back a SOAP fault to the DSS with ErrorCode set to "FileDigestsMissing" and the <Message> XML element within the <soap:Fault><detail> element containing the list of missing **FileDigest** values each separated by a "|" character.
3. For each entry found in the Revisions Table, initiate an HTTP download as follows:
   - If the Parent Server State's **CatalogOnlySync** flag is set to FALSE, construct the URL as given below and initiate the download.
   - If the Parent Server State's **CatalogOnlySync** flag is TRUE, initiate the download from using the **MUUrl** for the file.
4. If the download fails, silently ignore the error.
5. If the download succeeds, verify that the **FileDigest** field of the corresponding entry in the Content Store matches the SHA-1 hash value computed over the contents of the file. If the values do not match, delete the downloaded file. Otherwise, store the file in the Content Store.

Response:

If no errors occur during processing, the USS MUST return the success response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault. The SOAP fault SHOULD contain an <ErrorCode> element, as described in section 2.2.2. If the SOAP fault contains an <ErrorCode> element, its value MUST be one of the following.

If the DSS receives a SOAP fault containing an <ErrorCode> element, it MUST react to the fault, as described below. If the DSS receives a fault that does not contain an <ErrorCode> element, it MUST abort the protocol.

ErrorCode	Description
InvalidParameters	Parameters passed to a Web method are not valid. The "message" part of

ErrorCode	Description
	the exception will contain the parameter name. The DSS MUST abort the protocol.
InternalServerError	An internal error occurred on the server. The DSS MUST abort the protocol.
InvalidCookie	The cookie has a syntax, formatting, or other error. The DSS MUST restart the protocol from the beginning.
IncompatibleProtocolVersion	The version of the protocol used by DSS is incompatible with the version used by USS. The DSS MUST abort the protocol.
FileDigestsMissing	Some or all of the requested FileDigest values are not known to the USS. This may be due to deletion of some unneeded updates on the USS. The DSS MUST restart the protocol from the beginning.

3.1.4.10 Ping

This method is not used in the Windows Update Services: Server-Server Protocol. The DSS MUST NOT invoke it and the USS MUST silently ignore it.

3.1.4.11 GetRollupConfiguration

A DSS invokes the **GetRollupConfiguration** method to obtain information on the USS's report rollup configuration settings.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/GetRollupConfiguration" style="document" />
```

Request:

```
<s:element name="GetRollupConfiguration">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie" type="tns:Cookie" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: This field is reserved for future use. This field MUST be present, and the value MUST be set to the following:

Expiration (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999

EncryptedData = zero length array

Response:

```
<s:element name="GetRollupConfigurationResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetRollupConfigurat
ionResult" type="tns:RollupConfiguration" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetRollupConfigurationResult: On successful execution of this operation, this object MUST be returned. Its format is as follows:

```
<s:complexType name="RollupConfiguration">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="DoDetailedRollup"
type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1" name="RollupResetGuid"
type="s1:guid" />
    <s:element minOccurs="1" maxOccurs="1" name="ServerId"
type="s1:guid" />
    <s:element minOccurs="1" maxOccurs="1" name="RollupDownstreamS
erversMaxBatchSize" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="RollupComputersMax
BatchSize" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="GetOutOfSyncComputers
MaxBatchSize" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="RollupComputerStatus
MaxBatchSize" type="s:int" />
  </s:sequence>
</s:complexType>
```

DoDetailedRollup: This field MUST be present. It MUST be set to TRUE if the USS has been configured to collect detailed information on client computers from the DSS. Otherwise, it MUST be set to FALSE.

RollupResetGuid: This field is reserved for future use. This **field** MUST be present and MUST contain a well-formed GUID, adhering to the schema of the guid type as defined in section 2.2.1.1. However, the DSS MUST ignore its value.

ServerId: This field MUST be present. It MUST be a GUID that identifies the USS.

RollupDownstreamServersMaxBatchSize: This field MUST be present. It specifies the maximum number of DownstreamServerRollupInfo elements the DSS may pass in a call to **RollupDownstreamServers**. The value MUST be greater than zero.

RollupComputersMaxBatchSize: This field MUST be present. It specifies the maximum number of ComputerRollupInfo elements the DSS may pass in a call to **RollupComputers**. The value MUST be greater than zero.

GetOutOfSyncComputersMaxBatchSize: This field MUST be present. It specifies the maximum number of ComputerLastRollupNumber elements the DSS may pass in a call to **GetOutOfSyncComputers**. The value MUST be greater than zero.

RollupComputerStatusMaxBatchSize: This field MUST be present. It specifies the maximum number of ComputerStatusRollupInfo elements the DSS may pass in a call to **RollupComputerStatus**. The value MUST be greater than zero.

Request validation:

The USS performs no input validation.

Data processing:

The USS MUST compose a GetRollupConfigurationResponse message in response, as follows:

1. **DoDetailedRollup**: Set to the **DoDetailedRollup** value stored in the Server Configuration Table.
2. **RollupResetGuid**: Set to any well-formed GUID.
3. **ServerId**: Set to the **ServerID** value stored in the Server Configuration Table.
4. **RollupDownstreamServersMaxBatchSize**: Set to the maximum number of **DownstreamServerRollupClientSummary** structures the USS will accept in a single **RollupDownstreamServers** request, summed across all **DownstreamServerRollupInfo** structures in the request. The value is implementation-specific.

   <33>
5. **RollupComputersMaxBatchSize**: Set to the maximum number of ComputerRollupInfo structures the USS will accept in a single **RollupComputers** request. The value is implementation-specific.

   <34>
6. **GetOutOfSyncComputersMaxBatchSize**: Set to the maximum number of **ComputerLastRollupNumber** structures the USS will accept in a single **GetOutOfSyncComputers** request. The value is implementation-specific.

   <35>
7. **RollupComputerStatusMaxBatchSize**: Set to the maximum number of ComputerStatusRollupInfo structures the USS will accept in a single **RollupComputerStatus** request. The value is implementation-specific.

   <36>

Response:

If no errors occur during processing, the USS MUST return the response to the DSS. If an error occurs during processing, the USS MUST return a SOAP fault indicating that an error occurred. This protocol does not define a mechanism for returning information on the cause of the failure. The DSS MUST abort the protocol when a SOAP fault is received.

3.1.4.12 RollupDownstreamServers

A DSS invokes the **RollupDownstreamServers** method to send information on itself (and on DSSs that synchronize updates from it) to the USS.

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

The SOAP operation is defined as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/RollupDownstreamServers" style="document" />
```

Request:

```
<s:element name="RollupDownstreamServers">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie" type="tns:C
ookie" />
      <s:element minOccurs="1" maxOccurs="1" name="clientTime" type="s
:dateTime" />
      <s:element minOccurs="0" maxOccurs="1" name="downstreamServers"
type="tns:ArrayOfDownstreamServerRollupInfo" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: This field is reserved for future use. This field MUST be present, and the value MUST be set to the following:

Expiration (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999

EncryptedData = zero length array

clientTime: This field MUST be present and set to the current time on the DSS.

downstreamServers: This field MUST be present, but may contain zero elements in the array. Each element in the array contains information on an update server. Its format is as follows:

```
<s:complexType name="ArrayOfDownstreamServerRollupInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="DownstreamServ
erRollupInfo" nillable="true" type="tns:DownstreamServerRollupInfo" />
  </s:sequence>
</s:complexType>
```

ArrayOfDownstreamServerRollupInfo: An array structure, each element of which contains information on one update server (either the update server sending the request or one of its descendant DSSs). The format of each element is as follows:

```
<s:complexType name="DownstreamServerRollupInfo">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ServerId" type="s1:gu
id" />
    <s:element minOccurs="0" maxOccurs="1" name="FullDomainName" type=
"s:string" />
```

```
    <s:element minOccurs="1" maxOccurs="1" name="LastSyncTime" type="s
:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="ParentServerId" type=
"s1:guid" />
    <s:element minOccurs="0" maxOccurs="1" name="Version" type="s:stri
ng" />
    <s:element minOccurs="1" maxOccurs="1" name="IsReplica" type="s:bo
olean" />
    <s:element minOccurs="1" maxOccurs="1" name="LastRollupTime" type=
"s:dateTime" />
    <s:element minOccurs="0" maxOccurs="1" name="ServerSummary" type="
tns:DownstreamServerRollupServerSummary" />
    <s:element minOccurs="0" maxOccurs="1" name="ClientSummaries" type
="tns:ArrayOfDownstreamServerRollupClientSummary" />
  </s:sequence>
</s:complexType>
```

ServerId: This field MUST be present. It MUST be a GUID that identifies the update server being described.

FullDomainName: This field MUST be present. It MUST be set to the fully-qualified DNS name of the update server.

LastSyncTime: This field MUST be present. It MUST be set to the time when the update server last synchronized updates from its USS. If the update server has never synchronized with its USS, this value MUST be set to the following:

Date (Year-Month-Date) = 1753-01-01

Time (Hours:Minutes:Seconds)= 00:00:00

ParentServerId: This field MUST be present. It MUST be a GUID that identifies the update server's parent USS. If the update server's parent USS is the update server receiving the **RollupDownstreamServers** request, this field MUST be set to "00000000-0000-0000-0000-000000000000".

Version: This field MUST be present. It MUST be a string that identifies the version number of the update server. The string MUST contain no more than 32 characters and must consist of one to four integers that are separated by periods (".").<37>

IsReplica: This field MUST be present. It MUST be TRUE if the update server is a Replica DSS; otherwise, it MUST be FALSE.

LastRollupTime: This field MUST be present. If the update server has previously reported information on itself to its parent USS using **RollupDownstreamServers**, this field MUST be set to the time when it last reported. Otherwise, if the update server has never reported information on itself to its parent USS, this value MUST be set to the following:

Date (Year-Month-Date) = 1753-01-01

Time (Hours:Minutes:Seconds)= 00:00:00

ServerSummary: This field MUST be present. It contains aggregate summary information on the update server, the updates available on it, and the client computers that get updates from it. Its format is as follows:

```
<s:complexType name="DownstreamServerRollupServerSummary">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="UpdateCount" type="s:
int" />
    <s:element minOccurs="1" maxOccurs="1" name="DeclinedUpdateCount"
type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="ApprovedUpdateCount"
type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="NotApprovedUpdateCoun
t" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="UpdatesWithStaleUpdat
eApprovalsCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="ExpiredUpdateCount" t
ype="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="CriticalOrSecurityUpd
atesNotApprovedForInstallCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="WsusInfrastructureUpd
atesNotApprovedForInstallCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="UpdatesWithClientErro
rsCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="UpdatesWithServerErro
rsCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="UpdatesNeedingFilesCo
unt" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="UpdatesNeededByComput
ersCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="UpdatesUpToDateCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="CustomComputerTargetG
roupCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="ComputerTargetCount"
type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="ComputerTargetsNeedin
gUpdatesCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="ComputerTargetsWithUp
dateErrorsCount" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="ComputersUpToDateCoun
t" type="s:int" />
  </s:sequence>
</s:complexType>
```

DownstreamServerRollupServerSummary: A structure that contains summary information on the update server, the updates available on it, and the client computers that get updates from it.

UpdateCount: This field MUST be present. It indicates the number of updates available on the update server. The value MUST be 0 or greater.

DeclinedUpdateCount: This field MUST be present. It indicates the number updates that have been marked as hidden on the update server. The value MUST be 0 or greater.

ApprovedUpdateCount: This field MUST be present. It indicates the number of updates for which there exists at least one deployment with an Action of 0 (Install) or 1 (Uninstall) on the update server. The value MUST be 0 or greater.

NotApprovedUpdateCount: This field MUST be present. It indicates the number of updates that are neither marked as hidden nor have any deployments with an Action of 0 (Install) or 1 (Uninstall) on the update server. The value MUST be 0 or greater.

UpdatesWithStaleUpdateApprovalsCount: This field MUST be present. It indicates the number of updates that have at least one deployment with an Action of 0 (Install) or 1 (Uninstall) on the update server where the Deployment is associated with a revision of the update other than the latest revision. The value MUST be 0 or greater.

CriticalOrSecurityUpdatesNotApprovedForInstallCount: This field MUST be present. It indicates the number of updates that are intended to repair a security issue on the client computers, or are otherwise considered critical to the operation of the client computers, that have no Deployments with an Action of 0 (Install) on the update server. The value MUST be 0 or greater.

WsusInfrastructureUpdatesNotApprovedForInstallCount: This field MUST be present. It indicates the number of updates that client computers need to have installed to continue to get updates from the update server, and that have no deployments with an Action of 0 (Install) on the update server. The value MUST be 0 or greater.

UpdatesWithClientErrorsCount: This field MUST be present. It indicates the number of updates that at least one client computer has attempted and failed to install. The value MUST be 0 or greater.

UpdatesWithServerErrorsCount: This field MUST be present. It indicates the number of updates for which the update server has attempted to download content but was unable to complete the download due to an error. The value MUST be 0 or greater.

UpdatesNeedingFilesCount: This field MUST be present. It indicates the number of updates for which the update server must download content but has not completed the download. The value MUST be 0 or greater.

UpdatesNeededByComputersCount: This field MUST be present. It indicates the number of updates that have at least one client computer to which it is applicable but not yet installed on. The value MUST be 0 or greater.

UpdatesUpToDateCount: This field MUST be present. It indicates the number of updates that are known to be installed on all client computers to which it is applicable. The value MUST be 0 or greater.

CustomComputerTargetGroupCount: This field MUST be present. It indicates the number of target groups that have been created on this update server or that have been received from the USS. This count MUST NOT include target groups that were created during the initial setup of the update server. The value MUST be 0 or greater.<38>

ComputerTargetCount: This field MUST be present. It indicates the number of client computers that get updates from this update server. The value MUST be 0 or greater.

ComputerTargetsNeedingUpdatesCount: This field MUST be present. It indicates the number of client computers that get updates from this update server, on which at least one update is known to be applicable, but that has not yet attempted to install the update. The value MUST be 0 or greater.

ComputerTargetsWithUpdateErrorsCount: This field MUST be present. It indicates the number (of client computers) that gets updates from this update server (which failed in its last attempt to successfully complete the install of at least one update). The value MUST be 0 or greater.

ComputersUpToDateCount: This field MUST be present. It indicates the number of client computers known to have successfully installed all updates that are applicable to them. The value MUST be 0 or greater.

ClientSummaries: This field MUST be present, but it may contain zero elements in the array. Each element in the array contains summary information on the client computers that get updates from the update server. Its format is as follows:

```
<s:complexType name="ArrayOfDownstreamServerRollupClientSummary">
  <s:sequence>
```

```
    <s:element minOccurs="0" maxOccurs="unbounded" name="DownstreamSer
verRollupClientSummary" nillable="true" type="tns:DownstreamServerRoll
upClientSummary" />
  </s:sequence>
</s:complexType>
```

ArrayOfDownstreamServerRollupClientSummary: An array structure, each element of which contains summary information on a group (of client computers) that gets updates from the update server or its descendent update servers, identified by operating system version and locale. The format of each element is as follows:

```
<s:complexType name="DownstreamServerRollupClientSummary">
  <s:sequence>
    <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="OSServicePackMajorNum
ber" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="OSServicePackMinorNum
ber" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="OSLocale" type="s:str
ing" />
    <s:element minOccurs="1" maxOccurs="1" name="SuiteMask" type="s:sh
ort" />
    <s:element minOccurs="1" maxOccurs="1" name="OldProductType" type=
"s:unsignedByte" />
    <s:element minOccurs="1" maxOccurs="1" name="NewProductType" type=
"s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="SystemMetrics" type="
s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="ProcessorArchitecture
" type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="Count" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="ActivitySummaries" ty
pe="tns:ArrayOfDownstreamServerRollupClientActivitySummary" />
  </s:sequence>
</s:complexType>
```

OSMajorVersion: This field MUST be present. It indicates the operating system major version number of the client computers. The value MUST be 0 or greater.

OSMinorVersion: This field MUST be present. It indicates the operating system minor version number of the client computers. The value MUST be 0 or greater.

OSBuildNumber: This field MUST be present. It indicates the operating system build number of the client computers. The value MUST be 0 or greater.

OSServicePackMajorNumber: This field MUST be present. It indicates the operating system service pack major version number of the client computers. The value MUST be 0 or greater.

OSServicePackMinorNumber: This field MUST be present. It indicates the operating system service pack minor version number of the client computers. The value MUST be 0 or greater.

OSLocale: This field MUST be present. It MUST indicate the operating system locale of the client computers, using language IDs, as specified in [MS-LCID].

SuiteMask: This field MUST be present. It MAY contain additional data to be used to identify the operating system used by the client computers.

OldProductType: This field MUST be present. It MAY contain additional data to be used to identify the operating system used by the client computers.

NewProductType: This field MUST be present. It MAY contain additional data to be used to identify the operating system used by the client computers.

SystemMetrics: This field MUST be present. It MAY contain additional data to be used to identify the operating system used by the client computers.

ProcessorArchitecture: This field MUST be present. It indicates the CPU architecture for which the operating system on the client computers is built.<39>

Count: This field MUST be present. It indicates the number of client computers in this group.

ActivitySummaries: This field MUST be present, but it may contain zero elements in the array. Each element in the array contains summary information describing the activity of the client computers in this group with respect to a single update. Its format is as follows:

```
<s:complexType name="ArrayOfDownstreamServerRollupClientActivitySummary">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="DownstreamServer
RollupClientActivitySummary" nillable="true" type="tns:DownstreamServerRo
llupClientActivitySummary" />
  </s:sequence>
</s:complexType>
```

ArrayOfDownstreamServerRollupClientActivitySummary: An array structure, each element of which contains summary information describing the activity of the client computers in this group with respect to a single update. The format of each element is as follows:

```
<s:complexType name="DownstreamServerRollupClientActivitySummary">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="UpdateId" type="s1:gui
d" />
    <s:element minOccurs="1" maxOccurs="1" name="RevisionNumber" type=
"s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="InstallSuccessCount"
type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="InstallFailureCount"
type="s:int" />
  </s:sequence>
</s:complexType>
```

UpdateId: This field MUST be present. It identifies the update for which the client computer's activity is being described.

RevisionNumber: This field MUST be present. It identifies the specific Revision of the update for which the client computer's activity is being described.

InstallSuccessCount: This field MUST be present. It indicates the number of times this update has been installed successfully on client computers in this group since the last time information on this update server was reported to its parent using the **RollupDownstreamServers** method. Note that this is the number of successful install operations, not the number of client computers that have installed the update. If a single client computer has installed the update multiple times, it is counted multiple times, once per successful install operation.

InstallFailureCount: This field MUST be present. It indicates the number of times client computers in this group attempted to install this update but failed since the last time information on this update server was reported to its parent using the **RollupDownstreamServers** method. Note that this is the number of unsuccessful install attempts, not the number of client computers that have failed. If a single client computer attempts unsuccessfully to install the update multiple times, it is counted multiple times, once per failed install attempt.

Response:

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

This type has no fields.

Request validation:

The USS validates inputs as given below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS.

Input	Validation conditions
downstreamServers	MUST be present, and the total number of **DownstreamServerRollupClientSummary** structures in the request (summed across all **DownstreamServerRollupInfo** structures in the request) MUST NOT exceed the **RollupDownstreamServersMaxBatchSize** value returned by the USS in the preceding **GetRollupConfiguration** call.

Data processing:

The USS MUST process this message as follows:

1. The USS MAY use the clientTime value passed by the DSS to determine the difference between the USS system clock and DSS system clock, and adjust the other input time stamps to compensate for the difference.

   <40>

2. For each **DownstreamServerRollupInfo** structure in the request:

   1. Get the **ParentServerId** value and search the DSS Table for an entry with a **ServerID** matching this value. If no such entry is found, return a SOAP fault message and stop processing this request.

   2. Get the **ServerId** value and search the DSS Table for an entry with a **ServerID** matching this value.

      1. If no entry is found, create a new entry, initializing all values in the entry using values from the **DownstreamServerRollupInfo** structure.

If the **DownstreamServerRollupInfo** structure contains a **ParentServerId** value of "00000000-0000-0000-0000-000000000000", the **ParentServerId** value MUST instead use the **ServerID** value from the Server Configuration Table.

2. If an existing entry is found, and the **LastRollupTime** value from this entry in the table is less than or equal to the **LastRollupTime** value from the **DownstreamServerRollupInfo** structure, then do the following:

    1. Update the entry using the values from the **DownstreamServerRollupInfo** structure. If the **DownstreamServerRollupInfo** structure contains a **ParentServerId** value of "00000000-0000-0000-0000-000000000000", the **ParentServerID** value MUST instead use the **ServerID** value from the Server Configuration Table.

    2. For each **DownstreamServerRollupClientSummary** structure in the **ClientSummaries** array, search the Client Computer Activity Summaries Table for an entry with UpdateID, ServerID, and OS version values matching those from the **DownstreamServerRollupInfo** and **DownstreamServerRollupClientSummary** structures.

        1. If an existing entry is found, increment the **InstallSuccessCount** and **InstallFailureCount** values in the table entry by the corresponding value from the **DownstreamServerRollupClientSummary** structure.

        2. Else add a new entry to the table using the values from the **DownstreamServerRollupInfo** and **DownstreamServerRollupClientSummary** structure.

3. Else ignore this **DownstreamServerRollupInfo** structure.

Response:

If no errors occur during processing, the USS MUST return the success response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault indicating that an error occurred. This protocol does not define a mechanism for returning information on the cause of the failure.

3.1.4.13 RollupComputers

A DSS invokes the **RollupComputers** method to send information on client computers that get updates from it, or from any descendent update server, to the USS.

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

The SOAP operation is as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/RollupComputers" style="document" />
```

Request:

```
<s:element name="RollupComputers">
  <s:complexType>
    <s:sequence>
```

```
      <s:element minOccurs="0" maxOccurs="1" name="cookie" type="tns:C
ookie" />
      <s:element minOccurs="1" maxOccurs="1" name="clientTime" type="s
:dateTime" />
      <s:element minOccurs="0" maxOccurs="1" name="computers" type="tn
s:ArrayOfComputerRollupInfo" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: This field is reserved for future use. This field MUST be present, and the value MUST be set to the following:

Expiration (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999

EncryptedData = zero length array

clientTime: This field MUST be present and set to the current time on the DSS.

computers: This field MUST be present, but it may contain zero elements in the array. Each element in the array contains information on an update server. Its format is as follows:

```
<s:complexType name="ArrayOfComputerRollupInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="ComputerRollu
pInfo" nillable="true" type="tns:ComputerRollupInfo" />
  </s:sequence>
</s:complexType>
```

ArrayOfComputerRollupInfo: An array structure, each element of that contains information on one client computer. The format of each element is as follows:

```
<s:complexType name="ComputerRollupInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Details" type="tns:Co
mputerRollupDetails" />
  </s:sequence>
  <s:attribute name="ComputerId" type="s:string" />
  <s:attribute name="LastSyncTime" type="s:dateTime" use="required" />
  <s:attribute name="LastSyncResult" type="s:int" use="required" />
  <s:attribute name="LastReportedRebootTime" type="s:dateTime" use="re
quired" />
  <s:attribute name="LastReportedStatusTime" type="s:dateTime" use="re
quired" />
  <s:attribute name="LastInventoryTime" type="s:dateTime" use="require
d" />
  <s:attribute name="ParentServerId" type="s1:guid" use="required" />
</s:complexType>
```

Details: This field MUST be present if this DSS is reporting information on this client computer for the first time. This field MAY be present in subsequent calls. It contains detailed information on the client computer. Its format is as given below, following the description of the **ParentServerId** field.

<41>

ComputerId: This field MUST be present. It indicates the globally unique string that is used to identify the client computer.

LastSyncTime: This field MUST be present. It indicates the time when the client computer last contacted the update server to check for new updates. If the client computer has never contacted the update server to check for updates, this value MUST be set to the following:

Date (Year-Month-Date) = 1753-01-01

Time (Hours:Minutes:Seconds)= 00:00:00

LastSyncResult: This field MUST be present. It indicates the result of the client computer's last attempt to get updates from the update server. The valid values for this field are shown in the following table.

Value	Meaning
0	Unknown/client computer has never contacted the update server
1	Succeeded
2	Failed

LastReportedRebootTime: This field MUST be present. It indicates the time when the client computer last started, and then notified the update server that it has started (the time stamp indicates the time it started, not the time when the update server was notified). If the client computer has never notified the update server that it has started, this value MUST be set to the following:

Date (Year-Month-Date) = 1753-01-01

Time (Hours:Minutes:Seconds)= 00:00:00

LastReportedStatusTime: This field MUST be present. It indicates the time when the client computer last reported the status of updates on the client computer to the update server. If the client computer has never reported the status of updates, this value MUST be set to the following:

Date (Year-Month-Date) = 1753-01-01

Time (Hours:Minutes:Seconds)= 00:00:00

LastInventoryTime: This field MUST be present. It indicates the time when the client computer last reported software and hardware inventory information to the update server. If the client computer has never reported inventory information, this value MUST be set to the following:

Date (Year-Month-Date) = 1753-01-01

Time (Hours:Minutes:Seconds)= 00:00:00

ParentServerId: This field MUST be present. It indicates the GUID that identifies the update server that the client computer gets updates from.

```
<s:complexType name="ComputerRollupDetails">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="TargetGroupIdList" ty
```

```
pe="tns:ArrayOfGuid" />
    <s:element minOccurs="0" maxOccurs="1" name="RequestedTargetGroupN
ames" type="tns:ArrayOfString" />
  </s:sequence>
  <s:attribute name="IPAddress" type="s:string" />
  <s:attribute name="FullDomainName" type="s:string" />
  <s:attribute name="OSMajorVersion" type="s:int" use="required" />
  <s:attribute name="OSMinorVersion" type="s:int" use="required" />
  <s:attribute name="OSBuildNumber" type="s:int" use="required" />
  <s:attribute name="OSServicePackMajorNumber" type="s:int" use="requi
red" />
  <s:attribute name="OSServicePackMinorNumber" type="s:int" use="requi
red" />
  <s:attribute name="OSLocale" type="s:string" />
  <s:attribute name="ComputerMake" type="s:string" />
  <s:attribute name="ComputerModel" type="s:string" />
  <s:attribute name="BiosVersion" type="s:string" />
  <s:attribute name="BiosName" type="s:string" />
  <s:attribute name="BiosReleaseDate" type="s:dateTime" use="required"
 />
  <s:attribute name="ProcessorArchitecture" type="s:string" />
  <s:attribute name="SuiteMask" type="s:short" use="required" />
  <s:attribute name="OldProductType" type="s:unsignedByte" use="requir
ed" />
  <s:attribute name="NewProductType" type="s:int" use="required" />
  <s:attribute name="SystemMetrics" type="s:int" use="required" />
  <s:attribute name="ClientVersion" type="s:string" />
</s:complexType>
```

ComputerRollupDetails: A structure that contains detailed information on the client computer.

TargetGroupIdList: This field MUST be present, but it may contain zero elements in the array. It contains a list of GUIDs that identify the list of target groups that the client computer belongs to.

RequestedTargetGroupNames: This field MUST be present, but it may contain zero elements in the array. It lists the names of the target groups of which the client computer has been configured to be a member, regardless of whether or not the named target group exists. Individual strings in this array MUST NOT exceed 256 characters.

IPAddress: This field MUST be present. It indicates the client computer's IP address.

FullDomainName: This field MUST be present. It indicates the fully qualified DNS name of the client computer.

OSMajorVersion: This field MUST be present. It indicates the operating system major version number of the client computer.

OSMinorVersion: This field MUST be present. It indicates the operating system minor version number of the client computer.

OSBuildNumber: This field MUST be present. It indicates the operating system build number of the client computer.

OSServicePackMajorNumber: This field MUST be present. It indicates the operating system service pack major version number of the client computer.

OSServicePackMinorNumber: This field MUST be present. It indicates the operating system service pack minor version number of the client computer.

OSLocale: This field MUST be present. It indicates the operating system locale of the client computer. Details are specified in [MS-LCID].

ComputerMake: This field MUST be present. It indicates the name of the client computer's manufacturer. The string MUST contain no more than 64 characters.

ComputerModel: This field MUST be present. It indicates the name of the client computer's model. The string MUST contain no more than 64 characters.

BiosVersion: This field MUST be present. It indicates the version number of the client computer's BIOS. The string MUST contain no more than 64 characters.

BiosName: This field MUST be present. It indicates the name of the client computer's BIOS. The string MUST contain no more than 64 characters.

BiosReleaseDate: This field MUST be present. It indicates the release date of the client computer's BIOS.

ProcessorArchitecture: This field MUST be present. It is a string indicating the CPU architecture that the operating system on the client computer is built for. The string MUST contain no more than 50 characters.

SuiteMask: This field MUST be present. It contains additional integer data to be used to identify the operating system used by the client computer.

OldProductType: This field MUST be present. It contains additional integer data to be used to identify the operating system used by the client computer.

NewProductType: This field MUST be present. It contains additional integer data to be used to identify the operating system used by the client computer.

SystemMetrics: This field MUST be present. It contains additional integer data to be used to identify the operating system used by the client computer.

ClientVersion: This field MUST be present. It indicates the version number of the client software on the client computer that is communicating with the update server. The string MUST contain no more than 20 characters.<42>

Response:

```
<s:element name="RollupComputersResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="RollupComputersResu
lt" type="tns:ArrayOfChangedComputer" />
    </s:sequence>
  </s:complexType>
</s:element>
```

RollupComputersResult: On successful execution of this operation, this object MUST be returned. Its format is as follows:

```
<s:complexType name="ArrayOfChangedComputer">
```

```
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="unbounded" name="ChangedComput
er" nillable="true" type="tns:ChangedComputer" />
    </s:sequence>
  </s:complexType>
```

ArrayOfChangedComputer: An array structure, each element of which defines a client computer and a ComputerChangeType value indicating an action that the DSS must take on this client computer. Details on how the DSS MUST react are specified in section 3.2.4.5. The format of each element is as follows:

```
<s:complexType name="ChangedComputer">
  <s:attribute name="ComputerId" type="s:string" />
  <s:attribute name="Change" type="tns:ComputerChangeType" use="requir
ed" />
</s:complexType>
```

ComputerId: This field MUST be present, and it MUST match a **ComputerId** from the request sent by the DSS. It indicates the client computer that the instruction MUST be applied to.

ComputerChangeType: This field MUST be present. It specifies the action the DSS MUST take for the specified client computer. The valid values for this field are as follows:

```
<s:simpleType name="ComputerChangeType">
  <s:restriction base="s:string">
    <s:enumeration value="Deleted" />
    <s:enumeration value="NewParent" />
  </s:restriction>
</s:simpleType>
```

Request validation:

The USS validates inputs as given below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS.

Input	Validation conditions
computers	MUST be present, and the total number of ComputerRollupInfo structures in the request MUST NOT exceed the **RollupComputersMaxBatchSize** value returned by the USS in the preceding **GetRollupConfiguration** call.

Data processing:

The USS MUST process this message as follows:

1. If the **DoDetailedRollup** value in the Server Configuration Table is FALSE, return a SOAP fault message to the DSS with an <ErrorCode> of InternalServerError and stop processing the request.
2. For each ComputerRollupInfo structure in the request, get the **ParentServerId** value and search the DSS Table for an entry with a matching **ServerID**. If no entry is found, return a SOAP fault

message to the DSS with an <ErrorCode> of InternalServerError and stop processing the request.

3. The USS MAY use the clientTime value passed by the DSS to determine the difference between the USS system clock and DSS system clock, and adjust the other input time stamps to compensate for the difference.

   <43>

4. For each ComputerRollupInfo structure in the request, get the **ComputerId** value and search the Client Computers Table for an entry with a matching **ComputerId**.

   1. If no entry is found, create a new entry, initializing all values using values from the ComputerRollupInfo structure.

   2. If an existing entry is found, and the **LastSyncTime** value from this entry in the table is less than or equal to the **LastSyncTime** value from the ComputerRollupInfo structure, replace the values in the entry using the values from the ComputerRollupInfo structure.

   3. Else ignore this ComputerRollupInfo structure.

5. The USS MUST compose a RollupComputersResponse message in response, as follows:

   1. The **RollupComputersResult** array is initialized to an empty array.

   2. For each ComputerRollupInfo structure in the request, if the Details field does not exist and one of the following is true:

      1. A new entry was added to the Client Computers Table in step 3a for this ComputerRollupInfo structure.

      2. An existing entry in the Client Computers Table was updated in step 3b for this ComputerRollupInfo structure, and the **ParentServerID** value was changed to a new value.

      The USS MUST add an entry to the **RollupComputersResult** array with the following:

      1. **ComputerId**: Set to the **ComputerId** value from the ComputerRollupInfo structure.

      2. **Change**: Set to NewParent.

      This notifies the caller that it MUST populate the Details field the next time it reports information about this client computer to this USS.

   3. For each ComputerRollupInfo structure in the request, the USS MAY add an entry to the **RollupComputersResult** array with the following:

      1. **ComputerId**: Set to the **ComputerId** value from the ComputerRollupInfo structure.

      2. **Change**: Set to Deleted.

      This notifies the caller that the USS is no longer interested in receiving information about this client computer. <44>

      The same **ComputerId** MUST NOT appear more than once in the **RollupComputersResult** array. If the **ComputerId** already exists in the array as a result of step 5b, the same **ComputerId** MUST NOT be added in step 5c.

Response:

If no errors occur during processing, the USS MUST return the success response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault indicating that an error occurred. This protocol does not define a mechanism for returning information on the cause of the failure. The DSS MUST abort the protocol when a SOAP fault is received.

3.1.4.14 GetOutOfSyncComputers

A DSS invokes the **GetOutOfSyncComputers** method to determine what data needs to be reported to the USS.

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

The SOAP operation is as follows:

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

Request:

```
<s:element name="GetOutOfSyncComputers">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie" type="tns:C
ookie" />
      <s:element minOccurs="1" maxOccurs="1" name="parentServerId" typ
e="s1:guid" />
      <s:element minOccurs="0" maxOccurs="1" name="lastRollupNumbers"
type="tns:ArrayOfComputerLastRollupNumber" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: This field is reserved for future use. This field MUST be present, and the value MUST be set to the following:

Expiration (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999

EncryptedData = zero length array

parentServerId: This field MUST be present. It indicates the GUID that identifies the DSS making the call.

lastRollupNumbers: This field MUST be present, but it may contain zero elements in the array. Each element in the array contains information that is used to determine what information DSS must report for each client computer. Its format is as follows:

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

```
ollupNumber" nillable="true" type="tns:ComputerLastRollupNumber" />
  </s:sequence>
</s:complexType>
```

ArrayOfComputerLastRollupNumber: An array structure, each element of which contains information is used to determine what information DSS must report for a client computer. The format of each element is as follows:

```
<s:complexType name="ComputerLastRollupNumber">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="ComputerId" type="s:s
tring" />
    <s:element minOccurs="1" maxOccurs="1" name="RollupNumber" type="s
:int" />
  </s:sequence>
</s:complexType>
```

ComputerId: This field MUST be present. It indicates the globally unique string that is used to identify the client computer.

RollupNumber: This field MUST be present. It indicates the **RollupNumber** value that was used when the DSS last sent information about this client computer to the USS using **RollupComputerStatus**. If the DSS has never sent information about this client computer to the USS using **RollupComputerStatus**, the value MUST be 0. This value MUST be 0 or greater.

Response:

```
<s:element name="GetOutOfSyncComputersResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetOutOfSyncCompute
rsResult" type="tns:ArrayOfString" />
    </s:sequence>
  </s:complexType>
</s:element>
```

GetOutOfSyncComputersResult: This field MUST be present, but it may contain zero elements in the array. Every string in the array MUST match a **ComputerId** from the request sent by the DSS. This array indicates the list of client computers for which the DSS must send all available information in a following call to **RollupComputerStatus**.

Request validation:

The USS validates inputs as given below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS.

Input	Validation conditions
lastRollupNumbers	MUST be present, and the total number of **ComputerLastRollupNumber** structures in the request MUST NOT exceed the **GetOutOfSyncComputersMaxBatchSize** value returned by the USS in the preceding **GetRollupConfiguration** call.

Data processing:

The USS MUST process this message as follows:

1. If the **DoDetailedRollup** value in the Server Configuration Table is FALSE, return a SOAP fault message to the DSS with an <ErrorCode> of InternalServerError and stop processing the request.

2. If the DSS Table does not contain an entry with **ServerID** equal to the parentServerId value from the request, compose and return a GetOutOfSyncComputersResponse message containing an empty **GetOutOfSyncComputersResult** array.

3. Else, the USS MUST compose a GetOutOfSyncComputersResponse message in response, as follows:

    1. The **GetOutOfSyncComputersResult** array is initialized to an empty array.

    2. Use the entries in the DSS Table to construct a list containing the **ServerID** values of all DSSs synchronize with the update server identified by parentServerId, or with one of its descendant DSSs.

    3. For each **ComputerLastRollupNumber** structure in the request, get the **ComputerId** value and search the client computers table for an entry with a matching **ComputerId** value.

        1. If an entry is found, the **ParentServerID** value form the entry appears on the list constructed in step 3b, and the **LastReceivedRollupNumber** value from the entry does not match the **RollupNumber** value from **ComputerLastRollupNumber** structure, then add the **ComputerId** to the **GetOutOfSyncComputersResult** array.

        2. Else ignore this **ComputerLastRollupNumber** structure.

Response:

If no errors occur during processing, the USS MUST return the success response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault indicating that an error occurred. This protocol does not define a mechanism for returning information on the cause of the failure. The DSS MUST abort the protocol when a SOAP fault is received.

3.1.4.15 RollupComputerStatus

A DSS invokes the **RollupComputerStatus** method to send information on the status of updates on each of the client computers that get updates from it (or from any descendent update server) to the USS.

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

The SOAP operation is as follows:

```
<soap:operation soapAction="http://www.microsoft.com/SoftwareDistribut
ion/RollupComputerStatus" style="document" />
```

Request:

```
<s:element name="RollupComputerStatus">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="cookie" type="tns:C
ookie" />
      <s:element minOccurs="1" maxOccurs="1" name="clientTime" type="s
:dateTime" />
      <s:element minOccurs="1" maxOccurs="1" name="parentServerId" typ
e="s1:guid" />
      <s:element minOccurs="0" maxOccurs="1" name="computers" type="tn
s:ArrayOfComputerStatusRollupInfo" />
    </s:sequence>
  </s:complexType>
</s:element>
```

cookie: This field is reserved for future use. This field MUST be present, and the value MUST be set to the following:

Expiration (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999

EncryptedData = zero length array

clientTime: This field MUST be present and set to the current time on the DSS.

parentServerId: This field MUST be present. It indicates the GUID that identifies the DSS making the call.

computers: This field MUST be present, but it may contain zero elements in the array. Each element in the array contains information about the status of updates on a client computer. Its format is as follows:

```
<s:complexType name="ArrayOfComputerStatusRollupInfo">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="ComputerStatu
sRollupInfo" nillable="true" type="tns:ComputerStatusRollupInfo" />
  </s:sequence>
</s:complexType>
```

ArrayOfComputerStatusRollupInfo: An array structure, each element of which contains information on the status of updates on a client computer.

```
<s:complexType name="ComputerStatusRollupInfo">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="InstanceId" type="s1:
guid" />
    <s:element minOccurs="0" maxOccurs="1" name="ComputerId" type="s:s
tring" />
    <s:element minOccurs="1" maxOccurs="1" name="EffectiveLastDetectio
nTime" type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="RollupNumber" type="s
:int" />
    <s:element minOccurs="1" maxOccurs="1" name="IsFullRollup" type="s
:boolean" />
    <s:element minOccurs="0" maxOccurs="1" name="UpdateStatus" type="t
ns:ArrayOfComputerStatusRollupUpdateStatus" />
```

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

InstanceId: This field MUST be present. It MUST be a new GUID.

ComputerId: This field MUST be present. It indicates the globally unique string that is used to identify the client computer.

EffectiveLastDetectionTime: This field MUST be present. It indicates the time when the newest update (that the client computer has reported status for) was made available on the update server (this is the time when the update was made available, not when the status was reported). If the client computer has never reported the status of updates on the client computer, or if the update server did not have any updates available at the time the client computer reported status, this value MUST be set to the following:

Date (Year-Month-Date) = 1753-01-01

Time (Hours:Minutes:Seconds)= 00:00:00

RollupNumber: This field MUST be present. This value MUST be greater than the **RollupNumber** that was previously used for this client computer in the last **RollupComputerStatus** call.

If the update server has not previously sent information about this client computer to the USS using **RollupComputerStatus**, the value MUST be 1.

This value must be 1 or greater.

IsFullRollup: This field MUST be present. Set to TRUE if this **ComputerRollupInfo** structure describes the status of all updates on the client computer. Set to FALSE if this **ComputerRollupInfo** structure contains information only for updates for which there have been changes on the client computer. An update is considered to have changed if the status of the update on the client computer has changed at least once since the last time information on this client was reported. Note that if the status has changed to a new value, and then changed back again, it is considered to have changed.

UpdateStatus: This field MUST be present, but it may contain zero elements in the array. Each element in the array describes the status of a single update on the client computer. Its format is as follows:

```
<s:complexType name="ArrayOfComputerStatusRollupUpdateStatus">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="ComputerStatu
sRollupUpdateStatus" nillable="true" type="tns:ComputerStatusRollupUpd
ateStatus" />
  </s:sequence>
</s:complexType>
```

ArrayOfComputerStatusRollupUpdateStatus: An array structure, each element of which describes the status of a single update on the client computer.

```
<s:complexType name="ComputerStatusRollupUpdateStatus">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="UpdateId" type="s1:gu
id" />
```

```
    <s:element minOccurs="1" maxOccurs="1" name="SummarizationState" t
ype="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="LastChangeTime" type=
"s:dateTime" />
  </s:sequence>
</s:complexType>
```

UpdateId: This field MUST be present. It identifies the update for which the status on the client computer is being described.

SummarizationState: This field MUST be present. It indicates the status of the update on the client computer. The valid values are shown in the following table.

Value	Meaning
0	Not applicable
2	Not installed
3	Downloaded but not yet installed
4	Installed
5	Failed
6	Installed, but requires starting

LastChangeTime: This field MUST be present. It indicates the time when the status of the update on the client computer was changed to its current value.

Response:

```
<s:element name="RollupComputerStatusResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="RollupComputerStatu
sResult" type="s:boolean" />
    </s:sequence>
  </s:complexType>
</s:element>
```

RollupComputerStatusResult: This field MUST be present. The value MUST be TRUE if the USS successfully received and processed the request. If the USS was too busy or otherwise was unable to process the request, the value MUST be FALSE.

Request validation:

The USS validates inputs as given below. If any of the inputs are invalid, the USS MUST return a SOAP fault message to the DSS

Input	Validation conditions
computers	MUST be present, and the total number of **ComputerStatusRollupInfo** structures in the request MUST NOT exceed the **RollupComputerStatusMaxBatchSize** value returned by

Input	Validation conditions
	the USS in the preceding **GetRollupConfiguration** call.

Data processing:

The USS MUST process this message as follows:

1. If the **DoDetailedRollup** value in the Server Configuration Table is FALSE, return a SOAP fault message to the DSS with an <ErrorCode> of InternalServerError, and stop processing the request.

2. If the USS is under heavy load, the USS MAY return RollupComputerStatusResponse message with **RollupComputerStatusResult** set to FALSE, and stop processing the request. This notifies the DSS that it SHOULD wait, resend the request at a later time.

   <45>

3. The USS MAY use the clientTime value passed by the DSS to determine the difference between the USS system clock and DSS system clock, and adjust the other input time stamps to compensate for the difference.

   <46>

4. For each ComputerStatusRollupInfo structure in the request, get the **ComputerId** value and search the Client Computers Table for an entry with a matching **ComputerId** value.

   1. If an entry is found, do the following:

      1. Set the **LastReceivedRollupNumber** value in the table entry to the **RollupNumber** value from the **ComputerStatusRollupInfo** structure.

      2. Set the **EffectiveLastDetectionTime** value in the table entry to the **EffectiveLastDetectionTime** value form the **ComputerStatusRollupInfo** structure.

      3. If the **IsFullRollup** value from the **ComputerStatusRollupInfo** structure is TRUE, remove all entries from the Update Status Table with a **ComputerId** value matching the **ComputerId** value from the **ComputerStatusRollupInfo** structure.

      4. For each **ComputerStatusRollupUpdateStatus** structure in the **UpdateStatus** array from the **ComputerStatusRollupInfo** structure, search the Update Status Table for an entry with **ComputerId** and UpdateID values matching the **ComputerId** value from the **ComputerStatusRollupInfo** structure, and the UpdateId value from the **ComputerStatusRollupUpdateStatus** structure, respectively.

         1. If an entry is found, compare the **LastChangeTime** value from the entry with the **LastChangeTime** value from the **ComputerStatusRollupUpdateStatus** structure.

            1. If the **LastChangeTime** value from the table entry is greater, ignore this **ComputerStatusRollupUpdateStatus** structure.

            2. Else update the State and LastChangeTime values in the table entry using the **SummarizationState** and **LastChangeTime** values, respectively, from the **ComputerStatusRollupUpdateStatus** structure.

         2. If no entry is found, add a new entry to the Update Status table, initializing the values as follows:

1. Set **ComputerId** to the **ComputerId** value from the **ComputerStatusRollupInfo** structure.

2. Set UpdateID, State, and **LastChangeTime** values to the **UpdateId**, **SummarizationState**, and **LastChangeTime** values from the **ComputerStatusRollupUpdateStatus** structure, respectively.

2. Else ignore this **ComputerStatusRollupInfo** structure.

Response:

If no errors occur during processing, the USS MUST return the success response to the DSS.

If an error occurs during processing, the USS MUST return a SOAP fault indicating that an error occurred. This protocol does not define a mechanism for returning information on the cause of the failure. The DSS MUST abort the protocol when a SOAP fault is received.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

3.2 DSS Details

The DSS Authorization Web Service on the USS is used to authorize the release of updates to DSSs.

On failure, all methods specified in this section MUST return a SOAP fault (section 2.2.2) message that contains error information.

3.2.1 Abstract Data Model

The data model used by the DSS is the same as USS.

3.2.2 Timers

None.

3.2.3 Initialization

A DSS MUST have some implementation-specific way of learning the DNS name or IP address and the TCP/IP port of the USS that it is configured to synchronize from (for example, manual configuration or specified as part of the Start Synchronization trigger). No other actions are taken until an event is triggered by a higher layer, as described below.<47>

Start Synchronization trigger: The DSS MUST provide at least one way to trigger the start of the synchronization process that uses this protocol. The DSS implementation MAY use timer events to trigger the protocol initiation.<48>

On receipt of the Start Synchronization trigger, the DSS MUST start the Authorization step given in section 3.2.4.1.

Cancel Synchronization trigger: The DSS implementation MAY provide a higher layer with the capability to cancel a synchronization that is in progress. This protocol does not define when such a

Cancel request must be honored by the DSS and how to effect the cancellation. If the steps and message sequencing given in section 3.2.4 are followed, the protocol is designed to ensure that the DSS data store integrity is maintained after every step in the synchronization until processing is completed at the DSS.<49>

Start Reporting Data Synchronization trigger: The DSS MAY provide an additional way to trigger the start of the reporting process that uses this protocol but skips the Deployments Synchronization and Content Synchronization steps.<50>

A DSS MUST have some implementation-specific way of learning its own DNS name. It MUST also have an implementation-specific way of creating a globally unique ID to identify itself.

3.2.4 Message Processing Events and Sequencing Rules

The DSS MUST perform the following sequential steps (phases) every time it synchronizes with the USS.

1. Authorization

   The DSS provides credentials to the USS to obtain a Cookie from the USS during this step.

2. Metadata Synchronization

   The DSS uses the Cookie from the Authorization phase and gets information about new updates from the USS during this step.

3. Deployments Synchronization

   This step is applicable only if the DSS is configured as a replica server. The DSS gets the list of target groups and deployments from the USS during this step.

4. Content Synchronization

   The DSS downloads the content files associated with the updates from the USS during this step. This is performed asynchronously. This step is applicable only if the **CatalogOnlySync** flag in the Server Configuration for this DSS is set to FALSE.

5. Reporting Data Synchronization

   During this step, the DSS sends (to the USS) information on the update servers that synchronize from it and on the client computers that get updates from it.

   This step is performed only if both the USS and DSS support version 1.3 of the protocol.

The following sequence diagram shows the details of the message exchange. The content download step is carried out asynchronously. For simplicity, it is shown in sequence.

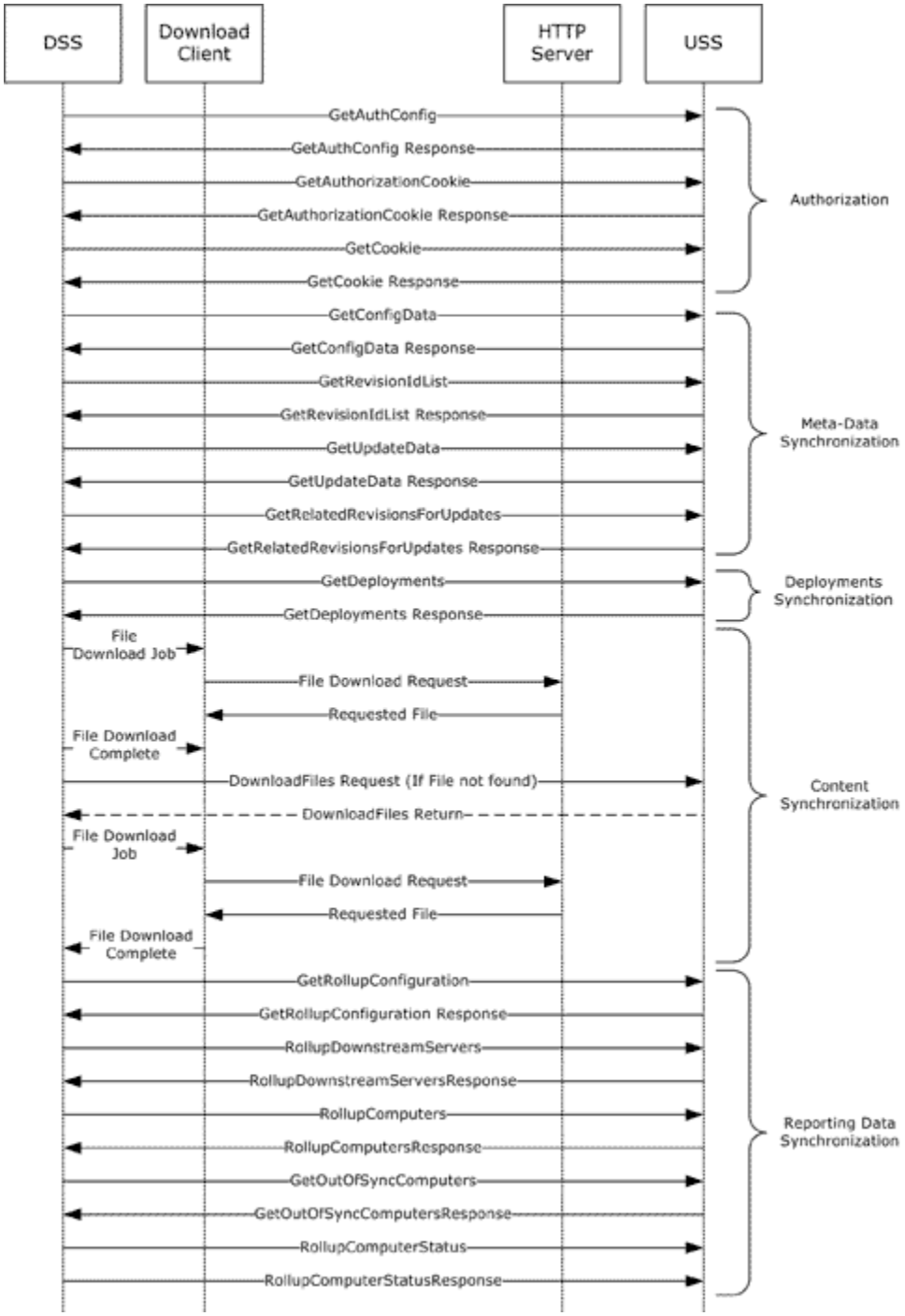

Figure 3: Message exchange

If the DSS receives a SOAP fault in response to any Web service call, it MUST attempt to parse the SOAP fault using the format specified in section 2.2.2, and then extract the **ErrorCode** value. If an **ErrorCode** value is available, the DSS MUST react to the error, as specified in section 2.2.2.3. If no ErrorCode value is available, the DSS MUST abort the protocol.

3.2.4.1 Authorization

The DSS provides credentials to the USS to obtain a Cookie from the USS during this step.

The DSS MUST obtain a Cookie from the USS as follows:

1. Compose and transmit a **GetAuthConfig** message to the USS.
2. Receive a GetAuthConfigResponse from the USS, extract the AuthPlugIn.ServiceUrl and save it in the to the Parent USS State Table.
3. Compose a **GetAuthorizationCookie** message by doing the following:
    1. Copy the **Account Name** field of the Server Configuration Table to the **GetAuthorizationCookie** request.
    2. Copy the **Account GUID** field of the Server Configuration Table to the **GetAuthorizationCookie** request.
4. Transmit the **GetAuthorizationCookie** request to the USS and receive a response from the USS.
5. Save the AuthorizationCookie received from **GetAuthorizationCookie** response into the Parent USS State.
6. Compose a **GetCookie** message as follows:
    1. Add an entry into the **authCookies** array of the request message by copying the AuthorizationCookie from the Parent USS State data store.
    2. Initialize the **oldCookie** element of the request message from the Last Cookie stored in the USS Parent State data store.
    3. Initialize the protocol version to the version being implemented by the DSS.
7. Transmit the **GetCookie** request to the USS and receive a response from the USS.
8. Save the Cookie returned in the **GetCookie** response in the Last Cookie field of the USS Parent State data store.

The DSS MUST treat the **CookieData** field of the AuthorizationCookie and the **EncryptedData** field of the Cookie as an opaque object. It is to be interpreted by the USS only. The **Expiration** field of the Cookie indicates whether it is expired. The DSS MUST NOT use an expired Cookie in any method calls in the following sections.

3.2.4.2 Metadata Synchronization

The DSS first synchronizes configuration settings, supported languages, categories, classifications, and detectoids during this step. It then synchronizes the update revisions information from the USS. Each request sent during this processing step uses the Cookie returned by **GetCookie** as an input parameter to various operations.

The DSS MUST synchronize metadata from the USS as follows:

1. Sends a **GetConfigData** request to get configuration settings and languages supported by the USS. The request message is composed as follows:
    1. Last Cookie from the Parent USS State is copied to the request message.

2. Last **ConfigAnchor** from the Parent USS State is copied to the request message.

2. Store the following information in the returned response in the Parent USS State for future use:

   - **CatalogOnlySync**
   - **LazySync**
   - **ServerHostsPsfFiles**
   - **MaxNumberOfUpdatesPerRequest**
   - **NewConfigAnchor**
   - **LanguageUpdateList**

3. Send a **GetRevisionIdList** request to the USS with the following filter:

   - **GetConfig**: Set to TRUE to get Categories, Update Classifications, Detectoids.
   - **Get63LanguageOnly**: Omit if the DSS implements version 1.1 of the protocol. Set to TRUE if the DSS implements version 1.2 or later of the protocol (as specified in section 1.7) but does not intend to support more than 63 languages. Otherwise, set to FALSE.

4. Store the Anchor returned by this operation response in the **Last SyncAnchor** field of the USS Parent State for future use within the protocol.

5. Compose a **GetUpdateData** request by copying the Revision IDs returned in the **GetRevisionIdList** response into the updateIDs element of the **GetUpdateData** request.

   The number of items requested in this operation MUST NOT exceed **MaxNumberOfUpdatesPerRequest**, as specified in **GetConfigData** (section 3.1.4.4).

6. Send the **GetUpdateData** request to the USS to obtain the metadata related to any new categories and detectoids using the **GetUpdateData** operation. If the metadata is received in the **XmlUpdateBlobCompressed** field of the response, it MUST be decompressed before storing it in the data model, as specified in **GetUpdateData** (section 3.1.4.6).

7. On successful completion of this operation, store the revision IDs and the metadata in the Categories, Update Classifications, and Detectoid Tables. The DSS MUST use the following properties in the XML metadata to detect if a revision is a category, update classification, or a detectoid, as specified in section 3.1.1:

   - Detectoid: UpdateType = "Detectoid"
   - Update Classification: UpdateType = "Category" and CategoryType="UpdateClassification"
   - Category: UpdateType = "Category" and CategoryType="UpdateClassification"

8. Send a **GetRevisionIdList** request to the USS to get the software/driver update revisions IDs. The input filter is used, as specified in the following list:

   - **GetConfig**: Set to false.
   - **Get63LanguageOnly**: Set to true if the DSS implements version 1.2 of the protocol, as specified in section 1.7, and intends to support more that 63 languages.

9. Store the Anchor returned by this operation response in the **Last SyncAnchor** field of the USS Parent State for future use within the protocol.

10. Compose a **GetUpdateData** request by copying the revision IDs returned in the **GetRevisionIdList** response into the updateIDs element of the **GetUpdateData** request.

    The number of items requested in this operation MUST NOT exceed **MaxNumberOfUpdatesPerRequest**, as specified in **GetConfigData** (section 3.1.4.4).

11. Send the **GetUpdateData** request to the USS to obtain the metadata related to the new update revisions returned in the **GetRevisionIdList**. If the metadata is received in the **XmlUpdateBlobCompressed** field of the response, it MUST be decompressed before storing it in the data model, as specified in section 3.1.4.6). Store the metadata in the Revision Table.

    The number of items requested in this operation MUST NOT exceed **MaxNumberOfUpdatesPerRequest**, as specified in **GetConfigData** (section 3.1.4.4).

12. From each metadata for an update revision, extract the following elements (as specified in section 3.1.1.1) for each file associated with this update revision and store the extracted data in the Revision Table:

    - FileName
    - FileDigest
    - PatchingType

13. From each metadata for an update revision, extract the **EulaID** (as specified in section 3.1.1.1) and store it in the EULAs Table.

14. After all update revision metadata is received, continue with the Deployments Synchronization step if the Replication Mode in DSS's Server Configuration is set to "Replica". Otherwise, go to the Content Synchronization step as given in section 3.2.4.4.

3.2.4.3 Deployments Synchronization

This step is applicable only if the DSS is configured as a replica server. The DSS gets the list of target groups and deployments from the USS during this step.

The DSS MUST synchronize the target groups and deployments from the USS as follows:

1. Compose and send a **GetDeployments** request to the USS that is initialized as follows:
    1. Last Cookie from the Parent USS State is copied to the request message.
    2. Last Deployment Anchor from the Parent USS State is copied to the request message.
    3. Last Sync Anchor from the Parent USS State is copied to the request message.
2. Store the Anchor in the returned response in the Parent USS State as Last Deployment Anchor.
3. Update the Target Groups Table using the Groups array returned in the response:
    1. Update the target group entry if it already exists in the table.
    2. Add a new entry into the target group Table if it does not exist.
    3. Delete the target group entry in the table if it does not exist in the Groups returned. All deployments in the Deployments Table that refer to the deleted target group MUST also be deleted.

4. For each entry in the **DeadDeployments** array in the response, locate the entry in the Deployments Table and set the Dead flag to TRUE.

5. For each entry in the **Deployments** array in the response:

    1. Update the Deployment entry if it already exists in the Deployments Table.

    2. Add the Deployment entry if it does not exist in the Deployments Table.

6. For each entry in the **HiddenUpdates** array of the response locate the update in the Revision Table and set the Hidden flag to TRUE.

7. Set the Accepted flag to TRUE in the EULAs Table for every entry found in the **AcceptedEulas** array in the response.

8. Add a new entry to the Synchronization History Table with the LastSyncTime value set to the current time.

3.2.4.4 Content Synchronization

The DSS downloads the content files associated with the updates from the USS during this step. The protocol does not define when the file download is started. It is considered to be implementation-specific. However, the following rules MUST be followed by the DSS to determine the need to download a file.

- **CatalogOnlySync** flag in the Server Configuration for this DSS is set to FALSE.
- The file does not already exist in the Content Store.
- Either the **LazySync** flag in the Server Configuration for this DSS is set to FALSE.

    OR

    If the **LazySync** flag is set to TRUE, the update revision associated with this file has an associated Deployment in the Deployment Table whose Action is set to "Install".

    OR

    A **DownloadFiles** request is received by this Update Server for this file.

- If the file's PatchingType is "Express and" the ServerHostsPsfFiles flag in the Server Configuration for this DSS is set to TRUE.

    <51>

The DSS MUST synchronize content for each file that it needs to download as follows:

1. If the **CatalogOnlySync** flag of the Parent USS State is set to FALSE, construct the URL as follows and download the file using the USS Content Download service:

    1. Use the USS Download Content service URL, as specified in section 2.1, and fill the server and server port, using values stored in the Parent USS Configuration data store.

    2. Set the <folder name> of the URL to the last two characters of the **FileDigest** that is represented in Base64.

    3. Set the <content file name> to the name of the file as given in the Revisions Table.

2. If the **CatalogOnlySync** flag of the Parent USS State is set to TRUE, download the file using the **MUUrl** for the file.

3. When downloading from the USS, if a "File Not Found" HTTP error is received, request the USS to download the file by sending the **FileDigest** of the file in the **DownloadFiles** (section 3.1.4.9) request. An implementation MAY choose to batch multiple such failed requests within a single method call for **DownloadFiles**.<52>

4. For any other error during the HTTP download of the file, the recovery action is implementation-specific and not defined by the protocol.

   <53>

3.2.4.5 Reporting Data Synchronization

The DSS reports information on its descendant DSSs and their client computers to the USS during this step. This step is performed only if both the USS and the DSS support version 1.3 of the protocol.

This step MAY run before or in parallel with the Content Synchronization step. Also, this step MAY be triggered and run on its own without running the previous steps in the protocol.

<54>

The DSS MUST send information on its descendant DSSs and their client computers to the USS as follows:

1. The DSS sends a **GetRollupConfiguration** request to obtain information on the USS's report rollup requirements. The request message is composed as follows:

   - **cookie**: Set to a **Cookie** structure with the following fields:

     1. **Expiration** (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999
     2. **EncryptedData** = zero length array

   The response is stored for use in the steps of the protocol that follow.

2. The DSS reports information on itself and any descendent DSSs to the USS as follows:

   1. The DSS constructs a **DownstreamServerInfo** structure representing itself, as follows:

      1. **ServerId**: The self-generated GUID identifying the DSS.
      2. **FullDomainName**: The fully qualified DNS name of the DSS.
      3. **LastRollupTime**: The current time.
      4. **ParentServerId**: "00000000-0000-0000-0000-000000000000".
      5. **IsReplica**: True if the DSS is configured as a replica server; otherwise, false.
      6. **LastSyncTime**: The **LastSyncTime** value from the Server State Table. If the value in the table is NULL, use 1753-01-01 00:00:00 (Year-Month-Date Hours:Minutes:Seconds).
      7. **ServerSummary**: Set to a **DownstreamServerRollupServerSummary** structure describing the status of updates on the DSS, and of client computers that get updates from the DSS. The member fields are set as follows:

1. **UpdateCount**: The number of unique UpdateIdentity GUID values in the Revisions Table (the Revision Number portion is ignored).

2. **DeclinedUpdateCount**: The number of unique UpdateIdentity GUID values in the Revisions Table, for which all entries with that UpdateIdentity GUID have the Hidden element set to TRUE.

3. **ApprovedUpdateCount**: The number of unique UpdateIdentity GUID values in the Revisions Table, for which:

    1. At least one entry with that UpdateIdentity GUID has the Hidden element set to FALSE.

    2. There is at least one entry in the Deployments Table with a matching UpdateIdentity GUID and an Action value of 0.

4. **NotApprovedUpdateCount**: The number of unique UpdateIdentity GUID values in the Revisions Table, for which:

    1. At least one entry with that UpdateIdentity GUID has the Hidden element set to FALSE.

    2. There is no entry in the Deployments Table with a matching UpdateIdentity GUID and an Action value of 0.

5. **ApprovedUpdateCount**: The number of entries in the Revisions Table for which:

    1. At least one entry with that UpdateIdentity GUID has the Hidden element set to FALSE.

    2. There exists at least one other entry in the Revisions Table with the same UpdateIdentity GUID but a larger UpdateIdentity Revision Number.

    3. There exists an entry in the Deployments Table with the same UpdateIdentity value and an Action value of 0.

6. **CriticalOrSecurityUpdatesNotApprovedForInstallCount**: The number of unique UpdateIdentity GUID values in the Revisions Table, for which:

    1. At least one entry with that UpdateIdentity GUID has the Hidden element set to FALSE.

    2. There is no entry in the Deployments Table with a matching UpdateIdentity GUID and an Action value of 0.

    3. The update that this entry represents is intended to repair a security issue on the client computers, or is otherwise considered to be critical to the operation of the client computers. How this determination is made is implementation dependent.

        <55>

7. **WsusInfrastructureUpdatesNotApprovedForInstallCount**: The number of unique UpdateIdentity GUID values in the Revisions Table, for which:

    1. At least one entry with that UpdateIdentity GUID has the Hidden element set to FALSE.

2. There is no entry in the Deployments Table with a matching UpdateIdentity GUID and an Action value of 0.

3. The update that this entry represents is required for the client computers to continue to get updates from this DSS. How this determination is made is implementation dependent.

   <56>

8. **CustomComputerTargetGroupCount**: The number of entries in the Target Groups Table for which the **IsBuiltin** value is FALSE.

9. **UpdatesWithClientErrorsCount**: The number of updates available on the DSS that at least one client computer has attempted and failed to install.

10. **UpdatesWithServerErrorsCount**: The number of updates available on the DSS for which the DSS has failed to download content.

11. **UpdatesNeedingFilesCount**: The number of updates for which the DSS must download content but has not completed the download.

12. **UpdatesNeededByComputersCount**: The number of updates available on the DSS to which at least one client computer is applicable but not yet installed on.

13. **UpdatesUpToDateCount**: The number of updates available on the DSS known to be installed on all client computers to which it is applicable.

14. **ComputerTargetCount**: The number of client computers that gets updates from this DSS.

15. **ComputerTargetsNeedingUpdatesCount**: The number of client computers that gets updates from this DSS on which at least one update is known to be applicable but not yet installed.

16. **ComputerTargetsWithUpdateErrorsCount**: The number of client computers that gets updates from this DSS and that has tried and failed to install at least one update.

17. **ComputersUpToDateCount**: The number of client computers that is known to have successfully installed all updates available on this DSS that are applicable to it.

The values for steps 2.1.9-2.1.17 are all computed from the values stored in the Revisions Table, Deployments Table, and Update Status Table.

8. **ClientSummary**: Set to an array of **DownstreamServerRollupClientSummary** structures. The array is populated by constructing one **DownstreamServerRollupClientSummary** structure for each entry in the Client Computer Activity Summary Table with a ServerID value equal to the GUID identifying this DSS.

   The fields of the structure are populated by copying the corresponding values from the table entry.

2. The DSS constructs a list of **DownstreamServerInfo** structures, one for each entry in the DSS Table. The list MUST be ordered in a way such that if one DSS synchronizes with another, the structure corresponding to the parent DSS appears before that of the child DSS.

   The fields in the **DownstreamServerInfo** structures are initialized using values from the corresponding DSS Table entry.

The **ClientSummary** array is populated by searching the Client Computer Activity Summary Table for entries with a ServerID value matching the ServerID from the DSS Table entry, and then constructing a **DownstreamServerRollupClientSummary** for each such entry by copying the values from the table entry.

3. The DSS appends the **DownstreamServerInfo** structure from step 1 to the end of the list constructed in step 2.

4. If the number of **DownstreamServerRollupClientSummary** structures in any one member of the resulting list is greater than the **RollupDownstreamServersMaxBatchSize** value received from the USS in step 1, split the offending **DownstreamServerInfo** structure into two or more structures as follows:

    1. Split the **ClientSummary** array of the original **DownstreamServerInfo** structure into two or more sub-arrays, each containing at least one member, such that the number of **DownstreamServerRollupClientSummary** structures in any one sub-array does not exceed the **RollupDownstreamServersMaxBatchSize** value from step 1.

    2. For each sub-array, construct a new **DownstreamServerInfo** structure. All fields in this structure, excluding the **ClientSummary** array, are set to the value of the corresponding field in the original **DownstreamServerInfo** structure. The **ClientSummary** array is set to the sub-array created in step 4.1.

    3. Remove the original **DownstreamServerInfo** structure from the list, and insert the new **DownstreamServerInfo** structures created in step 3 in its place.

5. The DSS sends the **DownstreamServerInfo** structures to the USS using **RollupDownstreamServers** messages. Each request is composed as follows:

    1. **cookie**: Set to a **Cookie** structure with the following fields:

        1. **Expiration** (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999

        2. **EncryptedData** = zero length array

    2. **clientTime**: Set to the current time.

    3. **downstreamServers**: Populate the array using one or more **DownstreamServerInfo** structures from the list.

    An implementation MAY choose to send the **DownstreamServerInfo** structures one at a time, or it MAY choose to send them in batches. In either case, ordering MUST be preserved.

    Example: If structure A appears before structure B in the list, the request containing structure B MUST NOT be sent before the request containing structure A is sent. They MAY be sent in the same request; however, in this example, structure A MUST appear before structure B in the **downstreamServers** array.

    Furthermore, the number of **DownstreamServerRollupClientSummary** structures in one request, summed across all **DownstreamServerInfo** structures in the request, MUST NOT exceed the **RollupDownstreamServersMaxBatchSize** value.

    <57>

    For each **RollupDownstreamServers** request sent, the USS will respond with a **RollupDownstreamServers** message. Upon receiving this response, the DSS removes from

the Client Activity Summary Table all rows corresponding to the **ServerID** values that were sent as part of the request.

3. If the **DoDetailedRollup** value from the **GetRollupConfigurationResponse** structure (received from the USS in step 1) is FALSE, stop. The following steps in the protocol MUST NOT be run. If the value is TRUE, continue.

4. The DSS reports information on client computers that get updates from it or from any of its descendant DSSs as follows:

    1. For each entry in the Client Computers Table, the DSS constructs a **ComputerRollupInfo** structure, populating the fields as follows:

        1. **ComputerId**, **LastSyncTime**, **LastSyncResult**, **LastReportedRebootTime**, **LastReportedStatusTime**, **LastInventoryTime**, and **ParentServerId**: Use the corresponding value from the table entry. If a time stamp value in the table is NULL, use 1753-01-01 00:00:00 (Year-Month-Date Hours:Minutes:Seconds) instead.

        2. The Details field MUST NOT be added if the HasDetailsChanged element of the table entry is FALSE.

            If the HasDetailsChanged element is TRUE, the Details field MUST be populated as follows:

            TargetGroupIdList, **RequestedTargetGroupNames**, **IPAddress**, **FullDomainName**, **OSMajorVersion**, **OSMinorVersion**, **OSBuildNumber**, **OSServicePackMajorNumber**, **OSServicePackMinorNumber**, **OSLocale**, **ComputerMake**, **ComputerModel**, **BiosVersion**, **BiosName**, **BiosReleaseDate**, **ProcessorArchitecture**, **SuiteMask**, **OldProductType**, **NewProductType**, **SystemMetrics**, and **ClientVersion**: Use the corresponding value form the table entry.

    2. The DSS sends the **ComputerRollupInfo** structures to the USS using **RollupComputers** messages. Each request is composed as follows:

        1. **cookie**: Set to a **Cookie** structure with the following fields:

            1. **Expiration** (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999

            2. **EncryptedData** = zero length array

        2. **clientTime**: Set to the current time.

        3. **computers**: Populate the array using one or more of the **ComputerRollupInfo** structures.

            An implementation MAY choose to send the **ComputerRollupInfo** structures one at a time, or it MAY choose to send them in batches. If the implementation chooses to send them in batches, the number of **ComputerRollupInfo** structures sent in a single request MUST NOT exceed the **RollupComputersMaxBatchSize** value obtained from the **GetRollupConfigurationResponse**.

            <58>

        For each **RollupComputers** message sent, the USS will respond with a RollupComputersResponse message, with the **RollupComputersResult** field set to an array of zero or more **ChangedComputer** structures. The DSS MUST process each **ChangedComputer** structure in each response as follows:

1. If the value of the Change field is deleted, get the **ComputerId** value from the **ChangedComputer** structure, find the entry in the Client Computers Table with the matching **ComputerId** value, and remove it from the table.

   If the entry is not found in the table, do nothing.

2. If the value of the Change field is NewParent, get the **ComputerId** value from the **ChangedComputer** structure, find the entry in the Client Computers Table with the matching ComputerId value, and set the **HasDetailsChanged** value to TRUE.

   If the entry is not found in the table, do nothing.

3. Repeat step 4.1, but only for entries in the Client Computers Table having a **HasDetailsChanged** value of TRUE. Repeat step 4.2, using the resulting list of **ComputerRollupInfo** structures; repeat step 4.3 for the resulting responses.

5. The DSS reports the status of each update on each client computer to the USS as follows:

   1. For each entry in the Client Computers Table, the DSS constructs a **ComputerLastRollupNumber** structure, populating the fields as follows:

      1. **ComputerId**: Use the **ComputerId** value from the table entry.

      2. **RollupNumber**: Use the LastSentStatusRollupNumber value from the table entry.

   2. The DSS sends the **ComputerLastRollupNumber** structures to the USS using **GetOutOfSyncComputers** messages. Each request is composed as follows:

      1. **cookie**: Set to a **Cookie** structure with the following fields:

         1. **Expiration** (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999

         2. **EncryptedData** = zero length array

      2. **parentServerId**: Set to the **ServerID** value from the Server Configuration Table.

      3. **lastRollupNumbers**: Populate the array using one or more of the **ComputerLastRollupNumber** structures.

         An implementation MAY choose to send the **ComputerLastRollupNumber** structures one at a time, or it MAY choose to send them in batches. If the implementation chooses to send them in batches, the number of **ComputerLastRollupNumber** structures sent in a single request MUST NOT exceed the **GetOutOfSyncComputersMaxBatchSize** value obtained from the **GetRollupConfigurationResponse**.

         <59>

      For each **GetOutOfSyncComputers** message sent, the USS will respond with a GetOutOfSyncComputersResponse message with the **GetOutOfSyncComputersResult** field set to an array of zero or more strings, each identifying a client computer.

      For each string, the DSS must search the Client Computers Table for an entry with the matching ComputerId value. If an entry is found, set the **LastStatusRollupTime** to NULL. If no entry is found, do nothing.

   3. For each entry in the Client Computers Table, the DSS constructs a **ComputerStatusRollupInfo** structure, populating the fields as follows:

1. **InstanceId**: Set to a new, randomly generated GUID.
2. **ComputerId**: Set to the **ComputerId** value from the table entry.
3. **EffectiveLastDetectionTime**: Search the Synchronization History Table for the row with the largest **SynchronizationTime** value that is less than the **EffectiveLastDetectionTime** value from the Client Computers Table row. If such a row is found, then use the **SynchronizationTime** value from that row; if no row is found, use 1753-01-01 00:00:00 (Year-Month-Date Hours:Minutes:Seconds).
4. **RollupNumber**: Use the **LastSentStatusRollupNumber** value from the table entry, plus 1.
5. **IsFullRollup**: If the LastStatusRollupTime value is NULL, set this to TRUE; else set this to FALSE.
6. **UpdateStatus**: If the **LastStatusRollupTime** value is NULL, find all entries in the Update Status Table with a **ComputerId** value matching the **ComputerId** value from the Client Computer Table entry. If the **LastStatusRollupTime** value is not NULL, find all matching Update Status Table entries having a **LastChangeTime** value greater than the **LastStatusRollupTime**.

   For each entry, construct a **ComputerStatusRollupUpdateStatus** structure as follows. This field should be set to an array containing the resulting **ComputerStatusRollupUpdateStatus** structures.

   1. **UpdateId**: Set to the **UpdateID** value from the Update Status Table entry.
   2. **SummarizationState**: Set to the **State** value from the Update Status Table entry.
   3. **LastChangeTime**: Set to the **LastChangeTime** value from the Update Status Table entry.

4. The DSS sends the **ComputerStatusRollupInfo** structures to the USS using RollupComputerStatus messages. Each request is composed as follows:

   1. **cookie**: Set to a **Cookie** structure with the following fields:

      1. **Expiration** (Year-Month-Date Hours:Minutes:Seconds) = 9999-12-31 23:59:59.9999999
      2. **EncryptedData** = zero length array

   2. **clientTime**: Set to the current time.
   3. **parentServerId**: Set to the **ServerID** value from the Server Configuration Table.
   4. **computers**: Populate the array using one or more of the **ComputerStatusRollupInfo** structures.

      An implementation MAY choose to send the **ComputerStatusRollupInfo** structures one at a time, or it MAY choose to send them in batches. If the implementation chooses to send them in batches, the number of **ComputerStatusRollupInfo** structures sent in a single request MUST NOT exceed the **RollupComputerStatusMaxBatchSize** value obtained from the **GetRollupConfigurationResponse**.

      <60>

For each RollupComputerStatus message sent, the USS will respond with a RollupComputerStatusResponse message, each containing a Boolean **RollupComputerStatusResult** value.

If the **RollupComputerStatusResult** value is FALSE, this indicates that the USS was too busy to accept the request, and the DSS MUST wait at least 1 minute before sending another request. The DSS MAY resend the same request (with the **clientTime** value updated to reflect the current time) after waiting.<61>

If the **RollupComputerStatusResult** value is TRUE, this indicates that the USS successfully received the message. For each **ComputerStatusRollupInfo** structure that was sent as part of the request, the DSS MUST find the corresponding entry in the Client Computers Table and update the entry as follows:

1. **LastStatusRollupTime**: Search the **UpdateStatus** array in the **ComputerStatusRollupInfo** structure for the entry with the largest **LastChangeTime**, and use this **LastChangeTime**.

2. **LastSentStatusRollupNumber**: Set to the current **LastSentStatusRollupNumber**, plus 1.

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

4 Protocol Examples

The following samples show sequences of messages exchanged in various steps within the protocol. All protocol messages are wrapped inside the <soap:Envelope> and <soap:Body> elements. To simplify the examples, only the first two messages, GetAuthConfig Request and GetAuthConfig Response, in sample 1 and the SOAP fault in sample 3 illustrate the use of these SOAP elements.

Sample 1: Authorization

The following sample message sequence shows how the Authorization step is performed between a DSS and a USS. All requests flow from the DSS to the USS, and all responses flow from the USS to the DSS. This sample shows three request-response exchanges.

- **GetAuthConfig**
- **GetAuthorizationCookie**
- **GetCookie**

```
<!-- AUTHORIZATION PHASE: GetAuthConfig() 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">
  <soap:Body>
    <GetAuthConfig
               xmlns="http://www.microsoft.com/SoftwareDistribution" />
  </soap:Body>
</soap:Envelope>
<!-- AUTHORIZATION PHASE: GetAuthConfig() 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>
    <GetAuthConfigResponse
               xmlns="http://www.microsoft.com/SoftwareDistribution">
      <GetAuthConfigResult>
        <LastChange>2006-06-07T20:41:50.883Z</LastChange>
        <AuthInfo>
          <AuthPlugInInfo>
            <PlugInID>DssTargeting</PlugInID>
      <ServiceUrl>DssAuthWebService/DssAuthWebService.asmx</ServiceUrl>
            <Parameter />
          </AuthPlugInInfo>
        </AuthInfo>
      </GetAuthConfigResult>
    </GetAuthConfigResponse>
  </soap:Body>
</soap:Envelope>

<!-- AUTHORIZATION PHASE: GetAuthorizationCookie() Request  -->
    <GetAuthorizationCookie
xmlns="http://www.microsoft.com/SoftwareDistribution/Server/DssAuthWebService">
      <accountName>HemantTest.redmond.microsoft.com</accountName>
      <accountGuid>ADB2FE48-0B2E-451e-8FC8-44B29845B0C6</accountGuid>
    </GetAuthorizationCookie>
<!-- AUTHORIZATION PHASE: GetAuthorizationCookie() Response - Success  -->
    <GetAuthorizationCookieResponse xmlns="http://www.microsoft.com/SoftwareDistrib
ution/Server/DssAuthWebService">
```

```
      <GetAuthorizationCookieResult>
        <PlugInId>DssTargeting</PlugInId>
        <CookieData>4VEGgCdh/cuG4BSwhzLDomn2BAcYKl45Po5CmuBbOlRXfwhmBopYbLUHLWP
          <!--............................... -->
          <!--.....CookieData BLOB TRIMMED..... -->
          <!--............................... -->
EHsBFG</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">
  <soap:Body>
    <GetCookie xmlns="http://www.microsoft.com/SoftwareDistribution">
      <authCookies>
        <AuthorizationCookie>
          <PlugInId>DssTargeting</PlugInId>
          <CookieData>4VEGgCdh/cuG4BSwhzLDomn2BAcYKl45Po5CmuBbOlRXfwhmBopYbLUHLWP
          <!--............................... -->
          <!--.....CookieData BLOB TRIMMED..... -->
          <!--............................... -->
EHsBFG</CookieData>
        </AuthorizationCookie>
      </authCookies>
      <protocolVersion>1.2</protocolVersion>
    </GetCookie>
<!-- AUTHORIZATION PHASE: GetCookie() Response - Success  -->
    <GetCookieResponse
              xmlns="http://www.microsoft.com/SoftwareDistribution">
      <GetCookieResult>
        <Expiration>2006-06-08T19:32:23.7190987Z</Expiration>
        <EncryptedData>edzLnPhI/Rjr6CBMlcP+fuCRNBKREOtAw9yEMe6EF3oXXvU/N67oNV/AnL
5eN3ZAA+jOoJzdcdJmc9za5M3B0jqBSnZsWch8waYEVNqd+eLx1EMBCK2v9lb+oXtMGw05PgQJAHtZPRk
5h4bCN4hxaCaI7JJaN7VWiWBQLh9HAKoBEm7WpOyJgnr7acYoEf64KOrB+aRc0QFuNnLsNYPNQKmxl7R+
cGkS</EncryptedData>
      </GetCookieResult>
    </GetCookieResponse>
```

Sample 2: Metadata and Deployments Synchronization

The following sample message sequences show how this protocol is used during the Metadata Synchronization and Deployments Synchronization phases. This is not a complete conversation between a DSS and a USS. Specifically, the array fields in the content of the message are trimmed to reduce the space required for this illustration. All requests flow from the DSS to the USS, and all responses flow from the USS to the DSS.

This sample shows four request-response exchanges.

- **GetConfigData**
- **GetRevisionIdList**
- **GetUpdateData**
- **GetDeployments**

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!-- META-DATA SYNC PHASE: GetConfigData() Request  -->
    <GetConfigData
              xmlns="http://www.microsoft.com/SoftwareDistribution">
      <cookie>
        <Expiration>2006-06-08T19:32:23.7190987Z</Expiration>
<EncryptedData>edzLnPhI/Rjr6CBMlcP+fuCRNBKREOtAw9yEMe6EF3oXXvU/N67oNV/AnL5eN3ZAA+
jOoJzdcdJmc9za5M3B0jqBSnZsWch8waYEVNqd+eLx1EMBCK2v9lb+oXtMGw05PgQJAHtZPRk5h4bCN4h
xaCaI7JJaN7VWiWBQLh9HAKoBEm7WpOyJgnr7acYoEf64KOrB+aRc0QFuNnLsNYPNQKmxl7R+cGkS</En
cryptedData>
      </cookie>
    </GetConfigData>

<!-- META-DATA SYNC PHASE: GetConfigData() Response  -->
    <GetConfigDataResponse
               xmlns="http://www.microsoft.com/SoftwareDistribution">
      <GetConfigDataResult>
        <CatalogOnlySync>true</CatalogOnlySync>
        <LazySync>true</LazySync>
        <ServerHostsPsfFiles>false</ServerHostsPsfFiles>
        <MaxNumberOfUpdatesPerRequest>100
        </MaxNumberOfUpdatesPerRequest>
        <NewConfigAnchor>14030,2006-06-08 15:38:16.022
        </NewConfigAnchor>
        <LanguageUpdateList>
          <ServerSyncLanguageData>
            <LanguageID>0</LanguageID>
            <ShortLanguage>all</ShortLanguage>
            <LongLanguage>all</LongLanguage>
            <Enabled>true</Enabled>
          </ServerSyncLanguageData>
          <ServerSyncLanguageData>
            <LanguageID>1033</LanguageID>
            <ShortLanguage>en</ShortLanguage>
            <LongLanguage>English</LongLanguage>
            <Enabled>true</Enabled>
          </ServerSyncLanguageData>
        </LanguageUpdateList>
      </GetConfigDataResult>
    </GetConfigDataResponse>

<!-- META-DATA SYNC PHASE: GetRevisionIdList() Request  -->
    <GetRevisionIdList
              xmlns="http://www.microsoft.com/SoftwareDistribution">
      <cookie>
        <Expiration>2006-05-26T22:59:25.1296674Z</Expiration>
        <EncryptedData>c7jG7GX4Czeq+uO8twkF3uNyVfDX6G2bXwIUKk114C+BOl80eXz9YToko+
KAMpO50h16kmztwMUn4i7bJ7AQw64t05XtgfmF9F42JG1mADwLmRUSAIxnWEpKJe4fmdUzHPKjUWbKBPi
2UijibE2jZ4y177B3vafHo3RtM7NM2lMi/VWXxHwlNKvMYvzOrOvXPNuDJtTM3AOHGzW2MlZ4daT7TUsd
5m6g</EncryptedData>
      </cookie>
      <filter>
        <GetConfig>true</GetConfig>
        <Get63LanguageOnly>false</Get63LanguageOnly>
      </filter>
    </GetRevisionIdList>

<!-- META-DATA SYNC PHASE: GetRevisionIdList() Response  -->
    <GetRevisionIdListResponse
               xmlns="http://www.microsoft.com/SoftwareDistribution">
      <GetRevisionIdListResult>
```

```
        <Anchor>4742,2006-05-26 18:59:26.192</Anchor>
        <NewRevisions>
          <UpdateIdentity>
            <UpdateID>0287014e-6772-4b4d-81f7-42063ac26e8a</UpdateID>
            <RevisionNumber>100</RevisionNumber>
          </UpdateIdentity>
          <!--........................ -->
          <!--.....ARRAY TRIMMED..... -->
          <!--........................ -->
          <UpdateIdentity>
            <UpdateID>ff2a96fb-cc89-4e5e-83b2-7d48aa179840</UpdateID>
            <RevisionNumber>101</RevisionNumber>
          </UpdateIdentity>
        </NewRevisions>
      </GetRevisionIdListResult>
    </GetRevisionIdListResponse>

<!-- META-DATA SYNC PHASE: GetUpdateData() Request  -->
    <GetUpdateData
              xmlns="http://www.microsoft.com/SoftwareDistribution">
      <cookie>
        <Expiration>2006-05-27T22:42:23.0632331Z</Expiration>
        <EncryptedData>XXRQXVUzH9NX5Qvos3IL4F31UmlUgdVSVFm450d+O9A32AyPyTo66g4VlMA
48PRBtRsvajGUYA+CDNeF/ylHxuy48MUWiTJK68AJHO1V+Z6sfBDDnCTVlFc8z7UUQHguLLTXx0B4hkHhm
5FSRMH03ei7mDJnhGstnEhu2hjASXU8uvespc7yt5BtoPCamiNEJQR2WgCImaw3S96AubSSKxDv9yEAChj
O</EncryptedData>
      </cookie>
      <updateIds>
        <UpdateIdentity>
          <UpdateID>0287014e-6772-4b4d-81f7-42063ac26e8a</UpdateID>
          <RevisionNumber>100</RevisionNumber>
        </UpdateIdentity>
        <!--........................ -->
        <!--.....ARRAY TRIMMED..... -->
        <!--........................ -->
        <UpdateIdentity>
          <UpdateID>36b1c48e-c741-4047-9b09-d38698ed34b3</UpdateID>
          <RevisionNumber>101</RevisionNumber>
        </UpdateIdentity>
      </updateIds>
    </GetUpdateData>
<!-- META-DATA SYNC PHASE: GetUpdateData() Response  -->
    <GetUpdateDataResponse
              xmlns="http://www.microsoft.com/SoftwareDistribution">
      <GetUpdateDataResult>
        <updates>
          <ServerSyncUpdateData>
            <Id>
              <UpdateID>0fa1201d-4330-4fa8-8ae9-b877473b6441</UpdateID>
              <RevisionNumber>1</RevisionNumber>
            </Id>
            <XmlUpdateBlobCompressed>TVNDRgAAAABfCwAAAAAACwAAAAAAAAAwEBAAEAAAAAA
AAAQQAAAAEAAxU2NgAAAAAAAAAAAAAAAAAAAYmxvYgBLKmjgFgs2NluAgI0DIGRjQRAAAgIAR
          <!--................................ -->
          <!--.....COMPRESSED BLOB TRIMMED..... -->
          <!--................................ -->
COEqHYXEGOORfjDFecM5DTHmuLzJUxYNb5WUkHxosE1dCv1cNwrigJVvwU8EsqzX2z0UOqAxw==</XmlUp
dateBlobCompressed>
          </ServerSyncUpdateData>
          <!--........................ -->
          <!--.....ARRAY TRIMMED..... -->
          <!--........................ -->
          <ServerSyncUpdateData>
```

```
            <Id>
              <UpdateID>17e993cd-cf5a-4276-9944-6af62ff7139c</UpdateID>
              <RevisionNumber>100</RevisionNumber>
            </Id>
            <XmlUpdateBlob>
              <upd:Update xmlns:pub="http://schemas.microsoft.com/msus/2002/12/Pu
blishing" xmlns:mar="http://schemas.microsoft.com/msus/2002/12/MsiApplicabilityRu
les" xmlns:upd="http://schemas.microsoft.com/msus/2002/12/Update">
                <upd:UpdateIdentity UpdateID="17e993cd-cf5a-4276-9944-6af62ff7139
c" RevisionNumber="100" />
                <upd:Properties DefaultPropertiesLanguage="en" UpdateType="Detect
oid" ExplicitlyDeployable="false" IsPublic="false" DetectoidType="Application Loc
ale" PublicationState="Published" CreationDate="2006-05-22T19:10:10.172Z" Publish
erID="dd79a08b-c3c2-4e05-9862-ee90f9585e33"></upd:Properties>
                <upd:LocalizedPropertiesCollection>
                  <upd:LocalizedProperties>
                    <upd:Language>en</upd:Language>
                    <upd:Title>SQL 2005 English ia64</upd:Title>
                  </upd:LocalizedProperties>
                </upd:LocalizedPropertiesCollection>
                <upd:Relationships>
                  <upd:Prerequisites>
                    <upd:UpdateIdentity UpdateID="60916385-7546-4e9b-836e-79d65e51
7bab" />
                  </upd:Prerequisites>
                </upd:Relationships>
                <upd:ApplicabilityRules>
                  <upd:IsInstalled>
                    <mar:MsiProductInstalled ProductCode="{3c4a397d-22b2-4ab0-849f
-f5e12672caca}" ExcludeVersionMax="true" VersionMin="9.00.1399.06" Language="1033"
 xmlns:mar="http://schemas.microsoft.com/msus/2002/12/MsiApplicabilityRules" />
                  </upd:IsInstalled>
                </upd:ApplicabilityRules>
              </upd:Update>
            </XmlUpdateBlob>
          </ServerSyncUpdateData>
        </updates>
        <fileUrls />
      </GetUpdateDataResult>
    </GetUpdateDataResponse>
<!-- DEPLOYMENTS SYNC PHASE: GetDeployments() Request  -->
    <GetDeployments
                xmlns="http://www.microsoft.com/SoftwareDistribution">
      <cookie>
        <Expiration>2006-05-26T22:59:25.1296674Z</Expiration>
        <EncryptedData>c7jG7GX4Czeq+uO8twkF3uNyVfDX6G2bXwIUKk114C+BOl80eXz9YToko+
KAMpO50h16kmztwMUn4i7bJ7AQw64t05Xtgfmf9F42JG1mADwLmRUSAIxnWEpKJe4fmdUzHPKjUWbKBPi
2UijibE2jZ4yl77B3vafHo3RtM7NM2lMi/VWXxHwlNKvMYvzOrOvXPNuDJtTM3AOHGzW2MlZ4daT7TUsd
5m6g</EncryptedData>
      </cookie>
      <syncAnchor>4742,2006-05-26 18:59:26.192</syncAnchor>
    </GetDeployments>
<!-- DEPLOYMENTS SYNC PHASE: GetDeployments() Response  -->
    <GetDeploymentsResponse
                xmlns="http://www.microsoft.com/SoftwareDistribution">
      <GetDeploymentsResult>
        <Anchor>4742,2006-05-26 18:59:26.192</Anchor>
        <Groups>
          <ServerSyncTargetGroup>
            <TargetGroupID>a0a08746-4dbe-4a37-9adf-9e7652c0b421</TargetGroupID>
            <Name>All Computers</Name>
            <IsBuiltin>true</IsBuiltin>
          </ServerSyncTargetGroup>
          <ServerSyncTargetGroup>
            <TargetGroupID>b73ca6ed-5727-47f3-84de-015e03f6a88a</TargetGroupID>
```

```
        <Name>Unassigned Computers</Name>
        <IsBuiltin>true</IsBuiltin>
      </ServerSyncTargetGroup>
    </Groups>
    <Deployments>
      <ServerSyncDeployment>
        <UpdateId>4b62aa32-52f6-4c15-b739-1c8e696eb33e</UpdateId>
        <RevisionNumber>100</RevisionNumber>
        <Action>0</Action>
        <AdminName>WUS Server</AdminName>
        <Deadline>9999-12-31T23:59:59.9999999</Deadline>
        <IsAssigned>true</IsAssigned>
        <GoLiveTime>2006-05-24T15:37:08.887</GoLiveTime>
        <DeploymentGuid>c2f2ca62-5413-419a-bdee-275a23087758</DeploymentGuid>
        <TargetGroupId>b73ca6ed-5727-47f3-84de-015e03f6a88a</TargetGroupId>
        <DownloadPriority>1</DownloadPriority>
      </ServerSyncDeployment>
      <!--..................... -->
      <!--.....ARRAY TRIMMED..... -->
      <!--..................... -->
      <ServerSyncDeployment>
        <UpdateId>52bdd083-4169-4094-8eea-6230c7c1520d</UpdateId>
        <RevisionNumber>102</RevisionNumber>
        <Action>0</Action>
        <AdminName>WUS Server</AdminName>
        <Deadline>9999-12-31T23:59:59.9999999</Deadline>
        <IsAssigned>true</IsAssigned>
        <GoLiveTime>2006-05-24T15:59:33.063</GoLiveTime>
        <DeploymentGuid>10a2030d-0ac1-434a-9c37-732fc36f9838</DeploymentGuid>
        <TargetGroupId>b73ca6ed-5727-47f3-84de-015e03f6a88a</TargetGroupId>
        <DownloadPriority>1</DownloadPriority>
      </ServerSyncDeployment>
    </Deployments>
    <DeadDeployments />
    <HiddenUpdates>
      <guid>9a9bafe2-fbfc-4b0b-8479-49a2bb08cd5a</guid>
      <!--..................... -->
      <!--.....ARRAY TRIMMED..... -->
      <!--..................... -->
      <guid>6c9d9217-0e1d-434a-a938-d9918703c361</guid>
    </HiddenUpdates>
    <AcceptedEulas />
  </GetDeploymentsResult>
</GetDeploymentsResponse>
```

Sample 3: Reporting Data Synchronization

The following sample message sequences show how this protocol is used during the Reporting Data Synchronization phase. This is not a complete conversation between a DSS and a USS. Specifically, the array fields in the content of the message are trimmed to reduce the space required for this illustration. Furthermore, the sample only shows one message for each of the following: **RollupDownstreamServers**, **RollupComputers**, **GetOutOfSyncComputers**, and **RollupComputerStatus**; a complete conversation can consist of any number of calls being made to each of these methods.

All requests flow from the DSS to the USS, and all responses flow from the USS to the DSS.

This sample shows five request-response exchanges.

1. **GetRollupConfiguration**
2. **RollupDownstreamServers**

3. **RollupComputers**

4. **GetOutOfSyncComputers**

5. **RollupComputerStatus**

```
<!-- REPORTING DATA SYNC PHASE: GetRollupConfiguration request -->
<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://schemas.xm
lsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xm
lns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetRollupConfiguration xmlns="http://www.microsoft.com/SoftwareDistribution">
      <cookie>
        <Expiration>9999-12-31T23:59:59.9999999</Expiration>
        <EncryptedData />
      </cookie>
    </GetRollupConfiguration>
  </soap:Body>
</soap:Envelope>

<!-- REPORTING DATA SYNC PHASE: GetRollupConfiguration response -->
<?xml version="1.0" encoding="utf-8"?>
<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/XMLS
chema">
  <soap:Body>
    <GetRollupConfigurationResponse xmlns="http://www.microsoft.com/SoftwareDistr
ibution">
      <GetRollupConfigurationResult>
        <DoDetailedRollup>true</DoDetailedRollup>
        <RollupResetGuid>3277f62c-fa08-468f-873f-2f9867bcf9e5</RollupResetGuid>
        <ServerId>059ab167-8a19-4bab-befc-daaab5b1d82b</ServerId>
        <RollupDownstreamServersMaxBatchSize>5000</RollupDownstreamServersMaxBat
chSize>
        <RollupComputersMaxBatchSize>1500</RollupComputersMaxBatchSize>
        <GetOutOfSyncComputersMaxBatchSize>20000</GetOutOfSyncComputersMaxBatchS
ize>
        <RollupComputerStatusMaxBatchSize>500</RollupComputerStatusMaxBatchSize>
      </GetRollupConfigurationResult>
    </GetRollupConfigurationResponse>
  </soap:Body>
</soap:Envelope>

<!-- REPORTING DATA SYNC PHASE: RollupDownstreamServers request -->
<?xml version="1.0" encoding="utf-8"?>
<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/XMLS
chema">
  <soap:Body>
    <RollupDownstreamServers xmlns="http://www.microsoft.com/SoftwareDistribution
">
      <cookie>
        <Expiration>9999-12-31T23:59:59.9999999</Expiration>
        <EncryptedData />
      </cookie>
      <clientTime>2007-03-06T03:28:02.5507362Z</clientTime>
      <downstreamServers>
        <DownstreamServerRollupInfo>
          <ServerId>050cee4a-aaff-470c-8fc3-7a94cb3a4293</ServerId>
          <FullDomainName>yasufs64.ntdev.corp.microsoft.com</FullDomainName>
          <LastSyncTime>0001-01-01T00:00:00</LastSyncTime>
          <ParentServerId>00000000-0000-0000-0000-000000000000</ParentServerId>
          <Version>3.0.6000.354</Version>
          <IsReplica>true</IsReplica>
          <LastRollupTime>2007-03-06T03:28:01.894503Z</LastRollupTime>
```

```
          <ServerSummary>
            <UpdateCount>1002</UpdateCount>
            <DeclinedUpdateCount>29</DeclinedUpdateCount>
            <ApprovedUpdateCount>2</ApprovedUpdateCount>
            <NotApprovedUpdateCount>971</NotApprovedUpdateCount>
            <UpdatesWithStaleUpdateApprovalsCount>0</UpdatesWithStaleUpdateAppro
valsCount>
            <ExpiredUpdateCount>1</ExpiredUpdateCount>
            <CriticalOrSecurityUpdatesNotApprovedForInstallCount>953</CriticalOr
SecurityUpdatesNotApprovedForInstallCount>
            <WsusInfrastructureUpdatesNotApprovedForInstallCount>3</WsusInfrastr
uctureUpdatesNotApprovedForInstallCount>
            <UpdatesWithClientErrorsCount>0</UpdatesWithClientErrorsCount>
            <UpdatesWithServerErrorsCount>0</UpdatesWithServerErrorsCount>
            <UpdatesNeedingFilesCount>0</UpdatesNeedingFilesCount>
            <UpdatesNeededByComputersCount>1</UpdatesNeededByComputersCount>
            <UpdatesUpToDateCount>972</UpdatesUpToDateCount>
            <CustomComputerTargetGroupCount>0</CustomComputerTargetGroupCount>
            <ComputerTargetCount>1</ComputerTargetCount>
            <ComputerTargetsNeedingUpdatesCount>1</ComputerTargetsNeedingUpdate
sCount>
            <ComputerTargetsWithUpdateErrorsCount>0</ComputerTargetsWithUpdateE
rrorsCount>
            <ComputersUpToDateCount>0</ComputersUpToDateCount>
          </ServerSummary>
          <ClientSummaries>
            <DownstreamServerRollupClientSummary>
              <OSMajorVersion>5</OSMajorVersion>
              <OSMinorVersion>2</OSMinorVersion>
              <OSBuildNumber>3790</OSBuildNumber>
              <OSServicePackMajorNumber>1</OSServicePackMajorNumber>
              <OSServicePackMinorNumber>0</OSServicePackMinorNumber>
              <OSLocale>en-US</OSLocale>
              <SuiteMask>272</SuiteMask>
              <OldProductType>3</OldProductType>
              <NewProductType>0</NewProductType>
              <SystemMetrics>0</SystemMetrics>
              <ProcessorArchitecture>AMD64</ProcessorArchitecture>
              <Count>1</Count>
              <ActivitySummaries>
                <DownstreamServerRollupClientActivitySummary>
                  <UpdateId>63ee225f-897d-42ae-841e-4a383c92f350</UpdateId>
                  <RevisionNumber>101</RevisionNumber>
                  <InstallSuccessCount>1</InstallSuccessCount>
                  <InstallFailureCount>0</InstallFailureCount>
                </DownstreamServerRollupClientActivitySummary>
              </ActivitySummaries>
            </DownstreamServerRollupClientSummary>
          </ClientSummaries>
        </DownstreamServerRollupInfo>
      </downstreamServers>
    </RollupDownstreamServers>
  </soap:Body>
</soap:Envelope>

<!-- REPORTING DATA SYNC PHASE: RollupDownstreamServers response -->
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xs
i="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/200
1/XMLSchema">
  <soap:Body>
    <RollupDownstreamServersResponse xmlns="http://www.microsoft.com/SoftwareD
istribution" />
  </soap:Body>
</soap:Envelope>
```

```
<!-- REPORTING DATA SYNC PHASE: RollupComputers request -->
<?xml version="1.0" encoding="utf-8"?>
<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>
    <RollupComputers xmlns="http://www.microsoft.com/SoftwareDistribution">
      <cookie>
        <Expiration>9999-12-31T23:59:59.9999999</Expiration>
        <EncryptedData />
      </cookie>
      <clientTime>2007-03-06T03:28:02.9726004Z</clientTime>
      <computers>
        <ComputerRollupInfo ComputerId="d07a4d51-7cb2-46d1-ad86-91195cb4325a" L
astSyncTime="2007-03-06T03:22:28.137Z" LastSyncResult="1" LastReportedRebootTim
e="0001-01-01T00:00:00" LastReportedStatusTime="2007-03-06T03:27:59.377Z" LastI
nventoryTime="0001-01-01T00:00:00" ParentServerId="050cee4a-aaff-470c-8fc3-7a94
cb3a4293">
          <Details IPAddress="127.0.0.1" FullDomainName="yasufs64.ntdev.corp.mi
crosoft.com" OSMajorVersion="5" OSMinorVersion="2" OSBuildNumber="3790" OSServi
cePackMajorNumber="1" OSServicePackMinorNumber="0" OSLocale="en-US" ComputerMak
e="Hewlett-Packard" ComputerModel="HP Compaq dc7600 Convertible Minitower" Bios
Version="786D1 v01.03" BiosName="Default System BIOS" BiosReleaseDate="2005-05-
18T00:00:00Z" ProcessorArchitecture="AMD64" SuiteMask="272" OldProductType="3"
NewProductType="0" SystemMetrics="0" ClientVersion="7.0.6000.354">
            <TargetGroupIdList>
              <guid>b73ca6ed-5727-47f3-84de-015e03f6a88a</guid>
            </TargetGroupIdList>
            <RequestedTargetGroupNames>
              <string>Windows Server 2003</string>
              <string>Servers</string>
            </RequestedTargetGroupNames>
          </Details>
        </ComputerRollupInfo>
      </computers>
    </RollupComputers>
  </soap:Body>
</soap:Envelope>

<!-- REPORTING DATA SYNC PHASE: RollupComputers response -->
<?xml version="1.0" encoding="utf-8"?>
<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>
    <RollupComputersResponse xmlns="http://www.microsoft.com/SoftwareDistributio
n">
      <RollupComputersResult />
    </RollupComputersResponse>
  </soap:Body>
</soap:Envelope>

<!-- REPORTING DATA SYNC PHASE: GetOutOfSyncComputers request -->
<?xml version="1.0" encoding="utf-8"?>
<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>
    <GetOutOfSyncComputers xmlns="http://www.microsoft.com/SoftwareDistribution"
>
      <cookie>
        <Expiration>9999-12-31T23:59:59.9999999</Expiration>
        <EncryptedData />
      </cookie>
```

```
      <parentServerId>050cee4a-aaff-470c-8fc3-7a94cb3a4293</parentServerId>
      <lastRollupNumbers>
        <ComputerLastRollupNumber>
          <ComputerId>d07a4d51-7cb2-46d1-ad86-91195cb4325a</ComputerId>
          <RollupNumber>2</RollupNumber>
        </ComputerLastRollupNumber>
      </lastRollupNumbers>
    </GetOutOfSyncComputers>
  </soap:Body>
</soap:Envelope>

<!-- REPORTING DATA SYNC PHASE: GetOutOfSyncComputers response -->
<?xml version="1.0" encoding="utf-8"?>
<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>
    <GetOutOfSyncComputersResponse xmlns="http://www.microsoft.com/SoftwareDistr
ibution">
      <GetOutOfSyncComputersResult />
    </GetOutOfSyncComputersResponse>
  </soap:Body>
</soap:Envelope>

<!-- REPORTING DATA SYNC PHASE: RollupComputerStatus request -->
<?xml version="1.0" encoding="utf-8"?>
<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>
    <RollupComputerStatus xmlns="http://www.microsoft.com/SoftwareDistribution">
      <cookie>
        <Expiration>9999-12-31T23:59:59.9999999</Expiration>
        <EncryptedData />
      </cookie>
      <clientTime>2007-03-06T03:39:24.2520342Z</clientTime>
      <parentServerId>050cee4a-aaff-470c-8fc3-7a94cb3a4293</parentServerId>
      <computers>
        <ComputerStatusRollupInfo>
          <InstanceId>14720b83-5998-41f7-b586-6088cd8acd0c</InstanceId>
          <ComputerId>d07a4d51-7cb2-46d1-ad86-91195cb4325a</ComputerId>
          <EffectiveLastDetectionTime>2007-03-06T01:47:46.42Z</EffectiveLastDet
ectionTime>
          <RollupNumber>3</RollupNumber>
          <IsFullRollup>true</IsFullRollup>
          <UpdateStatus>
            <ComputerStatusRollupUpdateStatus>
              <UpdateId>09de7a79-5c09-4f3a-972e-cc1bbb7ca7e4</UpdateId>
              <SummarizationState>4</SummarizationState>
              <LastChangeTime>2007-03-05T22:25:11.64Z</LastChangeTime>
            </ComputerStatusRollupUpdateStatus>
            <ComputerStatusRollupUpdateStatus>
              <UpdateId>5eb6225f-5d33-4213-a6b4-fcdd9f672d75</UpdateId>
              <SummarizationState>5</SummarizationState>
              <LastChangeTime>2007-03-05T22:25:11.64Z</LastChangeTime>
            </ComputerStatusRollupUpdateStatus>
            <!--.......................-->
            <!--.....ARRAY TRIMMED.....-->
            <!--.......................-->
          </UpdateStatus>
        </ComputerStatusRollupInfo>
      </computers>
    </RollupComputerStatus>
  </soap:Body>
</soap:Envelope>
```

```
<!-- REPORTING DATA SYNC PHASE: RollupComputerStatus response -->
<?xml version="1.0" encoding="utf-8"?>
<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>
    <RollupComputerStatusResponse xmlns="http://www.microsoft.com/SoftwareDistr
ibution">
      <RollupComputerStatusResult>true</RollupComputerStatusResult>
    </RollupComputerStatusResponse>
  </soap:Body>
</soap:Envelope>
```

Sample 4: SOAP Fault

The following sample message shows a SOAP fault response from the USS to the DSS when the DSS sent an invalid cookie. It shows the <detail> section of the SOAP fault filled with <ErrorCode> "InvalidCookie". The <Method> element also shows the operation that returned this error: "GetDeployments".

```
<!-- DEPLOYMENTS SYNC PHASE: GetDeployments() Failure Response - 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/XMLSchema">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Client</faultcode>
      <faultstring>System.Web.Services.Protocols.SoapException: Fault occurred at
Microsof
t.UpdateServices.Internal.SoapUtilities.ThrowException(ErrorCode errorCode, String
message
) at
Microsoft.UpdateServices.Internal.Authorization.AuthorizationManager.CrackCookieInSer
verSync(Cookie cookie) at
Microsoft.UpdateServices.Internal.ServerImplementation.GetDeploy
ments(Cookie cookie, String deploymentAnchor, String syncAnchor) at
Microsoft.UpdateServic
es.Internal.ServerSyncProxy.GetDeployments(Cookie cookie, String deploymentAnchor, String
syncAnchor)</faultstring>
      <faultactor>http://HemantTest:8530/serversyncWebService/serversyncWebService.asmx</
faultactor>
      <detail>
        <ErrorCode>InvalidCookie</ErrorCode>
        <Message />
        <ID>247dfffd-a6b4-4a57-b9d4-2e464b668cd3</ID>
 <Method>"http://www.microsoft.com/SoftwareDistribution/GetDeployments"
        </Method>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

5 Security

The following sections specify security considerations for implementers of the Windows Update Services: Server-Server Protocol.

5.1 Security Considerations for Implementers

The USS provides the DSS with all updates that ultimately are intended for client computers. Care must be taken to prevent a man-in-the-middle or other forms of spoofing attacks from providing the DSS with invalid or malformed metadata and rogue content. To ensure secure download of all data from the USS, the DSS performs several checks:

- It only accepts updates that are signed by Microsoft certificates.
- It only accepts content files whose SHA1 hash matches the SHA1 hash specified by the USS in the update metadata.

As a result, it is strongly recommended that the USS be configured so that all metadata communication is done over HTTPS instead of HTTP. Using SSL server certificate verification ensures that the client is talking to the real server and closes any possible man-in-the-middle attacks.

There are two strategies one can use to reduce the impact of Denial Of Service (DOS) attacks against the USS:

- Turn on authentication, and deny access to unauthenticated DSS. This will allow one to quickly disable access to rogue DSS machines.
- Be sure no single operation takes too much processing time on the USS. This will ensure that any attacker must keep up a steady stream of requests to deny access to the server, 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 may try to pass an unbounded set of parameters to various methods).

If the USS implementation stores and displays any data passed to it from DSS, care must be taken to ensure 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, or used within SQL).

5.2 Index of Security Parameters

None.

6 Appendix A: Windows Behavior

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

- Windows 2000 SP3
- Windows Server 2003
- Windows Server 2008

The relationship between Windows versions and WSUS versions is specified as follows:

Operating system version	WSUS 2.0 supported	WSUS 2.0 SP1 supported	WSUS 3.0 supported
Windows 2000 Server SP3 and later	X	X	
Windows Server 2003	X	X	
Windows Server 2003 SP1 or later	X	X	X
Windows Server 2008	X	X	X

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: This protocol is implemented by the Windows Server Update Services (WSUS) component in Windows 2000 Server SP3, Windows 2000 Server SP4, Windows 2000 Server SP5, Windows Server 2003, and Windows Server 2008. All Windows behaviors described in the following sections apply to all versions of this component, unless specified otherwise.

<2> Section 2.1: All Windows implementations support HTTPS for securing ports; although SSL is not configured by default when WSUS is installed.

<3> Section 2.1: In all Windows implementations, the DSS adds "xpress" to the HTTP "Accept-Encoding" request-header to request the encoding in "Win 2K3 Compression Algorithm", as specified in [MS-DRSR]. The USS compresses the reply in the requested algorithm.

<4> Section 2.1: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, ports used by the Web services are administratively configured during the initial setup of the USS.

<5> Section 2.1: Windows XP, Windows Server 2003, and Windows Server 2008 implementations supports HTTP1.1 Byte Range requests.

<6> Section 2.2.1.3: Windows Server 2003Windows XP, , and Windows Server 2008 implementations construct the **EncryptedData** field by converting the DSS identity, the target groups that the DSS belongs to, and the cookie expiration time into a sequence of bytes, and then by encrypting the sequence of bytes using the TripleDES algorithm.

<7> Section 2.2.1.5: Windows XP, Windows Server 2003, and Windows Server 2008 implementations construct the **EncryptedData** field by converting the DSS identity, the target groups that the DSS belongs to, the cookie expiration time, protocol version, and server's identity into a sequence of bytes, and then by encrypting the sequence of bytes using the TripleDES algorithm.

<8> Section 2.2.2.3: Windows XP, Windows Server 2003, and Windows Server 2008 implementations abort the protocol when this error is encountered. They do not automatically retry the operation.

<9> Section 2.2.2.3: Windows XP, Windows Server 2003, and Windows Server 2008 implementations abort the protocol when this error is encountered. They do not automatically retry the operation.

<10> Section 3.1.1: Windows implementations remove entries from this table after 90 days, by default. The threshold can be configured administratively, down to a minimum of 1 day.

<11> Section 3.1.1: In Windows implementations, entries are removed from this table only when specified by the administrator.

<12> Section 3.1.1: Windows XP, Windows Server 2003, and Windows Server 2008 implementations use the version number of the WSUS component. If the update server is using WSUS 3.0, the value is "3.0.6000.374". If the update server is using an earlier version, the value is "2.0.0.0".

<13> Section 3.1.1: In Windows implementations, a new entry is added to this table when a new client computer calls the **GetAuthCookie** method on the update server for the first time as part of the Windows Update Services: Client-Server Protocol (as specified in [MS-WUSP]). Existing entries are removed only when specified by the administrator.

<14> Section 3.1.1: In Windows implementations, these values are populated when the client computer calls the **RegisterComputer** method as part of the Windows Update Services: Client-Server Protocol (as specified in [MS-WUSP]), and it is assumed that the values can be interpreted as specified in [MS-WUSP]. The data in this table is made available to the administrators of the update server for informational purposes.

<15> Section 3.1.1: In Windows implementations, this field is updated when the client computer calls the **ReportEventBatch** method of the WUSP (as specified in [MS-WUSP]).

<16> Section 3.1.1: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the update server receives such notifications from client computers when the **ReportEventBatch** method is called as part of the WUSP, as specified in [MS-WUSP]. Entries are added or modified when such notifications are received.

<17> Section 3.1.1: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the update server receives notifications from client computers when the **ReportEventBatch** method is called as part of the WUSP, as specified in [MS-WUSP]. Entries are added or modified when such notifications are received.

<18> Section 3.1.1: In Windows implementations, client computers notify the update server when they have successfully installed or failed to install an update. This is done using the WUSP (as specified in [MS-WUSP]).

<19> Section 3.1.4.1: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, this field is not present.

<20> Section 3.1.4.1: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, this field is not present.

<21> Section 3.1.4.3: Windows implementations set the cookie expiration time to be 240 minutes from the time the cookie is created or the expiration time of the AuthorizationCookie, whichever is lower.

<22> Section 3.1.4.4: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, this field is of the form nnnn ',' yyyy '-' MM '-' dd ' ' hh ':' mm ':' ss '.' sss where

nnnn is a positive decimal integer value between 1 and 2,147,483,647 that is used internally to identify a point in time. The value is 3 initially and is incremented by 1 each time a Deployment is created or deleted.

yyyy, MM, dd,hh, mm, ss, and sss are the current year, month, day, hour, minute, second, and fraction of a second, respectively.

<23> Section 3.1.4.5: All Windows XP, Windows Server 2003, and Windows Server 2008 implementations set this to the Anchor returned in the last successful **GetRevisionIdList** operation response.

<24> Section 3.1.4.5: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, this field is not present.

<25> Section 3.1.4.5: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, this field is not present.

<26> Section 3.1.4.5: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, this field is not present.

<27> Section 3.1.4.5: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the Anchor is required to follow the format described in section 3.1.4.4, under the Windows behavior description for the **NewConfigAnchor** field.

<28> Section 3.1.4.6: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, this field is present only for updates that are available on the Microsoft Update Web site. For such updates, this URL points to a location on the Microsoft Update Web site from which the file can be downloaded.

<29> Section 3.1.4.6: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, this field is not present.

<30> Section 3.1.4.6: All Windows implementations use the **XmlUpdateBlobCompressed** if the size of the uncompressed XML metadata is more that 5,120 bytes.

<31> Section 3.1.4.8: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the Anchor is required to follow the format defined in section 3.1.4.4 under the Windows behavior description for the **NewConfigAnchor** field.

<32> Section 3.1.4.8: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the Anchor is required to follow the format defined in section 3.1.4.4 under the Windows behavior description for the **NewConfigAnchor** field.

<33> Section 3.1.4.11: In Windows implementations, this value is always set to 5,000.

<34> Section 3.1.4.11: In Windows implementations, this value is always set to 1,500.

<35> Section 3.1.4.11: In Windows implementations, this value is always set to 20,000.

<36> Section 3.1.4.11: In Windows implementations, this value is always set to 500.

<37> Section 3.1.4.12: Windows XP, Windows Server 2003, and Windows Server 2008 implementations use the version number of the WSUS component. If the update server is using WSUS 3.0, the value is "3.0.6000.374". If the update server is using an earlier version, the value is "2.0.0.0".

<38> Section 3.1.4.12: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, two target groups are created during initial setup of the WSUS component: the "All Computers" and the "Unassigned Computers" groups. Because these groups are created during the initial setup of the update server, they are not counted by this field. If any additional target groups are created (by the administrator or by an application that interacts with the update server), those groups will be counted.

<39> Section 3.1.4.12: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the **OSMajorVersion**, **OSMinorVersion**, **OSBuildNumber**, **OSServicePackMajorNumber**, **OSServicePackMinorNumber**, **OSLocale**, **SuiteMask**, **OldProductType**, **NewProductType**, **SystemMetrics**, and **ProcessorArchitecture** fields are populated using the corresponding values passed by the client computer to the RegisterComputer method on the update server as part of the WUSP (as specified in [MS-WUSP]).

<40> Section 3.1.4.12: In Windows implementations, the USS calculates the difference between the current time (according to its system clock) and the **clientTime** value. If the absolute value of the difference is greater than 1 minute, the USS adds the difference to all other time stamps contained in the request.

<41> Section 3.1.4.13: In all Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the DSS will omit this field for a given client computer if none of the values have changed.

<42> Section 3.1.4.13: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the **OSMajorVersion**, **OSMinorVersion**, **OSBuildNumber**, **OSServicePackMajorNumber**, **OSServicePackMinorNumber**, **OSLocale**, **ComputerModel**, **BiosVersion**, **BiosName**, **BiosReleaseDate**, **SuiteMask**, **OldProductType**, **NewProductType**, and **SystemMetrics** fields are populated using the corresponding values passed by the client computer to the **RegisterComputer** method on the update server as part of the WUSP (as specified in [MS-WUSP]).

The **ComputerMake** field is populated using the ComputerManufacturer value passed by the client computer to the **RegisterComputer** method.

The **ClientVersion** field is populated by combining the ClientVersionMajorNumber, ClientVersionMinorNumber, ClientVersionBuildNumber, and ClientVersionQfeNumber values passed by the client computer to the **RegisterComputer** method with "." inserted between each of the four values.

<43> Section 3.1.4.13: In Windows implementations, the USS calculates the difference between the current time (according to its system clock) and the clientTime value. If the absolute value of the difference is greater than 1 minute, the USS adds the difference to all other time stamps contained in the request.

<44> Section 3.1.4.13: In Windows implementations, the USS maintains a list containing the **ComputerId** fields of client computers whose records have recently been deleted from the persisted store.

When the USS receives a call to **RollupComputers**, it checks the **ComputerId** from each ComputerRollupInfo structure against this list. If the **ComputerId** is found on the list, an entry of this form is added to the **RollupComputersResult** with this **ComputerId**.

<45> Section 3.1.4.15: In Windows implementations, the USS will do this if it is under extremely heavy load.

<46> Section 3.1.4.15: In Windows implementations, the USS calculates the difference between the current time (according to its system clock) and the clientTime value. If the absolute value of the difference is greater than 1 minute, the USS adds the difference to all other time stamps contained in the request.

<47> Section 3.2.3: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the DNS name of the USS is configured by the administrator using the WSUS administration UI.

<48> Section 3.2.3: In all Windows implementations, a DSS initiates this protocol either by a configuration-defined timer event (for example, every night at 1:00 A.M.) or by a manual request from the administrator UI. It also provides an API through which other programs can trigger the synchronization.

<49> Section 3.2.3: All Windows implementations provide an option in the WSUS administrator UI that allows the administrator to cancel a synchronization that is in progress.

If a cancel request is made while a SOAP call is in progress (a request has been sent but the response has not yet been received), the update server will wait until the response is received before aborting the protocol. If no call is currently in progress, the update server will abort the protocol immediately.

<50> Section 3.2.3: All Windows implementations provide an API through which other programs can trigger reporting data synchronization.

<51> Section 3.2.4.4: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, whenever a new entry is added to the Revisions table or to the Deployments table, the update server will determine if any of the files associated with the update meet the above criteria. If so, the update server will start to download all such files.

In addition, if either the CatalogSyncOnly flag or the LazySync flag in the Server Configuration is changed, the update server will re-examine all files to see if any of them need to be downloaded. It will then start the download for all files that need to be downloaded.

<52> Section 3.2.4.4: The Windows implementation does not batch multiple **FileDigest** values into a single call.

<53> Section 3.2.4.4: In Windows XP, Windows Server 2003, and Windows Server 2008 implementations, the Background Intelligent Transfer Service (for more information, see [MSDN-BITS]) is used to perform the download.

<54> Section 3.2.4.5: In Windows implementations, this step is run immediately following the completion of the Deployments Synchronization step if the DSS is configured as a replica server, or following the Metadata Synchronization step if the DSS is configured as an Autonomous Server. All Windows implementations also provide an API through which other programs can trigger Reporting Data Synchronization without running the preceding steps in this protocol.

<55> Section 3.2.4.5: An update is considered to be critical if it is a member of the "Critical Updates" or the "Security Updates" Update Classification.

<56> Section 3.2.4.5: In Windows implementations, such updates are identified by an attribute matching the XPATH query "/Update/Property/@IsMandatory" and having a value of TRUE.

<57> Section 3.2.4.5: Windows implementations send the **DownstreamServerInfo** structures in batches, minimizing the number of requests required to send all the information.

<58> Section 3.2.4.5: Windows implementations send the **ComputerRollupInfo** structures in batches, minimizing the number of requests required to send all the information.

<59> Section 3.2.4.5: Windows implementations send the **ComputerLastRollupNumber** structures in batches, minimizing the number of requests required to send all the information.

<60> Section 3.2.4.5: Windows implementations send the **ComputerStatusRollupInfo** structures in batches, minimizing the number of requests required to send all the information.

<61> Section 3.2.4.5: Windows implementations wait between one and five minutes (chosen randomly), and then resend the failed message. If the request is still unsuccessful after three retries (four total attempts), the protocol is aborted with a failure.

7 Index

T

U

V