[MS-WSUSO]:
Windows Server Update Services System Overview

This document provides an overview of the Windows Server Update Services System Overview Protocol Family. It is intended for use in conjunction with the Microsoft Protocol Technical Documents, publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A Protocol Family System Document does not require the use of Microsoft programming tools or programming environments in order to implement the Protocols in the System. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

Abstract

The Windows Server Update Services System enables IT administrators to distribute and manage software updates from a central location to a large number of computers. Administrators are able to approve software updates to groups of computers and retrieve status reports to monitor the state of update installations across those computers. The Windows Server Update Services System consists of one or more Windows Server Update servers and many Windows Server Update clients. The Windows Server Update server enables administrators to synchronize updates from a parent Software Update server, organize computers into groups for efficient update management, approve software updates for installation and generate reports on update installation activity. Multiple Servers can be configured as a hierarchy to allow a variety of deployment options with either autonomous control or centralized control. The Windows Server Update client is capable of detecting software updates that are applicable from the available set of updates on the server, install such updates and report installation activity back to the server.

This system requires communication between the Windows Server Update Services client and server to enable clients to discover software updates available on the server. In addition, it also requires communication between servers to propagate software update information, the updates, and administrative intent in a hierarchical deployment of the system.

This document describes the intended functionality of the Windows Server Update Services System and how the protocols in this system interact. It provides examples of some of the common user scenarios. It does not restate the processing rules and other details that are specific for each protocol. These details are described in the protocol specifications for each of the protocols and data structures that make up this system.

Revision Summary

Date	Revision History	Revision Class	Comments
05/22/2009	0.1	Major	First Release.
07/02/2009	0.2	Minor	Updated the technical content.
08/14/2009	1.0	Major	Updated and revised the technical content.
09/25/2009	2.0	Major	Updated and revised the technical content.
11/06/2009	3.0	Major	Updated and revised the technical content.
12/18/2009	4.0	Major	Updated and revised the technical content.
01/29/2010	5.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
03/12/2010	6.0	Major	Updated and revised the technical content.
04/23/2010	6.0.1	Editorial	Revised and edited the technical content.
06/04/2010	6.0.2	Editorial	Revised and edited the technical content.
07/16/2010	6.0.3	Editorial	Changed language and formatting in the technical content.
08/27/2010	7.0	Major	Significantly changed the technical content.
10/08/2010	8.0	Major	Significantly changed the technical content.
11/19/2010	8.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	8.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	8.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This Protocol Family System Document (PFSD) is primarily intended to cover the Protocol Family as a whole. In conjunction with Member Protocol Technical Documents (TDs), which are intended to cover Member Protocols, it presents the rules for information exchange relevant to those Member Protocols and the Protocol Family that are used to interoperate or communicate with a Windows operating system in its various environments.

This document describes the Windows Server Update Services System. It provides an overview of the two member protocols that comprise this system and describes how they interact to provide solutions to the update management problems. In addition, specific scenarios are presented that highlight the design goals for the member protocols. Note that the details of the communication itself is specified in the Member Protocol Technical Documents and not duplicated here unless specifically used to clarify a concept.

The Windows Server Update Services System is designed to provide centralized software update management in an enterprise computing environment. It provides automated update discovery and delivery, and administrative control over update availability. The system is appropriate for use in Windows environments where computers are managed by an IT administrator. The intended audiences for this document are architects, developers, and testers who are involved in building an interoperable Windows Server Update Services System.

1.1 Glossary

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

globally unique identifier (GUID)

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

client computer
content
deployment
man-in-the-middle
metadata
revision
Simple Object Access Protocol (SOAP)
target group
update content directory
update server
Windows Update Agent (WUA)

The following terms are specific to this document:

administrative intent: This defines the **WSUS Administrator's** choices with respect to which of the available updates need to be installed on each of the computers being managed. This is expressed as a combination of **target groups**, group membership and update approvals.

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 **USS**.

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 **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**.

replica DSS: A **DSS** that obtains both **updates** and update **deployments** from its **USS**.

replica server: A replica server is used only to distribute approvals, groups, and updates. Replica servers are not administered separately.

reporting data: Reporting data describes data about update installation activity.

software update: A software **update** is any **update**, **update** rollup, service pack, feature pack, critical **update**, security **update**, or hotfix that is used to improve or to fix a software product.

synchronization: The process by which a **DSS** obtains update **metadata**, target groups, and update approvals from an **upstream server (USS)** in order to reconcile its state with the **USS**.

update: The combination of **metadata** and its associated **content**. An **update** is identified by a **GUID**.

update client: A computer that implements the Windows Update Services: Client-Server Protocol to get updates from an **update server**. A client can be a desktop computer, a server, or the **update server** itself.

update metadata: This is XML-formatted data containing information on an update. The combination of metadata and its associated content that is updated.

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

upstream server (USS): An **update server** that provides **updates** to other **update server**.

WSUS Administrator: A user who deploys the latest Microsoft product updates to computers running the Windows operating system. The WSUS Administrator can fully manage the distribution of updates that are released through Microsoft Update to computers on their network. They are responsible for creating **target group** and creating update approvals.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. Note that in [RFC2119] terms, most of these specifications should be imperative, to ensure interoperability. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

Any specification that does not explicitly use one of these terms is mandatory, exactly as if it used MUST.

1.2 References

This section contains normative and informative references relevant to the Windows Server Update Services System.

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-GPOL] Microsoft Corporation, "Group Policy: Core Protocol Specification", June 2007.

[MS-GPSO] Microsoft Corporation, "Group Policy System Overview", February 2009.

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

[MS-WUSP] Microsoft Corporation, "Windows Update Services: Client-Server Protocol Specification", September 2007.

[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., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

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

1.2.2 Informative References

[MC-BUP] Microsoft Corporation, "Background Intelligent Transfer Service (BITS) Upload Protocol Specification", October 2007.

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

2 Overview

Section 1, "Introduction", describes this Protocol Family System Document. This section introduces the system that is being documented.

2.1 System Summary

It is often difficult for IT administrators to keep the computers on their organization's network updated in a timely manner with software updates that are critical for secure operation. To address this need, IT administrators require centralized management for distribution of software updates. In addition to keeping software up to date, automated updates allow the IT administrator to test the updates before making them generally available and provide statistics about the penetration of the updates. This establishes a feedback loop to improve administrator confidence about the compliance of the managed computers around critical and security updates. From a scalability perspective, an update system needs to provide a solution that tailors the software updates to specific computer configurations without needing to evaluate every available software update. This is essential because updates required by a single computer are based on the hardware and software configuration and usually represent a minority of all available updates.

The Windows Server Update Services System provides the following solutions to software update problems:

- Software update discovery by computers
- Delivery of relevant updates to computers
- Update distribution controls for administrators
- Monitoring of software update activity

In this discussion a software update can be an update to an application or distribution of an application or driver for a hardware device.

For update discovery, the Windows Server Update Services System evaluates the rules contained in update metadata to determine whether a software update is required by the target computer. Update delivery is performed through HTTP file download.

Administrators control update distribution by placing computers into target groups and creating update approvals. A target group is a collection of computers (for example, servers or desktop computers) defined by the WSUS Administrator. These are used to treat a set of computers collectively rather than having to perform actions on a per computer basis. An update approval is an administrative intent about whether a software update ought to be made available for a given target group. Administrators use update approvals to control the availability of software updates to the computers that they manage.

In addition, administrators can configure a distributed server environment to manage update distribution to remote locations efficiently. Software update installation is monitored by computers sending update detection and installation information using a reporting channel.

The Windows Server Update Services System consists of the following:

- One or more update servers that act as distribution points for updates, and receive and relay update status information. The update servers are essentially repositories for software update metadata.

- One update client on each computer that is configured to receive updates. The update client communicates with an update server to discover updates and send installation information.

In order to discover and install software updates, the update clients send a series of requests to the update server to progressively determine software updates that are applicable while eliminating updates that are not. At the end of the discovery phase the update client downloads the installation files from the update server to complete the update installation. The update client sends back information about the update detection and installation phases as events to the update server.

The Windows Server Update Services System can be implemented in a Windows environment by enabling the update server component on a computer with a Windows Server operating system and configuring the update client on computers with a Windows operating system to communicate with the update server. See section 8 of this document for the versions of Windows this system supports.

In a distributed deployment of the system with more than one update server, a downstream server (DSS) communicates with an upstream server (USS) to synchronize the update metadata and, optionally, administrative intent such as target groups and approvals. In addition, the downstream server can be configured to relay information about the update installation information that it collected from update clients to the upstream server.

The Windows Server Update Services System defines a set of internal protocols to enable server-to-server and client-to-server communication, listed in section 2.2. There are no external protocols defined by this system.

The following figure shows an example layout of multiple update servers deployed as a hierarchy. Computers are configured to receive updates from one of the update servers in the hierarchy. The server to server communication is depicted using solid lines, and client to server communication is depicted using dashed lines. Within the context of each server pair communicating with each other, the parent server forms the USS, and the child server forms the DSS.

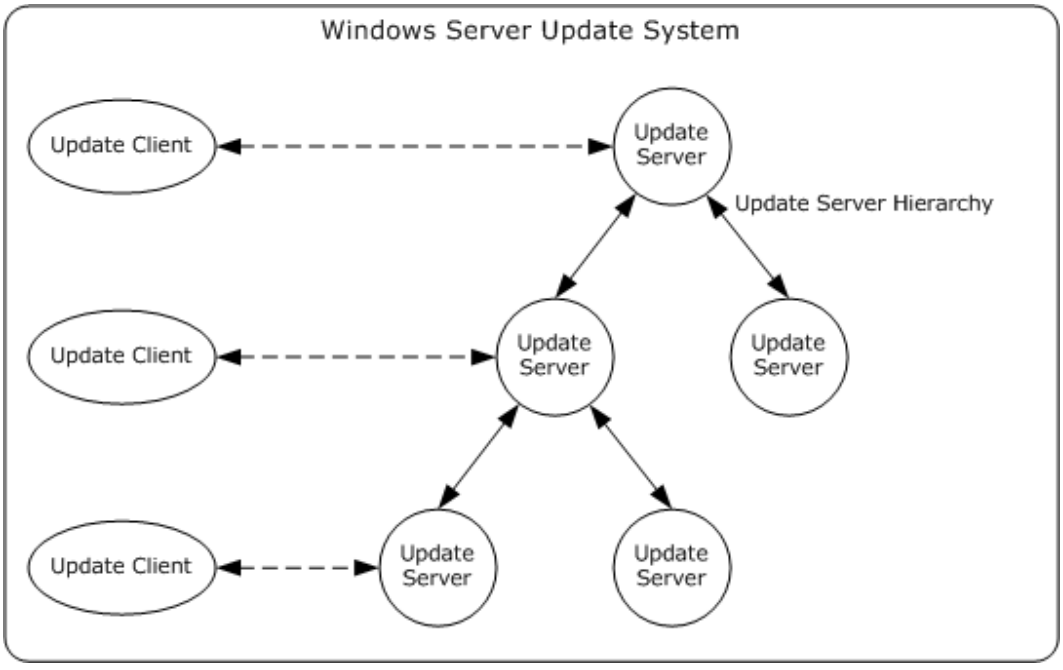

Figure 1: Windows Server Update Services System overview diagram

2.2 List of Member Protocols

The Windows Server Update Services System implements the following protocols:

Windows Update Services: Client-Server Protocol, as specified in [MS-WUSP]. This protocol enables software update discovery, delivery and reporting facilities for communication between an update client and update server.

Windows Update Services: Server-Server Protocol, as specified in [MS-WSUSSS]. This protocol enables software update synchronization, transfer of administrative intent, and relay of reporting data for communication between one update server and another update server.

[MS-WUSP] and [MS-WSUSSS] download the files associated with an update using HTTP. These files can optionally be downloaded using [MC-BUP].

2.3 Relevant Standards

The Windows Server Update Services System uses and extends the following standards:

Hypertext Transfer Protocol - HTTP/1.1 as specified in [RFC2616]. Enables file transfer from update server to update client.

SOAP Version 1.2 Part 1: Messaging Framework, as specified in [SOAP1.2-1/2003]. Serves as the base protocol from which most of the internal protocols for this system are defined.

3 Foundation

This section describes the theoretical and practical information needed to understand this document and this system.

3.1 Background Knowledge and System-Specific Concepts

This section summarizes:

- Background knowledge required to understand this document.
- Concepts that are specific to this system.

It is assumed that the reader of this document has the following background knowledge:

- **SOAP Web service** based protocols
- Use of XML to package data

The reader would benefit in understanding the system specific concepts described as follows.

A software update in the context of this system is either an update to an application or an update to a driver for a hardware device. The system treats any type of update the same way. The system defines a software update as update metadata plus the update files. The metadata contains information about other updates it depends on, rules that define under which conditions the update is applicable on a target computer, information about binary files that are used in the update installation process, and how the binary files ought to be applied on the target computer to complete the installation.

The Windows Server Update Services System has a hierarchical topology of servers with individual child servers either being configured as **autonomous DSS** or **replica DSS**, as defined in [MS-WSUSSS] section 1.3. A DSS synchronizes Update metadata and content as defined in [MS-WSUSSS] sections 3.2.4.2 and 3.2.4.4 (respectively). If the DSS is configured as a replica DSS, it additionally synchronizes the Update Approvals as defined in [MS-WSUSSS] section 3.2.4.3.

The update metadata, content, and deployment thus synchronized on a server in the system is used to determine available, applicable software updates for an individual update client. The protocol between an update client and its update server is defined in [MS-WUSP].

Individual update clients report the update installation activity to its update server as defined in [MS-WUSP] (section 3.2.4). Data from individual update clients are propagated by a DSS to its USS, based on the DSS and USS configuration as defined in [MS-WSUSSS] section 3.2.4.5. The reporting data provides the basis on which update installation reports can be generated by administrators to gauge the penetration and health of update distribution.

In this system, the term reporting data is used to describe data about update installation activity. Reporting data is generated by the update client on the target computer and it is sent to update servers. When the system is configured as a hierarchy, it can send the reporting data from a DSS to a USS. The reporting data provides the basis on which update installation reports can be generated by administrators to gauge the penetration and health of update distribution.

3.2 System Purposes

The Windows Server Update Services System provides the **WSUS Administrator** with the ability to control automated delivery of software updates to computers in an environment where the computers are being managed by one or more WSUS administrators. The system is designed to

solve the need to quickly and efficiently distribute software updates to computers without the need for an administrator to manually install the updates.

For WSUS administrators, the system is designed to accomplish the following:

- **Act as a centralized repository for software updates**. The system enables administrators to review available updates from one location, the update server. They can perform all the update related activities on the server without the need to examine client computers where the updates will be eventually applied.
- **Provide ways to control the delivery of software updates**. There is a variety of software updates with varying severity or criticality. Similarly, the computers being managed have varying requirements with respect to allowing change through software updating. For example, servers running line of business applications have strict requirements with regard to downtime and update testing. To balance these needs, the system provides administrators with tools to group computers of similar requirements together and allows or disallows installation of selected updates to such groups.
- **Enable automatic software update discovery, delivery and installation on client computers**. Not all updates available on the update server will be applicable on a given computer. The applicability of an update is based on the specific hardware and software configuration of the computer. The system performs the determination of update applicability automatically and only installs relevant updates on computers.
- **Allow deployment options which enable distributed administration or scale-out**. In large enterprises, there can be multiple branch offices with different administrators. In addition, the total number of computers might be such that one update server cannot handle the processing load. The Windows Server Update Services System solves this problem by allowing a deployment configuration where multiple update servers are deployed in a parent-child relationship forming a tree. The DSS servers can be independently administered or can receive administrative intent from the USS servers.
- **Provide reporting data about update installation activity**. This sets up a feedback loop for the administrator to check the update installation status. When the update servers are deployed as a hierarchy, the system allows for the root server to receive rolled up reporting data from all of the client computers across the hierarchy, enabling the administrator to assess the overall health of the computing environment across the enterprise.
- **Enable software update delivery and installation in a secure and scalable way**. Applying software updates essentially means executing external code and placing external code on a computer. This makes it critical that system has safeguards in place to prevent attackers from leveraging these capabilities to take control of or damage the computers.

For the end-user, the system is designed to accomplish the following:

- **Update software on the computer automatically or with minimum intervention**. Typically, the end-user wants the computer software to be kept up-to-date. The system allows for a scheduled discovery and installation mode where the update client periodically contacts the update server to determine if new updates are applicable and performs installation of the updates.
- **Allow configuration options to control the software update installation**. In cases where the end-user wishes additional control, the system allows client configuration options that enable more manual settings where update installation is driven by the end-user.

3.3 System Use Cases

3.3.1 Stakeholders and Interests Summary

The stakeholders in the Windows Server Update Services System are administrators, users, and management tools.

WSUS Administrator: The WSUS Administrator is the individual responsible for fully managing the distribution of updates that are released through Microsoft Update to computers on their network. The administrators install agents on the computers they want to manage. The WSUS Administrator will approve updates, monitor the update installations, configure server(s) for updates, and synchronize servers.

Computer User: A Computer User is a user of a computer that is managed by the update server. The computer can include laptops, desktops, and servers. The user can be required to keep their computer updated with the latest security patches and service packs. The update server can require the user either to install the updates right away or give the option of scheduling a more convenient time. The Computer User can also be required to configure the client and start update scan. Optionally, all of these tasks can be automated.

Server Management Tool: A tool that provides a administrative interface to the update server. The primary interest of the Server Management Tool is to enable a WSUS Administrator to read and update the update server configuration, display available updates, manage target groups and group membership, and approve updates. The Server Management Tool enables displaying update installation status reports to the WSUS Administrator.

Client Management Tool: A tool that provides a management interface to the update client. The primary interest of the Client Management Tool is to enable a Computer User to read and update the update client configuration, start a scan for updates, and control the download and installation of updates.

3.3.2 Supporting Actors and System Interests Summary

Management System: A supporting actor is a management system such as the Group Policy System. The management system is used by the WSUS Administrator to assign user-specified values to distribute configuration settings that control the client's behavior.

There are no other systems in which this system is an actor.

3.3.3 Use Case Diagram

The following diagram provides an overview of the Windows Server Update Services System use cases with the WSUS Administrator and Computer User as the primary actors.

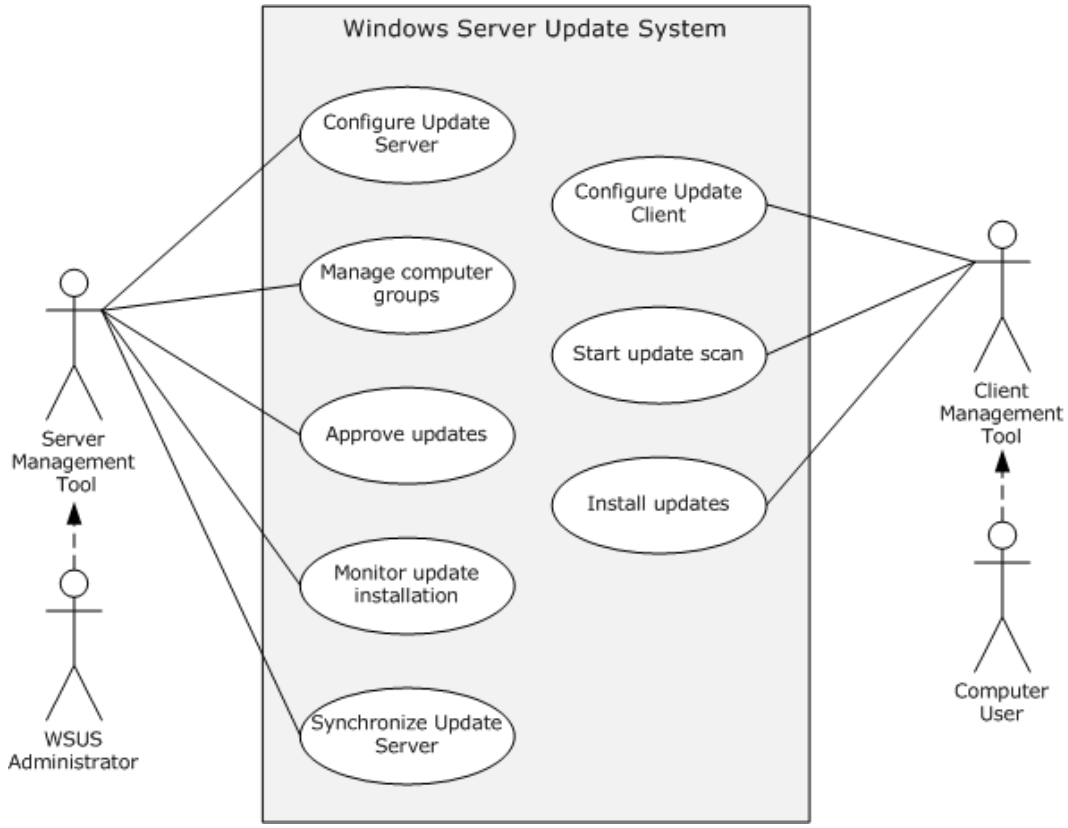

Figure 2: Windows Server Update Services System use cases

3.3.4 Use Case Descriptions

3.3.4.1 Configure Update Server -- WSUS Administrator

Goal: To configure the update server according to the deployment requirements.

Configuration options can include:

- A fully qualified domain name (FQDN) or the IP address of the **USS**.
- Whether a **DSS** receives only updates or if it receives update approvals from the USS.
- Whether a DSS sends detailed reporting data of the computers it manages or just a summary.
- Whether computer membership is based on groups assigned using client configuration or manual membership assignment on the server.
- Whether synchronization is manually triggered by the administrator or set up to run on a schedule.

The complete list of configuration options is specified in [MS-WSUSSS] Section 3.1.1.

Context of Use: This use case is initiated after the update server installation is complete and before the update server is used to service clients.

Direct Actor: The direct actor of this use case is the Server Management Tool.

Primary Actor: The primary actor is the **WSUS Administrator**.

Supporting Actors: None.

Stakeholders and Interests: The WSUS Administrator's primary interest is to customize the Windows Server Update Services System deployment to meet the needs of the organization's environment.

Preconditions: The update server is operational.

Minimal Guarantees: None.

Success Guarantee:

- If the server configuration is successful, the update server will persist configuration changes to the server data store.
- If the server configuration is successful, the server components that depend on the configuration use the updates configuration to determine their behavior.

Trigger: The Server Management Tool triggers this use case on behalf of the WSUS Administrator.

Main Success Scenario:

1. The server management tool identifies the relevant configuration elements that have changed.
2. The configuration is modified according to the WSUS Administrator's input.
3. The configuration is successfully saved to the data store.

Extensions:

- The USS from which the update server receives updates.
- Whether the update server receives updates or update approvals from the USS.
- Whether DSS's of the update server send detailed reporting data or a summary of the computers they manage.
- Whether computer membership is based on groups assigned using client configuration or manual membership assignment on the update server.
- Whether synchronization is manually triggered by the administrator or set up to run on a schedule.

3.3.4.2 Manage Computer Groups -- WSUS Administrator

Goal: To create computer target groups and establish membership of computers managed by the update server within those target groups.

Context of Use: The use case is initiated when the WSUS Administrator wants to do the following:

- Create an initial plan for how updates ought to be delivered to computers.
- Fine-tune the plan as requirements evolve.
- Add new computers to the system and assign them to target groups.

Direct Actor: The direct actor of this use case is the Server Management Tool.

Primary Actor: The primary actor is the WSUS Administrator.

Supporting Actors: None.

Stakeholders and Interests: WSUS Administrators perform this use case with the intent of organizing managed computers into groups to which they can apply the updating strategy. To minimize effort, the administrator's primary interest is to manage computers based on classes of computers rather than individual computers.

For example, it is typical for the administrator to create separate groups for desktop computers and servers as these classes of computers have very different updating requirements. Similarly, it is common for the administrator to create a test group for a representative set of computers. The administrator uses such a group to test the effects of updates and ensure normal operation before the software update is rolled out more generally.

Preconditions:

- The update server is operational.
- The update server is configured to use server-side targeting.

Minimal Guarantees: None.

Success Guarantee:

- For group creation, a target group entry is persisted in the server data store.
- For membership assignment, the information tying the computer to a target group is persisted in the server data store.
- The target group and computer membership information along with the update approvals are utilized during the client detection phase (use case 3.3.4.7) to determine which updates ought to be made available.

Trigger: The Server Management Tool initiates this use case on behalf of the WSUS Administrator.

Main Success Scenario:

1. The target group creation or target group membership is obtained from the WSUS Administrator.
2. This information is persisted in the server data store.

Extensions: None.

3.3.4.3 Approve Update -- WSUS Administrator

Goal: To approve an update to a target group for either installation or uninstallation. This is the primary use case that enables the system purpose of controlling update delivery.

Context of Use: This use case is initiated when the administrator wants to enable available updates to be installed or uninstalled on managed computers.

Direct Actor: The direct actor of this use case is the Server Management Tool.

Primary Actor: The primary actor is the WSUS Administrator.

Supporting Actors: None.

Stakeholders and Interests: The WSUS Administrator's primary interest for this use case is to distribute software updates to the managed computers.

Preconditions:

- The update server is operational.
- The update server is configured to be an autonomous update server.
- The update server has synchronized updates.
- Target groups have been created.

Minimal Guarantees: None.

Success Guarantees:

- The update approval information tying an update to a target group is persisted in the server data store.
- This information is utilized by the update server during the client detection phase (use case 3.3.4.7).

Trigger: The Server Management Tool triggers this use case on behalf of the WSUS Administrator.

Main Success Scenario:

1. The update approval information is obtained from the WSUS Administrator.
2. This information is persisted in the server data store.

Extensions: None

3.3.4.4 Monitor Update Installation -- WSUS Administrator

Goal: To generate update installation and applicability reports. How the system is implemented determines the type of reports that are generated. Reports can vary between high-level summary reports to detailed reports for a given update or a given computer.

Context of Use: This use case is initiated when the administrator wants to review which updates are needed by computers and which have installation failures.

Direct Actor: The direct actor of this use case is the Server Management Tool.

Primary Actor: The primary actor is the WSUS Administrator.

Supporting Actors: None.

Stakeholders and Interests: The WSUS Administrator's primary interest for this use case is to determine the health of the system and the results of this update distribution strategy.

Preconditions:

- The update server is operational.
- The update server has synchronized updates.

- Computers managed by the update server have completed detection and installation of updates (use cases 3.3.4.7 and 3.3.4.8).

Minimal Guarantees: None.

Success Guarantees:

- Based on the report type, the update server calculates the results of the update installation and applicability, and generates a formatted report.

Trigger: The Server Management Tool triggers this use case on behalf of the administrator.

Main Success Scenario:

1. The update server collects the raw update events from the data store related to the report being generated.
2. Any calculations required to summarize the data are performed.
3. The results are formatted and displayed in the management tool.

Extensions: None.

3.3.4.5 Synchronize Server -- WSUS Administrator

Goal: To synchronize a DSS in an update server hierarchy with updates and approvals from a USS.

Context of Use: This use case is initiated when the administrator wants to bring the update server into synchronization with a USS. This use case can also be initiated on a schedule.

Direct Actor: The direct actor for this use case is the server synchronization component on the update server.

Primary Actor: The primary actor for this use case is the WSUS Administrator.

Supporting Actors: None.

Stakeholders and Interests: The WSUS Administrator's primary interest is to ensure uniform access to updates across the enterprise when the system is deployed as a hierarchy of servers.

Preconditions:

- The update server is operational.
- The update server is configured as a DSS.
- The USS is operational.
- Network connectivity is present between the DSSs and USSs.

Minimal Guarantees: None.

Success Guarantee:

- Updates available on the USS that are not already present on the DSS are obtained and persisted in the DSS data store.
- When the DSS is configured as a replica server, the computer target groups and update approvals are also obtained and persisted in the DSS data store.

Trigger:

- The update server triggers this use case automatically.
- In addition, the Server Management Tool can trigger this use case on behalf of the administrator at any time.

Main Success Scenario:

1. The DSS contacts the USS to determine new updates that have been created since the last successful synchronization. If this is the first synchronization, then all updates are selected.
2. The selected updates are imported into the local data store.

Extensions:

- If the DSS is a replica server, then target groups and approvals are also selected.
- If the DSS is a replica server, the selected target groups and approvals are created in the local data store.

3.3.4.6 Configure Update Client --Computer User

The following use case diagram describes the interaction between the Computer User and the WSUS Administrator to update the client with user specified settings.

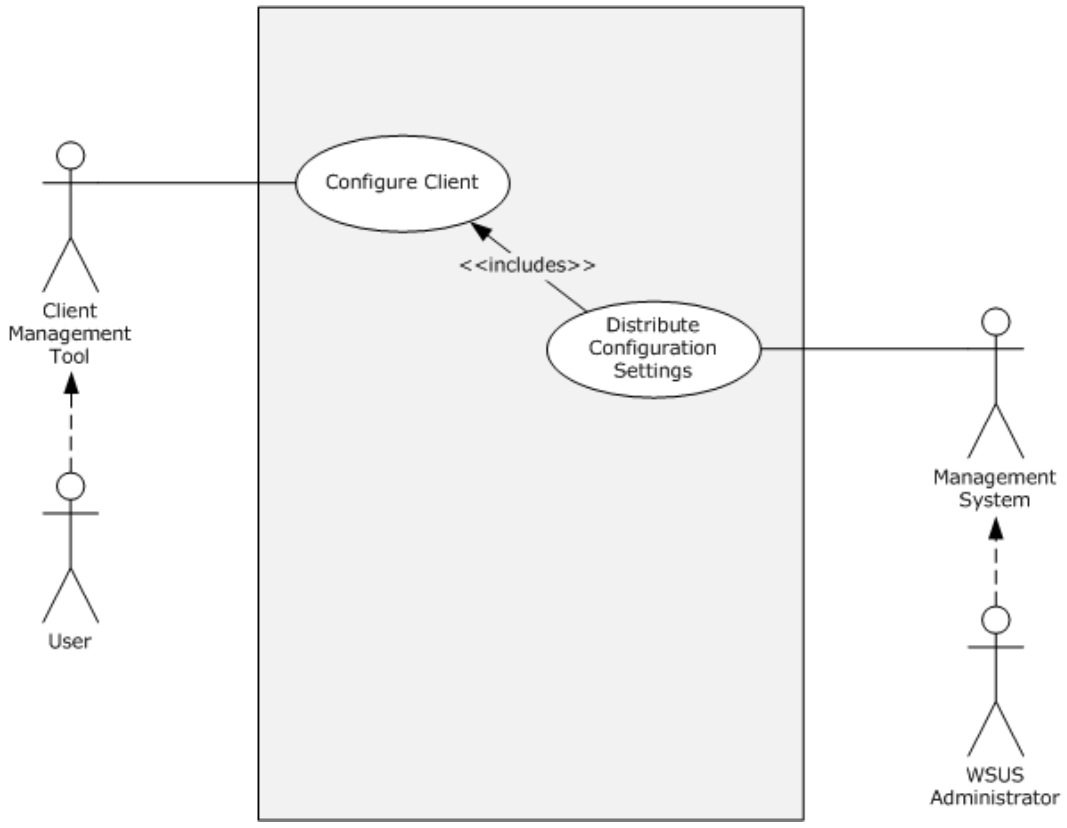

Figure 3: Configure Update Client use case

Goal: To assign user-specified values to configuration settings that control the client's behavior.

Context of Use: This use case is initiated at any time in order to initialize or modify configurable aspects of the client's behavior.

Direct Actor: The direct actor of this use case is a Client Management Tool.

Primary Actor: The primary actor of this use case is the WSUS Administrator or the Computer User.

Supporting Actors: A management system such as the Group Policy System.

Stakeholders and Interests:

- Computer Users, as described in section 3.3.2, expect that after this use case is executed, the client behaves in a manner consistent with the new configuration.
- WSUS Administrators, as described in section 3.3.2 can execute this use case on client computers they manage in order to apply an organizational policy regarding client configuration.

Preconditions:

- The client management tool is available to the Computer User and can communicate with the client.

Minimal Guarantees:

- A failure to modify configuration settings is not destructive to previously configured settings (previously configured settings are retained).

Success Guarantee:

- If the configuration is successful, then the user-specified values of the configuration settings are persistently stored by the update client and the update client's behavior affected by these settings is consistent with the configured values.

Trigger:

- A Computer User can trigger this use case by use of a Client Management Tool, or
- The WSUS Administrator can trigger this use case through a configuration system, for example, Group Policy as defined in [MS-WUSP] section 3.2.1.

Main Success Scenario:

1. The Computer User uses the Client Management Tool to provide values for one or more configuration settings.
2. The Client Management Tool communicates with the client (either directly or indirectly, for example by modifying shared state) in order to update its configuration settings with the values specified by the Computer User.
3. Future operations performed by the client behave in ways that are consistent with the configured values.

Extensions: None.

3.3.4.7 Start Update Scan -- Computer User

Goal: To discover changes in the set of updates available to the client computer and their deployments since the last time the use case was executed. Additionally, update metadata is retrieved from the update server for new updates. (The result of the first execution of this use case on a client computer is discovery of the entire set of updates available to the client computer, and their deployments and metadata.)

Context of Use: This use case is executed periodically to poll for changes in the update set and deployments.

Direct Actor: The direct actor of this use case is a Client Management Tool.

Primary Actor: The primary actor of this use case is a Computer User.

Supporting Actors: None.

Stakeholders and Interests:

- WSUS Administrators, as described in section 3.3.1 can execute this use case to understand the set of updates which are applicable to a given computer, or to understand the set of computers to which a given update is applicable.

Preconditions:

- The update client is required to know the network location of its server.
- The update client is required to be able to reach its server over the network.

Minimal Guarantees:

- If the update scan fails, the client will attempt to report the failed operation back to the server.

Success Guarantee:

- If the update scan is successful, the client's update set, update deployments, and update metadata are synchronized with the server.
- If the update scan is successful, the client will attempt to report the successful operation back to the server.

Trigger:

- A Computer User can trigger this use case by use of a Client Management Tool.
- An automated agent can trigger this use case periodically on a schedule.

Main Success Scenario:

1. The Computer User triggers the use case.
2. The client communicates with the server to synchronize the update set, update deployments, and update metadata. This data is cached on the client.
3. The client reports success back to the server.

Extensions: None.

3.3.4.8 Install Updates -- Computer User

Goal: To carry out the directive the WSUS Administrator specified for each update that is applicable to the client computer. These directives include Install and Uninstall. An update installation can also entail a download of update files.

Context of Use: This use case is executed after Start Update Scan (see use case 3.3.4.7) in order to carry out the WSUS Administrator directive that was synchronized from the server by the Start Update Scan use case.

Direct Actor: The direct actor of this use case is a Client Management Tool.

Primary Actor: The primary actor of this use case is a Computer User.

Supporting Actors: None.

Stakeholders and Interests:

- Computer Users, as described in section 3.3.1, expect that after this use case is executed, their computer is up to date.

Preconditions:

- The client is required to have executed the Start Update Scan use case and determined that there are applicable updates.

Minimal Guarantees:

- If the update is being installed and the download of update files is unsuccessful, the client will attempt to report the failed download back to the server.
- If the update installation or uninstallation is unsuccessful, the client will attempt to report the failed operation back to the server.

Success Guarantee:

- If the update installation or (uninstallation) is successful, then the update is installed on (uninstalled from) the client computer.
- If the update installation or uninstallation is successful, the client will attempt to report the successful operation back to the server.

Trigger:

- A Computer User can trigger this use case by use of a Client Management Tool.
- An automated agent can trigger this use case periodically on a schedule.

Main Success Scenario:

1. The Client Management Tool triggers this use case.
2. If the update is to be installed, the update client downloads the update from the update server.
3. The update client reports success back to the update server.

Extensions: None.

4 System Context

This section describes the relationship between this system and its environment.

4.1 System Environment

The system environment requires the following:

- Networked environment
- Implementation of SOAP, HTTP and HTTPS
- Persistent storage facility

4.1.1 Network Connectivity

The system requires a networked environment in which clients and servers are connected. However, constant connectivity is not required; client and server implementations can take advantage of intermittent periods of connectivity to communicate. This requirement is necessary because a primary function of the system is to transfer update-related information between clients and servers and servers and servers.

If the requirement is not satisfied, the system will not function. Temporary losses of connectivity can be mitigated by retrying the protocol operation at a later time.

4.1.2 Underlying Protocols

All member protocols of the system are implemented as a layer on top of other protocols. These underlying protocols include SOAP, HTTP and HTTPS. The system environment is required to provide implementations of these underlying protocols, for example, as operating system libraries.

If the requirement is not satisfied, the system will not function.

4.1.3 Persistent Storage Facility

The system requires a persistent storage facility so that Abstract Data Models (ADMs) can be maintained. Examples of such a facility include file systems and databases.

If the requirement is not satisfied, the system will not function.

4.2 System Assumptions and Preconditions

Given the environment described in section 4.1, the system has the following assumptions and preconditions.

- An update client is required to be initialized with the location of its update server.
- A DSS is required to be initialized with the location of its USS.

4.3 System Relationships

This section describes the relationships between the system and external components, system dependencies, and other systems influenced by this system.

4.3.1 Black Box Relationship Diagram

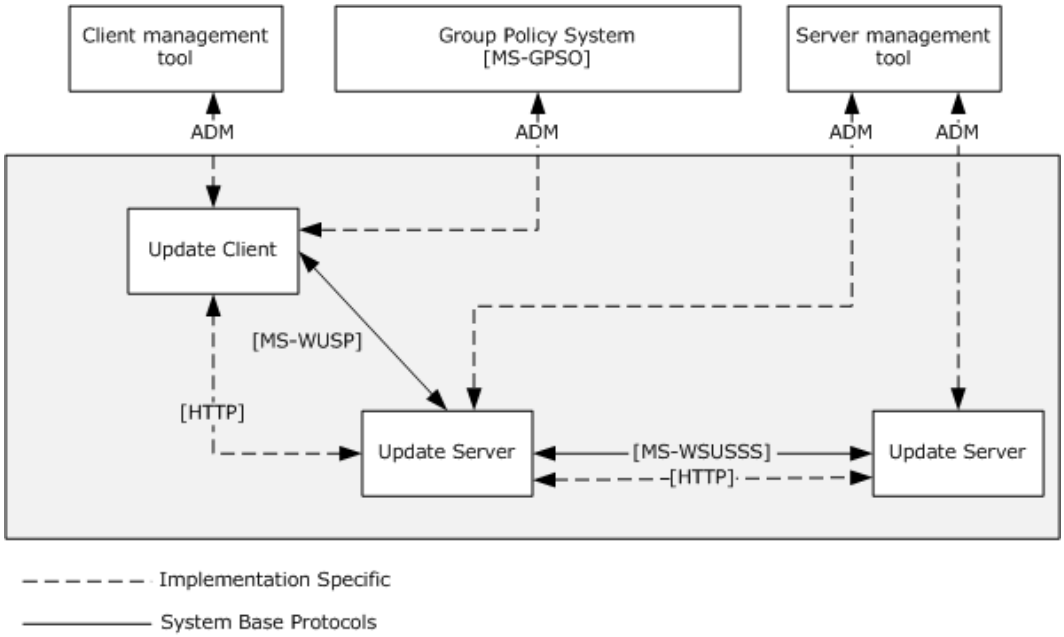

Figure 4: Black box relationship diagram

The system is depicted in the gray box and is composed of an **update client** and two **update servers**. Specifically, the diagram depicts two update servers in the upstream server (USS) and downstream server (DSS) roles and an update client.

[MS-WUSP]: Communication between an update client and an update server uses the Windows Update Services: Client-Server Protocol (WUSP).

[MS-WSUSSS]: Communication between the DSS and USS uses the Windows Update Services: Server-Server Protocol (WSUSSS).

[HTTP]: Content is downloaded by the update client and DSS using HTTP. Optionally, this content can be downloaded using [MC-BUP].

ADM: The abstract data model representing the state of the system is persisted in local storage on each of the individual computers.

Update client and update server management tools can interact with the system in an implementation-defined manner by altering the state of the abstract data model elements that are part of the member protocol or by using one of the specified protocol initialization mechanisms.

The ADM elements for the WUSP **server** and ways to populate them are specified in [MS-WUSP] sections 3.1.1 and 3.1.1.1. Out of the ADM elements specified, the server management tool can be used to alter the state of the **Client Computers Table**, **TargetGroup Table**, and the **Deployment Table** to express administrative intent.

The various initialization triggers that the server management tool can fire to initiate the Windows Update Services: Server-Server Protocol are specified in [MS-WSUSSS] section 3.2.3.

The Group Policy System can interact with the system by altering the state of the abstract data model element **Policy Table** as specified in [MS-WUSP] section 3.2.1.

4.3.2 System Dependencies

This system has no dependencies outside of those enumerated in section 4.1.

4.3.3 System Influences

The system SHOULD use a configuration system to configure update clients. Configurable aspects of update clients are described in [MS-WUSP] in sections 1.5, 2.1, 2.2.2.1.1, 3.2.2, and 3.2.4. An example of such a configuration system is:

Group Policy: Core Protocol [MS-GPOL]: Used to control the WUSP client.

HTTP is used to download files associated with software updates for both WUSP and WSUSSS communication. These interactions are shown in the black box diagram using dashed lines as being a possible implementation of the system. Optionally, [MC-BUP] can be used for downloading the files.

4.4 System Applicability

This system is appropriate for management of software updates for groups of computers. Additionally, it is applicable to situations in which an organization contains several groups of computers that are to be managed separately.

The system is not useful for situations in which computers requiring software updates are not on the network with the system's servers.

4.5 System Versioning and Capability Negotiation

There is no capability negotiation that is associated with this system. Any deviations from a specific version's implementation of these protocol specifications are documented in the respective protocol document. Capability negotiations between client and server implementations of these protocols are specified in the System Versioning and Capability Negotiation sections in their respective technical documents (TDs). Please see [MS-WUSP] section 1.7 and [MS-WSUSSS] section 1.7 for further details.

If the DSS or USS or client is not of a sufficiently high version to support the transfer of the data, then the client will ultimately not receive that data.

4.6 System Vendor-Extensible Fields

The system does not define any vendor-extensible fields.

5 System Architecture

This section describes the basic structure of the system and the interrelationships among its parts, consumers, and dependencies.

5.1 Abstract Data Model

5.1.1 Member Protocol Abstract Data Models

Each member protocol specifies an abstract data model (ADM) as follows:

- Windows Update Services: Client-Server Protocol [MS-WUSP] specifies an ADM for data used for communication between clients and servers. The ADM is maintained on the server. This ADM is specified in [MS-WUSP] section 3.1.1.
- Windows Update Services: Server-Server Protocol [MS-WSUSSS] specifies an ADM for data used for communication between USS and DSS. The same ADM is used on both USS and DSS. This ADM is specified in [MS-WSUSSS] section 3.1.1.

5.1.2 Data Elements with System-Level Significance

Some elements of the two member protocol ADMs have system-level significance, either because they are shared between both member protocols or because they influence the behavior of the entire system.

The two member protocol ADMs model the structure of a shared data element in different ways. This is due to differences in the way each protocol uses a data element. (For example, each protocol may use a different subset of the data element).

As a general requirement, when member protocols share state and one of the protocols performs an operation on that shared state that is specified as atomic in the sections below, other protocols sharing that state MUST honor the atomicity of that transaction. They MUST NOT attempt to perform operations on the uncommitted transaction or expose the uncommitted transaction in their ADM.

5.1.2.1 Update

This data element represents a single update with a unique identity throughout the system. It is composed of an update ID and metadata in XML form that describes the update, as specified in [MS-WUSP] section 3.1.1.1.

This data element is shared between both member protocols. The WUSP ADM models this element with the Revision, Metadata, and Driver tables. The WUSP protocol only reads from this data element. The WSUSSS ADM models this element with the Revision table. The WSUSSS protocol both reads from and writes to this data element. This data element in the Revision table in the WSUSSS ADM model is populated as defined in [MS-WSUSSS] section 3.1.1.1 and section 3.2.4.2. WSUSSS MUST perform operations on the Revision table for this data element in an atomic transaction.

The system loads, stores, and communicates updates. It is persisted for as long as the update is relevant.

5.1.2.2 Update Deployments

This data element represents an administratively approved action to be taken on a particular update. It is primarily composed of an update ID, an action to be taken on the update, additional

information about how a client computer treats the update, and a target group of client computers to which the deployment is applicable.

This data element is shared between both member protocols. Both the WUSP ADM and WSUSSS ADM model the update deployments with the **Deployment** table. The WUSP protocol only reads from this data element. The WSUSSS protocol both reads from and writes to this data element. This data element in the Deployment table in the WSUSSS ADM model is populated as defined in [MS-WSUSSS] section 3.1.1.1 and section 3.2.4.3. WSUSSS MUST perform operations on the Deployment table for this data element in an atomic transaction.

The system loads, stores and communicates update deployments. It is persisted until an administrator revokes approval of the action.

5.1.2.3 Target Groups

This data element represents a group of client computers that can be managed as a unit. It is primarily composed of a unique **target group** name.

This data element is shared between both member protocols. The WUSP ADM models target groups with the TargetGroup table. The WUSP protocol only reads from this data element. The WSUSSS ADM models target groups with the TargetGroup table. The WSUSSS protocol both reads from and writes to this data element. This data element in the TargetGroup table in the WSUSSS ADM model is populated as defined in [MS-WSUSSS] section 3.1.1.1 and section 3.2.4.3. WSUSSS MUST perform operations on the TargetGroup table for this data element in an atomic transaction.

The system loads, stores and communicates target groups. A target group is persisted until an administrator removes it.

5.1.2.4 Client Computer

This data element represents a single client computer. It is primarily composed of a computer ID, its target group membership, data describing the configuration of the computer, and data describing its software updating activities.

This data element is shared between both member protocols. The WUSP ADM models client computers with the **Client Computers Table**. The WUSP protocol both reads from and writes to this data element. The WSUSSS ADM models client computers with the Client Computers table. The WSUSSS protocol both reads from and writes to this data element.

The system loads, stores and communicates with client computers.

5.1.2.5 Policy Table

This data element represents the configurations for a **client computer**. It is composed of the update server and its target group membership.

This data element is written by Group Policy System [MS-GPSO] and is read by WUSP Client ADM as specified in [MS-WUSP] section 3.2.1.

5.2 White Box Relationships

This section describes the primary relationship between the Windows Update Services: Client-Server Protocol (WUSP) and the Windows Update Services: Server-Server Protocol (WSUSSS) protocol.

5.2.1 Relationships Among Member Protocols

The primary relationship between the Windows Update Services: Client-Server Protocol (WUSP) and the Windows Update Services: Server-Server Protocol (WSUSSS) are shared ADM elements. Several ADM elements are shared between the member protocols.

Sharing takes place on an update server. An update server can participate in protocol exchanges with a USS or a DSS using WSUSSS as well as protocol exchanges with update clients using WUSP. It maintains an ADM for each protocol and these ADMs substantially overlap. Thus, elements of the ADM that are modified by a WSUSSS protocol exchange can be consumed by a WUSP protocol exchange, and vice versa.

Consider the topology shown in the following diagram.

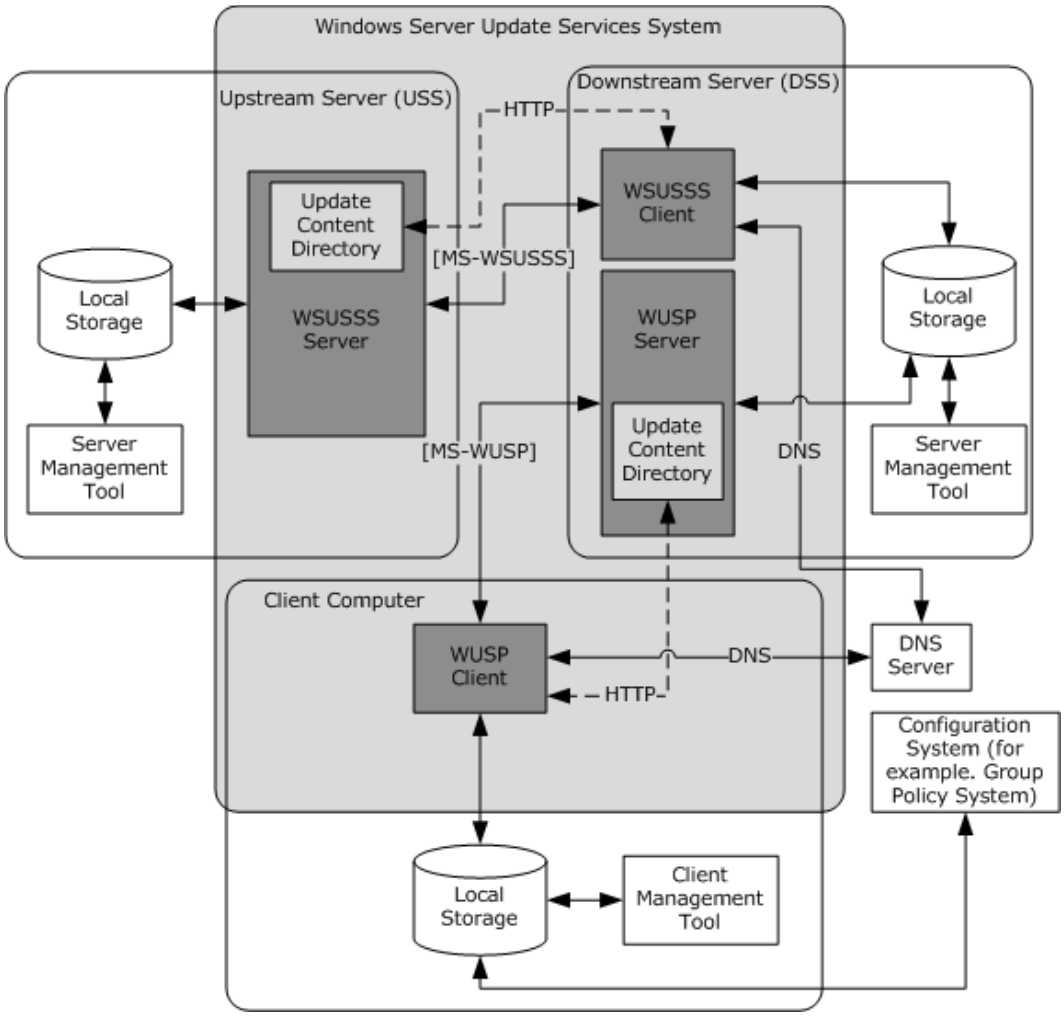

Figure 5: Example of a system topology

An update client synchronizes updates from an update server, which can be a DSS that itself synchronizes updates from a USS. Two examples are illustrated in the preceding figure.

In the first example, suppose a new update is introduced to the USS. The DSS initiates an update synchronization from the USS using WSUSSS, which returns the new update. The DSS adds the new update to its WSUSSS ADM. Later, the update client initiates an update synchronization from the downstream server using WUSP. Since an update is a shared ADM element, it also appears in the downstream server's WUSP ADM and is returned to the update client.

The second example describes data introduced from the client computer. The update client can report client computer information to the DSS using WUSP. The DSS inserts this data into its WUSP ADM. Since this data is shared with its WSUSSS ADM, this data (or a derivative of it) is sent to the USS using a WSUSSS protocol exchange.

A data dependency between the protocols is introduced through the sharing of ADM elements. In the first example, the update client cannot synchronize the new update via WUSP until the DSS has synchronized it from the USS via WSUSSS. In the second example, the USS cannot learn of the client computer information until the update client has reported its client computer information to the DSS.

5.2.2 Relationships Between the System and External Entities

The system can depend on other systems or external entities to provide configuration data or other services. The relationships specified in this section are not required to implement the system, but are examples of such relationships.

5.2.2.1 External Configuration System

An external system that provides configuration data can be used to configure clients and servers of this system. One such system is the Group Policy System [MS-GPSO].

An external configuration system can influence the system by configuring client's ADM values for:

- Update server
- Target groups

Configuration of these values is specified in [MS-WUSP] section 3.2.1.

5.2.2.2 Update Content Download Using HTTP

[MS-WUSP] and [MS-WSUSSS] download the update content using HTTP. The content can optionally be downloaded using MC-BUP.

5.3 Member Protocol Functional Relationships

5.3.1 Member Protocol Roles

The system is composed of two protocols:

- Windows Update Services: Client-Server Protocol, as specified in [MS-WUSP].
- Windows Update Services: Server-Server Protocol, as specified in [MS-WSUSSS].

Windows Update Services: Client-Server Protocol is the protocol specifying update-client-to-update-server communication. Its purpose is to transfer **update** metadata and deployments from an update server to an update client, and to report status information from an update client to an update server. These operations might rely on the update server having acquired update metadata and

deployments from another update server using the Windows Update Services: Server-Server Protocol.

Windows Update Services: Server-Server Protocol is the protocol specifying update-server-to-update-server communication. Its purpose is to transfer update metadata and deployments from an upstream server (USS) to a downstream server (DSS), and to send aggregated status information from a DSS to a USS. These operations might rely on the DSS having acquired status information from update clients using the Windows Update Services: Client-Server Protocol.

5.3.2 Member Protocol Groups

The system does not have any recognizable internal group structure.

5.4 System Internal Architecture

5.4.1 Communication Within the System

The following figure shows communication within the system.

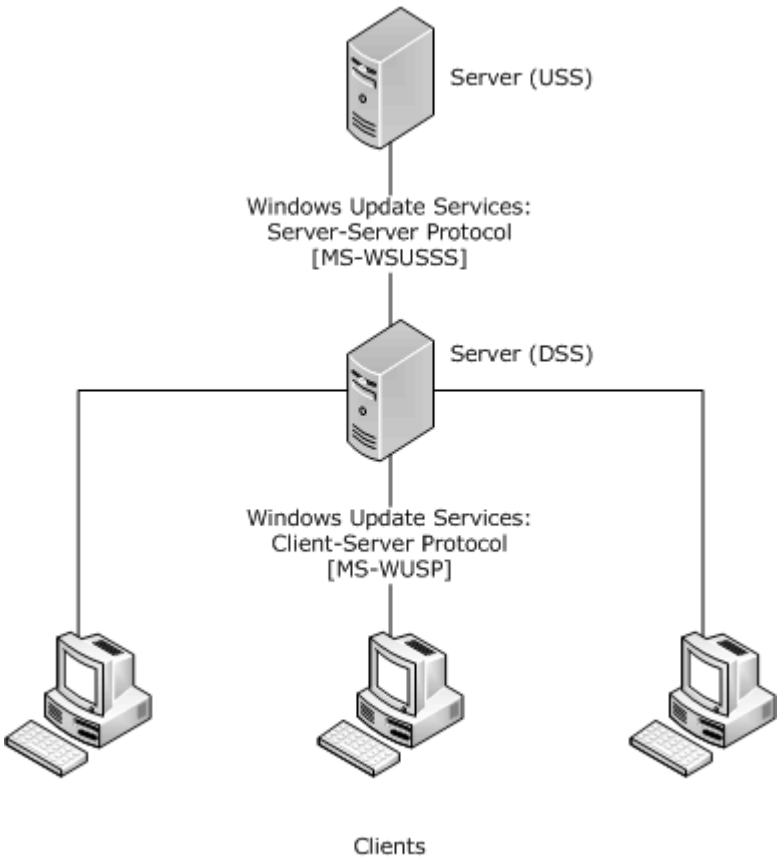

Figure 6: Communication within the system

Within the system, the logical endpoints of communication are either clients or servers. (Note that a particular computer can be both a client and a server from the point of view of the system.)

Communications between clients and servers are specified in [MS-WUSP]. Client-server communication primarily involves the following information transfers:

- Update metadata, update content and update deployments from the server to the client.
- Client computer information from the client to the server.

Communication between servers is specified in [MS-WSUSSS]. Server to server communication primarily involves the following information transfers:

- Update metadata, update content, update deployments and target groups from the USS to the DSS.
- Client computer information from the DSS to the USS.

5.4.2 Communication Between the System and External Entities

As stated in section 6.2, communication between the system and external entities is not required. However, one possible communication involves the provision of configuration data for the system by an external configuration system as described in [MS-GPSO].

5.5 Failure Scenarios

5.5.1 Network Failure

A common failure scenario is the inability of an update client or update server to contact an update server due to an underlying network failure. This can be caused by a lack of connectivity to the update server or any number of problems in the underlying networking layers.

This failure results in an inability for update clients or update servers to synchronize updates, deployments, or target groups, or to report back client computer information. The failure is typically recognized in the subsystems implementing the protocols by inspecting error information provided by lower level operating system networking components (for instance, error codes or exceptions).

Recovering from this error entails retrying the protocol operation at a later time.

5.5.2 Data Stores Corrupted

This failure scenario occurs when an update client or update server's data store becomes corrupt. This can occur due to an error in the underlying storage facility or hardware, which is implementation-specific.

This failure results in an inability for update clients or update servers to retrieve stored information. This can result in data loss. The failure is typically recognized by inspecting error information provided by storage components.

Recovering from this error entails resetting the data store to an empty state and resynchronizing data from a server. This can be either a manual operation or done automatically if the implementation can recognize data store corruption; in either case this recovery is not part of the protocol itself.

5.5.3 Update Content Is Corrupted

This failure scenario occurs when the installation files associated with an update are corrupted or modified either due to unintentional causes, such as network unreliability, or an intentional attempt to breach security.

As a safeguard, the client should only accept **content** for an update that is signed by a trusted certificate and whose hash matches the value retrieved from the **update metadata** (see [MS-WUSP] section 5). These cryptographic techniques ensure that the client can detect corrupt content in all cases.

Once corruption is detected, as part of the recovery steps, it is suggested that the client not attempt installation and that the client delete the suspect files.

Because no change is made to the client environment, when corrupt content is encountered, there are no negative consequences from recovery.

6 System Details

This section contains the details that complete the descriptions in earlier sections of the document. These details are needed to understand and implement this system.

6.1 Architectural Details

This section provides a series of examples illustrating the use of the Windows Server Update Services System. The examples are:

- Update synchronization to DSS
- Approval sync to replica DSS
- Initial update sync to update client
- Differential update sync to update client
- Rollup of reporting data to USS
- Update Client is pointed to a new update server

6.1.1 Update synchronization to DSS

This example describes the scenario where a DSS is configured to synchronize with an USS and the synchronization is triggered using the server management tool or triggered on a schedule from the down stream server. This scenario supports the use case in section 3.3.4.3 (Synchronize Server - WSUS Administrator).

The following sequence diagram shows the sequence of messages exchanged between the DSS and USS to accomplish this scenario. All communication between the USS and the DSS is initiated by the DSS and happens over the WSUSSS member protocol.

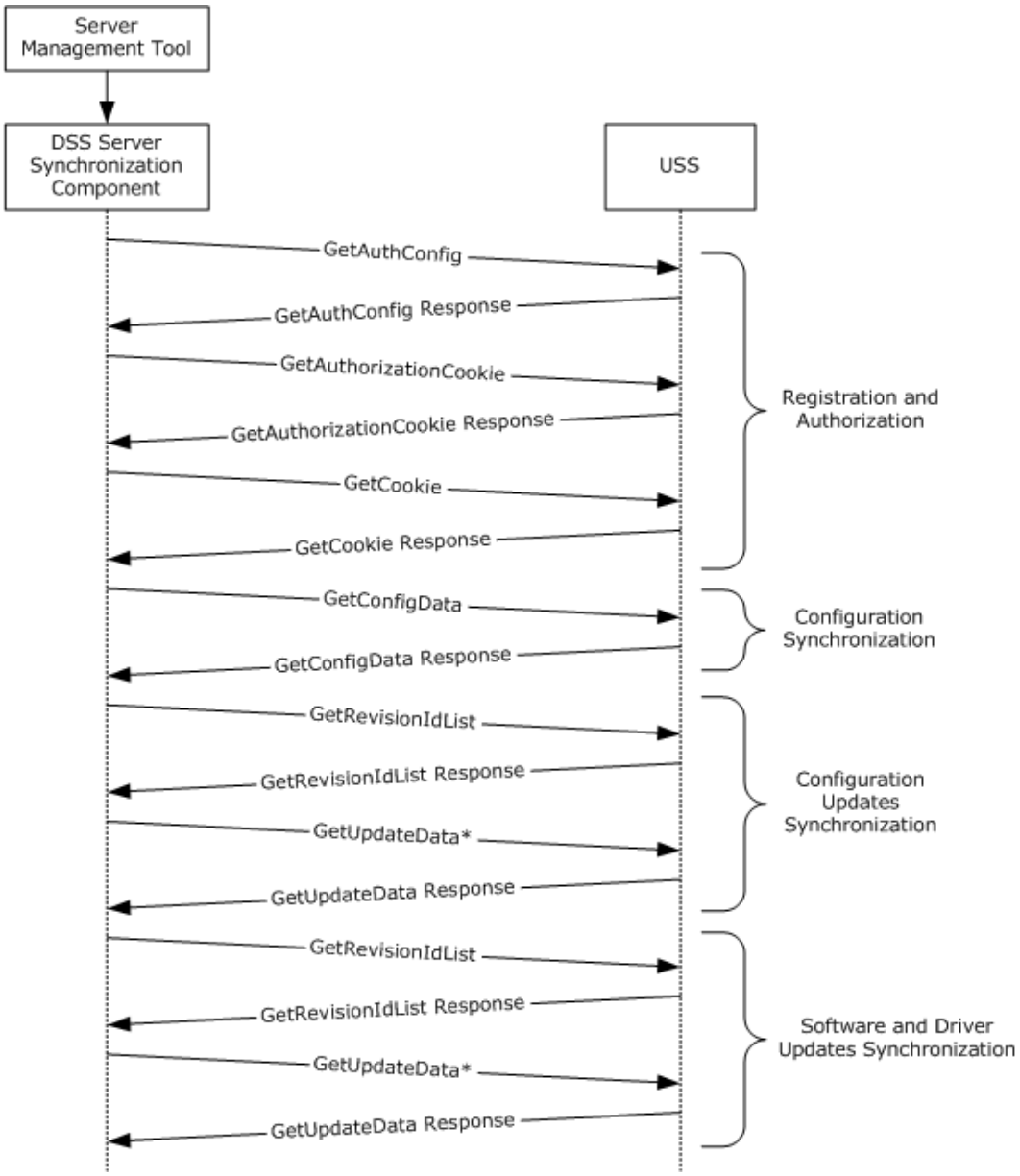

Figure 7: Sequence diagram for initial update sync to DSS

The phases in the message flow are described in the following sections.

6.1.1.1 Registration and Authorization

During this phase of the message exchanges, the DSS establishes its identity to the USS and the USS provides a cookie to the DSS that is to be used for the remaining requests. If this is the first synchronization, this results in the USS creating an entry for the DSS in its data store. The following sections in [MS-WSUSSS] specify the message details:

- Section 3.1.4.1: GetAuthConfig

- Section 3.1.4.2: GetAuthorizationCookie
- Section 3.1.4.3: GetCookie

6.1.1.2 Configuration Synchronization

During this phase, the DSS obtains the configuration data from the USS that governs the metadata and content synchronization phases of the protocol and persists this information in its data store. This is described in [MS-WSUSSS] section 3.1.4.4: GetConfigData.

6.1.1.3 Configuration Updates Synchronization

If this is an initial synchronization, the DSS obtains and persists the full set of categories, update classifications and detectoids available on the USS. On subsequent synchronizations, the DSS obtains and persists the categories, update classifications and detectoids that have been added to the USS since the last successful synchronization. The DSS obtains the list of such updates using a single **GetRevisionIdList** request ([MS-WSUSSS] section 3.1.4.5); note that on subsequent synchronizations, the DSS must include the anchor it received from the USS as part of the previous synchronization in the **GetRevisionIdList** request. The update metadata is then obtained through a series of batched **GetUpdateData** requests ([MS-WSUSSS] sections 3.1.4.6 and 3.2.4.2).

6.1.1.4 Software and Driver Updates Synchronization

If this is the initial synchronization, the DSS obtains and persists the full set of software and driver updates available on the USS. The DSS first obtains the list of software and driver updates using a single **GetRevisionIdList** request ([MS-WSUSSS] section 3.1.4.5). The update metadata is then obtained through a series of batched **GetUpdateData** requests ([MS-WSUSSS] sections 3.1.4.6 and 3.2.4.2). This phase differs from the previous phase with respect to the type of updates being synchronized. The DSS controls this using the parameters for the **GetRevisionIdList** request. On subsequent synchronizations, the DSS obtains and persists software and driver updates that have been added to the USS since the last successful synchronization; the DSS must include the anchor it received from the USS as part of the previous synchronization as specified in the **GetRevisionIdList** request.

Additional details about message sequencing and processing for this scenario are specified in [MS-WSUSSS]

- Section 3.2.4.1: Authorization
- Section 3.2.4.2: Metadata Synchronization

By the end of this scenario, the DSS has the same set of updates in its data store as the USS.

6.1.2 Initial Approval Sync to Replica DSS

When the DSS is configured as a replica of the USS, it synchronizes target groups and update approvals also from the USS in addition to updates. This example describes the message exchanges required to accomplish this synchronization when a DSS communicates with a USS. This is a continuation of the initial update synchronization example described in section 6.1.1 for a replica DSS that is specific to replica down stream servers. This scenario supports the use case in section 3.3.4.5 (Synchronize Server -WSUS Administrator).

The following sequence diagram shows the messages exchanged between the DSS and USS for this example. All the messages involved are part of the WSUSSS member protocol.

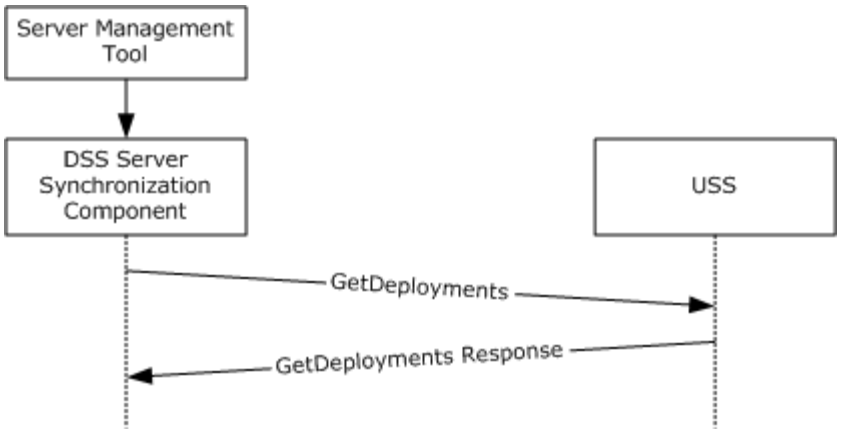

Figure 8: Sequence diagram for initial approval sync to replica DSS

The USS provides the list of all target groups and update approvals as part of the GetDeployments response during the first synchronization. The response also contains an anchor for use in subsequent GetDeployments requests. On subsequent synchronizations, only the changes since that previous synchronization are returned. [MS-WSUSSS] section 3.1.4.8 (GetDeployments) and section 3.2.4.3 (Deployments Synchronization) specify the message details and processing requirements.

6.1.3 Initial Update Sync to Update Client

The goal of this example is for a particular Update Client to synchronize update metadata and deployments from a particular Update Server for the first time. In this case, the Update Client has no cached data from previous synchronizations with the Update Server. This example is part of the Start update scan use case.

This scenario can be initiated on a schedule, by an automated agent, or by a user by means of a client management tool. After this scenario occurs successfully, the Update Client has synchronized update metadata and deployment and can cache this data to make subsequent synchronizations faster.

The following sequence diagram illustrates the interactions between parts of the system during this scenario. The notes following the diagram describe the messages with reference to the [MS-WUSP] technical document.

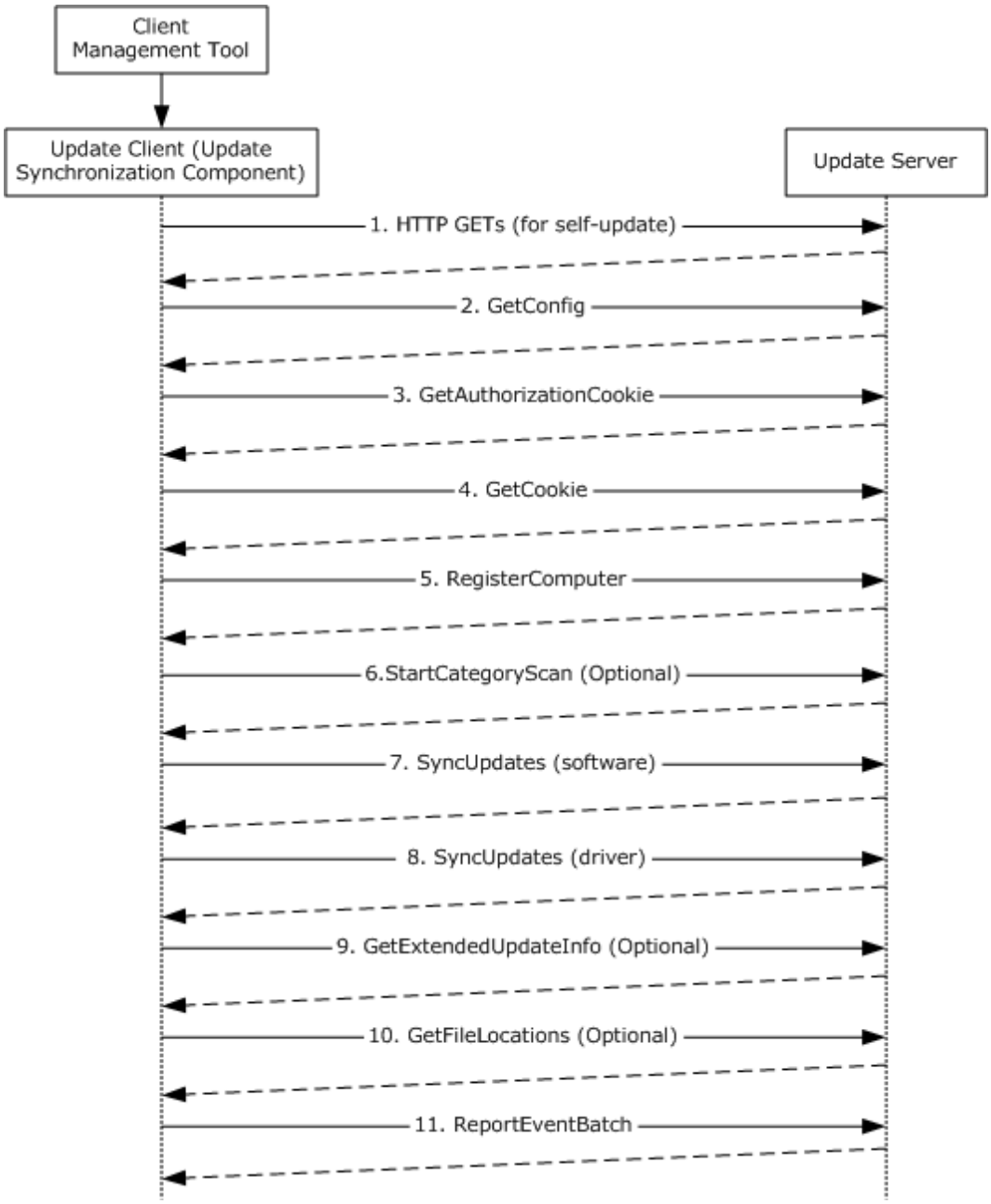

Figure 9: Sequence diagram for initial update sync to client

The message flow shown in the previous figure is as follows:

The update client makes a series of HTTP GET requests to determine the availability of applicable update client software. If applicable update client software is available, it downloads the software using HTTP GET requests and updates itself. The message details are specified in [MS-WUSP] section 3.1.5.1.

The update client sends a GetConfig message to the update server, which responds with configuration data. The message details are specified in [MS-WUSP] section 3.1.5.2.

The update client sends a GetAuthorizationCookie message to the update server, which responds with an authorization cookie. The message details are specified in [MS-WUSP] section 3.1.5.3.

The update client sends a GetCookie message to the update server, which responds with a cookie. The message details are specified in [MS-WUSP] section 3.1.5.4.

The update client sends a conditionally RegisterComputer message to the update server. The message details and conditions under which it is utilized are specified in [MS-WUSP] section 3.1.5.5.

If the update sync is to be limited to a subset of the updates available on the server rather than to all updates,then the update client sends the StartCategoryScan message first in order to obtain the category information to be used in the later SyncUpdates message. The message details are specified in [MS-WUSP] section 3.1.5.6.

The update client sends a SyncUpdates message for software updates to the update server, which responds with update information. There can be multiple iterations of this message and response. The message details are specified in [MS-WUSP] section 3.1.5.7.

The update client sends a SyncUpdates message for drivers to the update server. The message details are specified in [MS-WUSP] section 3.1.5.7.

If updates are determined to be applicable, the update client sends a GetExtendedUpdateInfo message to the server, which responds with extended update information for each of the new updates. The message details are specified in [MS-WUSP] section 3.1.5.9.

If there is a new update, the message GetFileLocations will be sent to get the new FileLocation information. The message details are specified in [MS-WUSP] section 3.1.5.10.

The update client sends a ReportEventBatch message to the update server, which responds with the status. The message details are specified in [MS-WUSP] section 3.1.5.11.

6.1.4 Differential Update Sync to Update Client

The goal of this example is for a particular Update Client to synchronize update metadata and deployments from a particular Update Server after having already synchronized at a previous point in time. In this case, the Update Client has cached data from previous synchronizations with the Update Server, which it uses to optimize the synchronization. This example is part of the Start update scan use case. This scenario supports the use case in section 3.3.4.7.

The scenario described in this example can be initiated on a schedule by an automated agent, or by a user by means of a client management tool. After this scenario occurs successfully, the Update Client has synchronized update metadata and deployment and can cache this data to make subsequent synchronizations faster.

The following sequence diagram illustrates the interactions between parts of the system during this scenario.

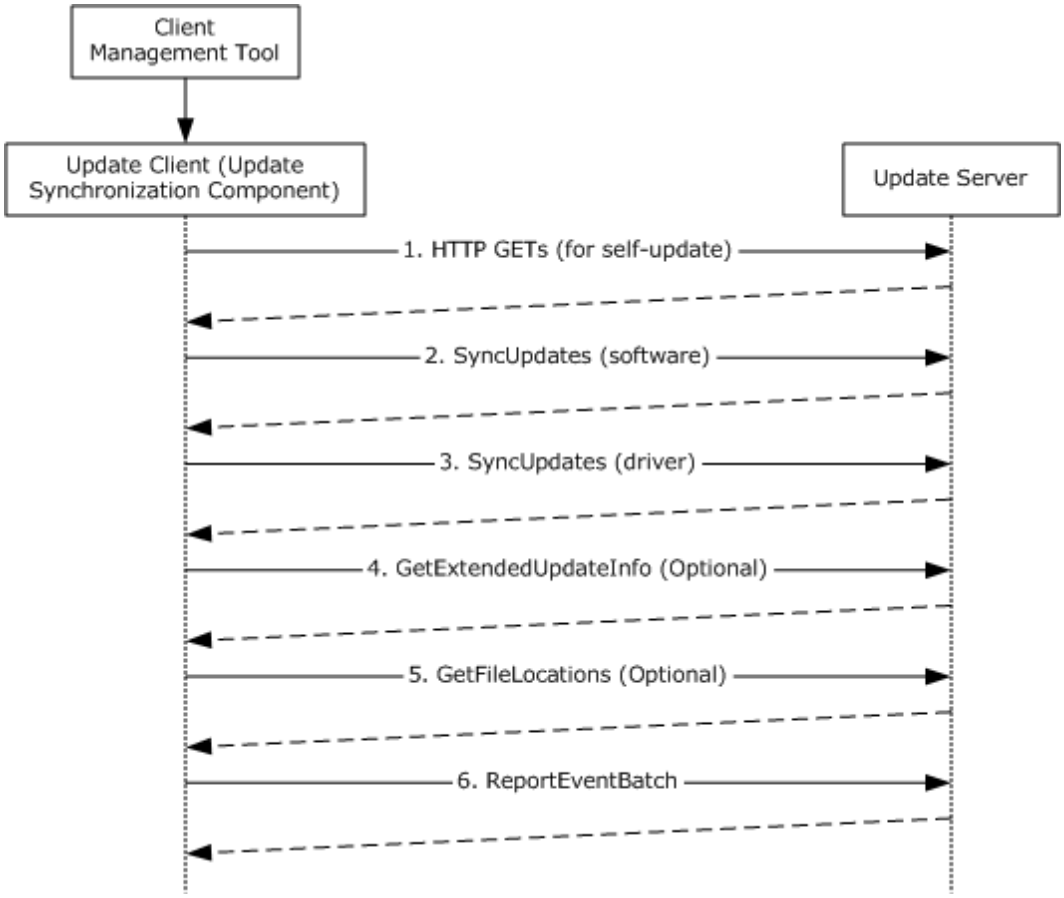

Figure 10: Sequence diagram for differential update sync to client

Assuming that the client has a valid authorization cookie from a previous synchronization, the message flow shown in the previous figure is as follows:

The update client makes a series of HTTP GET requests to determine the availability of applicable update client software. If so, it downloads the software using HTTP GET requests and updates itself. A differential update sync is less likely to find applicable update client software if a previous update sync was done recently.

The update client sends a SyncUpdates message for software updates to the update server, which responds with update information. There can be multiple iterations of this message and response. The message details are specified in [MS-WUSP] section 3.1.5.7.

The update client sends a SyncUpdates message for drivers to the update server. The message details are specified in [MS-WUSP] section 3.1.5.7.

If updates are determined to be applicable, the update client sends a GetExtendedUpdateInfo message to the server, which responds with extended update information. The message details are specified in [MS-WUSP] section 3.1.5.9.

If there's a new update, the message GetFileLocations will be sent to get the new FileLocation information.The message details are specified in [MS-WUSP] section 3.1.5.10.

The update client sends a ReportEventBatch message to the update server, which responds with the status. The message details are specified in [MS-WUSP] section 3.1.5.11.

6.1.5 Rollup of Reporting Data to USS

In this example, the goal of the scenario is for the DSS to send update installation and applicability information about the clients and descendent DSSs to the USS. This supports the use case described in section 3.3.4.4 (Monitor Update Installation -WSUS Administrator) by allowing the USS to collect information about all the update servers in a hierarchy when they are configured appropriately.

The conditions under which this scenario can be initiated are specified in [MS-WSUSSS] section 3.2.4.5.

The following sequence diagram shows the messages exchanged between the DSS and USS during this scenario when the DSS is configured as a replica server and the USS configuration indicates that detailed rollup is needed.

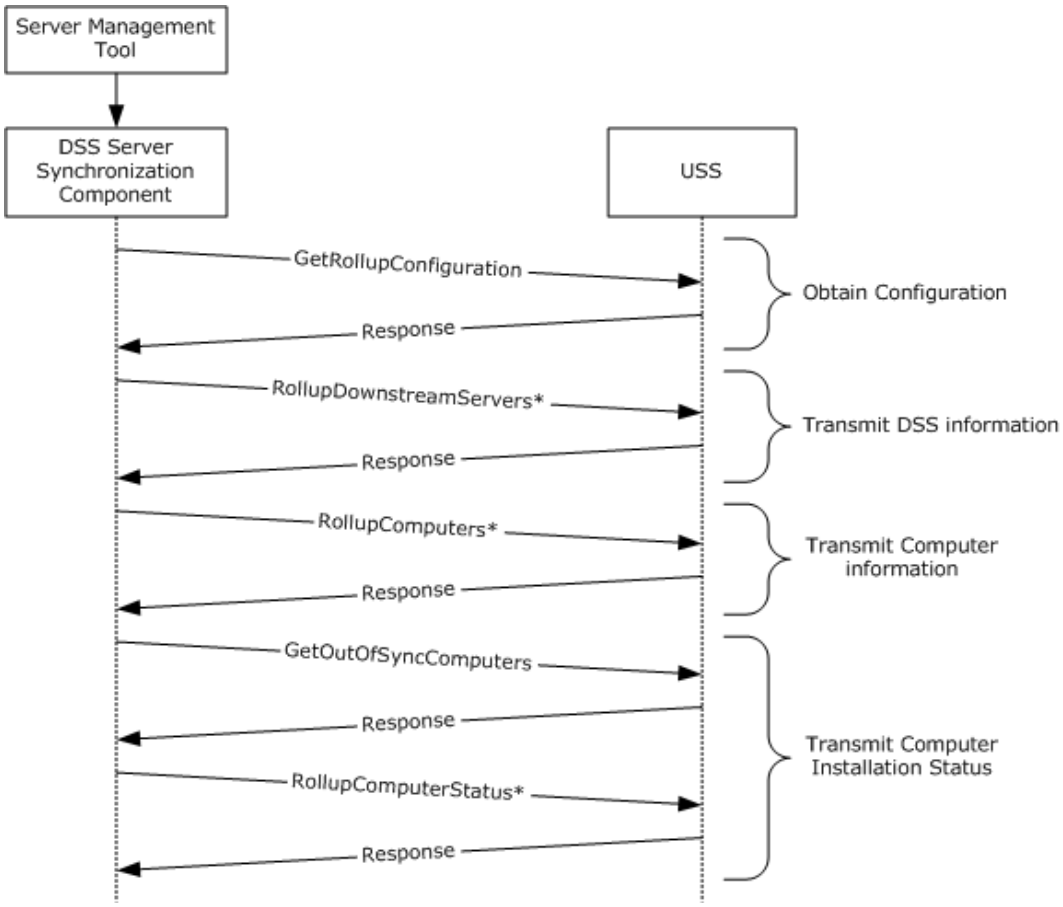

Figure 11: Sequence diagram for rollup of reporting data to USS

Reporting Data Synchronization specifies message processing and sequencing for the messages specified in [MS-WSUSSS] section 3.2.4.5. As the amount of data to be sent from a DSS to USS as part of this scenario can be large, some of the messages are required to be put into batches by the DSS in order not to overwhelm the USS.

6.1.6 Update Client is Pointed to a New Update Server

In this example, the goal of the scenario is for a particular Update Client to synchronize update metadata and deployments from a different Update Server than it used for its previous synchronization. This is part of the Configure Update Client and Start Update Scan use cases. This scenario supports the use case in section 3.3.4.7 (Start Update Scan - Computer User).

This scenario can be initiated by a WSUS Administrator by means of a management system (such as described in [MS-GPSO]) or by a user by means of a Client Management Tool. After this scenario occurs successfully, the update client has synchronized update metadata and deployment from the new Update Server and purged cached data from the old update server.

The following sequence diagram illustrates the interactions between parts of the system during this scenario. (Note that in this diagram, the Update Client begins by attempting a differential update sync and then realizes that its update server has changed.)

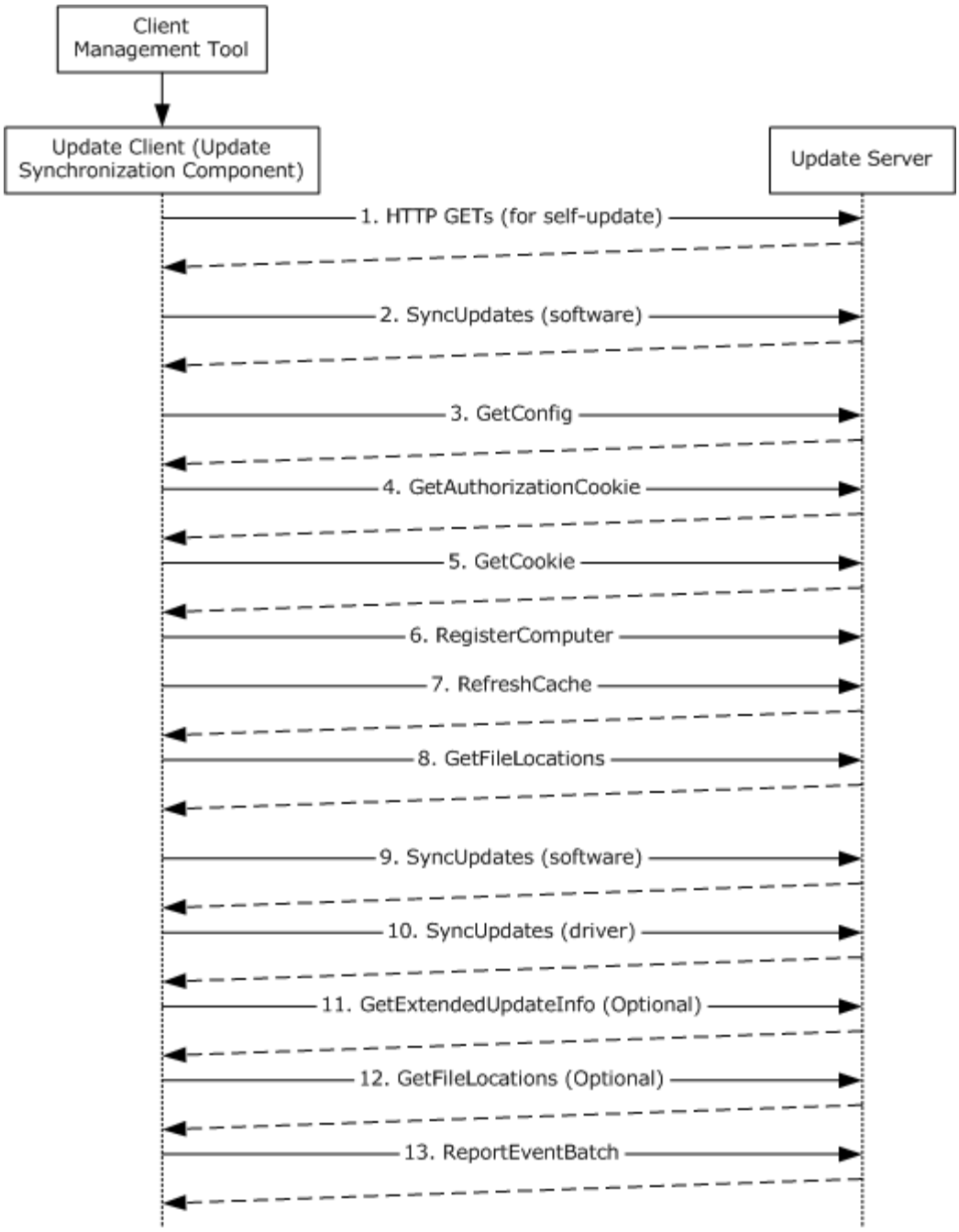

Figure 12: Sequence diagram for pointing update client to new update server

The message flow shown in the previous figure is as follows:

The update client makes a series of HTTP GET requests to determine the availability of applicable update client software. If the software is available, it downloads the software using HTTP GET requests and updates itself.

The update client sends a SyncUpdates message [MS-WUSP] section 3.1.5.7 for software updates to the update server. The update server detects that the client's last sync was against a different server and returns a ServerChanged fault. The fault details are specified in [MS-WUSP] section 2.2.2.4

The update client sends a GetConfig message to the update server, which responds with configuration data. The message details are specified in [MS-WUSP] section 3.1.5.2.

The update client sends a GetAuthorizationCookie message to the update server, which responds with an authorization cookie. The message details are specified in [MS-WUSP] section 3.1.5.3.

The update client sends a GetCookie message to the update server, which responds with a cookie. The message details are specified in [MS-WUSP] section 3.1.5.4.

The update client conditionally sends a RegisterComputer message to the update server. The message details and the conditions under which it is utilized are specified in [MS-WUSP] section 3.1.5.5.

The update client sends a RefreshCache message to the update server, which responds with update identifiers. The message details are specified in [MS-WUSP] section 3.1.5.8.

The update client sends a GetFileLocations message to the update server, which responds with file locations. The message details are specified in [MS-WUSP] section 3.1.5.10.

The update client sends a SyncUpdates message for software updates to the update server, which responds with update information. There can be multiple iterations of this message and response. The message details are specified in [MS-WUSP] section 3.1.5.7.

The update client sends a SyncUpdates message for drivers to the update server. The message details are specified in [MS-WUSP] section 3.1.5.7.

If updates are determined to be applicable, the update client sends a GetExtendedUpdateInfo message to the server, which responds with extended update information. The message details are specified in [MS-WUSP] section 3.1.5.9.

If there is a new update, the message GetFileLocations will be sent to get the new FileLocation information.The message details are specified in [MS-WUSP] section 3.1.5.10.

The update client sends a ReportEventBatch message to the update server, which responds with a status. The message details are specified in [MS-WUSP] section 3.1.5.11.

6.2 Communication Details

The system does not define any constraints on the communication between components of the system beyond those described in the specifications of the member protocols, as listed in section 2.2.

6.3 Transport Requirements

The system does not impose any transport requirements beyond those described in the specifications of the member protocols, as listed in section 2.2.

6.4 Timers

An update server SHOULD use a timer to trigger periodic synchronization with its USS. The frequency of the timer is implementation-specific. The synchronization time and frequency SHOULD

be configurable such that the WSUS Administrator can set a schedule based on the administrative schedule, requirements on the timeliness of the updates, and the topology of the system deployed in an environment as defined in [MS-WSUSSS] section 3.2.3. The periodic synchronization of the update server with its USS impacts update clients synchronizing with the update server due to the relationships between the protocols described in section 5.2.1.

Similarly, an update client SHOULD use a timer to trigger periodic reporting to its update server. The frequency of the timer is implementation-specific. The periodic reporting of the update client to its update server impacts the update server's USS due to the relationship between the protocols described in section 5.2.1.<1>

6.5 Non-Timer Events

There are no non-timer events with system-level significance.

6.6 Initialization and Reinitialization Procedures

Initialization of the system entails the following:

- Update servers generate a **globally unique identifier** to identify themselves to other update servers.
- Update clients generate a globally unique identifier to identify themselves to update servers.
- Update servers are initialized with the location (for example, DNS name and port) of their upstream server (USS).
- Update clients are initialized with the location (for example, DNS name and port) of their update server.

An update server or update client can be individually reinitialized without reinitializing the entire system.

6.7 Status and Error Returns

The system does not define any errors beyond those described in the specifications of the member protocols, as listed in section 2.2.

Section 3 of the member protocol specifications describes the errors relevant to each protocol.

7 Security

This section documents system-wide security issues that are not otherwise described in the Technical Documents (TDs) for the Member Protocols. It does not duplicate what is already in the Member Protocol TDs unless there is some unique aspect that applies to the system as a whole.

The following high-level threats must be considered when implementing this system:

- **Man-in-the-middle** attacks: Update metadata and administrative intent can be tampered with by man-in-the-middle attacks to deny availability of critical or security updates to DSSs or computers. Due to this possibility, it is strongly recommended that the communication channel used by the member protocols is secured using SSL.
- Spoofing: The system identifies computers and DSSs using globally unique identifiers. Given those identifiers, a malicious user can spoof reporting data for DSSs and computers in a way to mask the actual health of the computers. Due to this possibility, it is recommended that the communication channel between update servers, and update client and update server use some form of authentication to prevent rogue entities from masquerading as valid clients or servers.

8 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft Windows® 2000 operating system
- Windows® XP operating system
- Windows® XP operating system Service Pack 1 (SP1)
- Windows® XP operating system Service Pack 2 (SP2)
- Windows® XP operating system Service Pack 3 (SP3)
- Windows Server® 2003 operating system
- Windows Server® 2003 R2 operating system
- Windows Server® 2003 operating system with Service Pack 1 (SP1)
- Windows Server® 2003 operating system with Service Pack 2 (SP2)
- Windows Vista® operating system
- Windows Vista® operating system with Service Pack 1 (SP1)
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 6.4: The Update Client periodically reports to its update server with a randomly chosen frequency of between once every minute and once every fifteen minutes.

9 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

10 Index

S

T

U

V

W