[MS-WSRM-Diff]:

Windows System Resource Manager (WSRM) Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards ~~as well as overviews of the interaction among each of these technologies~~support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you ~~may~~can make copies of it in order to develop implementations of the technologies that are described in ~~the Open Specifications~~ this documentation and ~~may~~can distribute portions of it in your implementations ~~using~~that use these technologies or in your documentation as necessary to properly document the implementation. You ~~may~~can also distribute in your implementation, with or without modification, any ~~schema, IDL's~~schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications~~.~~ documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that ~~may~~might cover your implementations of the technologies described in the Open Specifications~~.~~ documentation. Neither this notice nor Microsoft's delivery of ~~the~~this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open ~~Specification may~~Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in ~~the Open Specifications~~this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks**. The names of companies and products contained in this documentation ~~may~~might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, ~~e-mail~~email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications ~~do~~documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available ~~standard~~standards specifications and network programming art~~,~~ and ~~assumes~~, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
1/25/2008	0.1	Major	MCPP RSAT Initial Availability
3/14/2008	1.0	Major	XSD schema element descriptions, Boolean ABNF data format added; glossary pass performed.
5/16/2008	1.0.1	Editorial	Changed language and formatting in the technical content.
6/20/2008	1.0.2	Editorial	Changed language and formatting in the technical content.
7/25/2008	1.0.3	Editorial	Changed language and formatting in the technical content.
8/29/2008	1.0.4	Editorial	Changed language and formatting in the technical content.
10/24/2008	1.0.5	Editorial	Changed language and formatting in the technical content.
12/5/2008	2.0	Major	Updated and revised the technical content.
1/16/2009	3.0	Major	Updated and revised the technical content.
2/27/2009	4.0	Major	Updated and revised the technical content.
4/10/2009	5.0	Major	Updated and revised the technical content.
5/22/2009	5.1	Minor	Clarified the meaning of the technical content.
7/2/2009	5.1.1	Editorial	Changed language and formatting in the technical content.
8/14/2009	5.2	Minor	Clarified the meaning of the technical content.
9/25/2009	5.3	Minor	Clarified the meaning of the technical content.
11/6/2009	5.3.1	Editorial	Changed language and formatting in the technical content.
12/18/2009	5.4	Minor	Clarified the meaning of the technical content.
1/29/2010	6.0	Major	Updated and revised the technical content.
3/12/2010	6.1	Minor	Clarified the meaning of the technical content.
4/23/2010	7.0	Major	Updated and revised the technical content.
6/4/2010	7.0.1	Editorial	Changed language and formatting in the technical content.
7/16/2010	8.0	Major	Updated and revised the technical content.
8/27/2010	8.1	Minor	Clarified the meaning of the technical content.
10/8/2010	9.0	Major	Updated and revised the technical content.
11/19/2010	9.1	Minor	Clarified the meaning of the technical content.
1/7/2011	10.0	Major	Updated and revised the technical content.
2/11/2011	11.0	Major	Updated and revised the technical content.
3/25/2011	11.0	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	11.0	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
6/17/2011	11.1	Minor	Clarified the meaning of the technical content.
9/23/2011	12.0	Major	Updated and revised the technical content.
12/16/2011	13.0	Major	Updated and revised the technical content.
3/30/2012	14.0	Major	Updated and revised the technical content.
7/12/2012	14.1	Minor	Clarified the meaning of the technical content.
10/25/2012	14.2	Minor	Clarified the meaning of the technical content.
1/31/2013	14.2	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	14.2	None	No changes to the meaning, language, or formatting of the technical content.
11/14/2013	14.2	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	14.2	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	14.2	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	14.2	~~No Change~~None	No changes to the meaning, language, or formatting of the technical content.
10/16/2015	14.2	~~No Change~~None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This is a specification of the Windows System Resource Manager (WSRM) Protocol. It is based on the **Remote Procedure Call (RPC)** Protocol [C706] [MS-RPCE].

WSRM exposes a set of **Distributed Component Object Model (DCOM) Remote Protocol** interfaces for managing processor and memory resources and **accounting** functions on a computer. Using these interfaces, the following operations can be performed:

- Create, enumerate, modify, or delete rules governing resource consumption.
- Retrieve, store, or delete long-term, persistent resource consumption histories.
- Directly control **resource management** enforcement.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative ~~and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms~~. All other sections and examples in this specification are informative.

1.1 Glossary

~~The~~This document uses the following terms ~~are specific to this document~~:

accounting: Information gathered and maintained by the **management service** about the runtime behavior of processes. The **management service** provides an accounting state switch with two settings: enabled and disabled. When enabled, accounting information is gathered and persisted across invocations of the **management service**. Accounting information gathered by the **management service** on one computer can be persisted by the **management service** on a different computer. When the accounting state is disabled, no accounting data is gathered or persisted.

accounting client: A WSRM server whose **accounting** data is being logged on another WSRM server.

accounting process: Process whose properties are stored in the WSRM **accounting** database.

accounting server: A WSRM server that is logging the **accounting** data of another WSRM server.

Active Directory: A general-purpose network directory service. **Active Directory** also refers to the Windows implementation of a directory service. **Active Directory** stores information about a variety of objects in the network. Importantly, user accounts, computer accounts, groups, and all related credential information used by the Windows implementation of Kerberos are stored in **Active Directory**. **Active Directory** is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS). [MS-ADTS] describes both forms. For more information, see [MS-AUTHSOD] section 1.1.1.5.2, Lightweight Directory Access Protocol (LDAP) versions 2 and 3, Kerberos, and DNS.

application pool: A grouping of one or more URLs or websites served by an implementation of the core web server in IIS.

Augmented Backus-Naur Form (ABNF): A modified version of Backus-Naur Form (BNF), commonly used by Internet specifications. ABNF notation balances compactness and simplicity with reasonable representational power. ABNF differs from standard BNF in its definitions and uses of naming rules, repetition, alternatives, order-independence, and value ranges. For more information, see [RFC5234].

authentication level: A numeric value indicating the level of authentication or message protection that **remote procedure call (RPC)** will apply to a specific message exchange. For more information, see [C706] section 13.1.2.1 and [MS-RPCE].

calendar: A method of controlling which **resource allocation policy (RAP)** is selected as the **current resource policy**. The calendar maintains start and end dates and times for **RAP** and is either enabled or disabled. When enabled, the **management service** continuously monitors start and end dates and times of the scheduled **RAP** to activate the correct **current resource policy**. When disabled, the **RAP** scheduled on the calendar has no effect on which **RAP** is the **current resource policy**.

committed memory: The number of bytes that have been allocated by processes, and to which the operating system has committed a RAM page frame or a page slot in the page file (or both).

Component Object Model (COM): An object-oriented programming model that defines how objects interact within a single process or between processes. In **COM**, clients have access to an object through interfaces implemented on the object. For more information, see [MS-DCOM].

condition: A method of controlling which **RAP** is selected as the **current resource policy**. Conditions are rules that are automatically triggered in response to notifications of any of the **conditional events**. A condition is composed of a **condition state** and **RAP**. When a **conditional event** is triggered, conditions with the associated Name attribute value are evaluated in the order of their ID attribute value; that is, a condition with the ID value 0 will be evaluated first and so on. In condition evaluation, the **condition state** is evaluated and if it is found to be TRUE, the **RAP** associated with that condition is selected as the **current resource policy**. If no condition has its **condition state** as TRUE, the condition with the name ANY is evaluated.

condition state: A part of a **condition** consisting of a predicate that evaluates some current state of the computer being managed. The predicate is a series of expressions separated by AND and OR operators, evaluated in order. Expressions are selected from the following fixed set: an equality or inequality test of the amount of hardware memory, an equality or inequality test of the number of processors, or a predicate test of the online or offline status of a cluster node or cluster resource group.

conditional events: Unscheduled events that can trigger the following WSRM policy changes: Processor hot add, Memory hot add, Cluster node goes up or down, or Cluster resource group goes online or offline.

conflicting objects: Objects, one being imported and one in the current **WSRM configuration**, which match in name and type but differ in content.

current resource policy: While in the running **management state**, the **management service** always selects exactly one **RAP** to be the **current resource policy**.

Distributed Component Object Model (DCOM): The Microsoft Component Object Model (COM) specification that defines how components communicate over networks, as specified in [MS-DCOM].

domain: A set of users and computers sharing a common namespace and management infrastructure. At least one computer member of the set must act as a domain controller (DC) and host a member list that identifies all members of the domain, as well as optionally hosting the **Active Directory** service. The domain controller provides authentication (2) of members, creating a unit of trust for its members. Each domain has an identifier that is shared among its members. For more information, see [MS-AUTHSOD] section 1.1.1.5 and [MS-ADTS].

domain name: A **domain name** or a NetBIOS name that identifies a **domain**.

dynamic fair-share scheduling (DFSS): A scheduling algorithm that is fair toward every process in the system.

event identifier: A numerical value used to identify an event which can occur at the WSRM server.

exclusion list: A list of processes that cannot be managed because of the negative system impact such management could create. Processes in an exclusion list are unmanaged and can consume resources freely. Both system-defined and user-defined exclusion lists are defined.

fully qualified domain name (FQDN): An unambiguous **domain name** that gives an absolute location in the Domain Name System's (DNS) hierarchy tree, as defined in [RFC1035] section 3.1 and [RFC2181] section 11.

Group Policy: A mechanism that allows the implementer to specify managed configurations for users and computers in an **Active Directory** service environment.

HRESULT: An integer value that indicates the result or status of an operation. A particular HRESULT can have different meanings depending on the protocol using it. See [MS-ERREF] section 2.1 and specific protocol documents for further details.

Interface Definition Language (IDL): The International Standards Organization (ISO) standard language for specifying the interface for remote procedure calls. For more information, see [C706] section 4.

Internet Information Services (IIS): The services provided in Windows implementation that support web server functionality. **IIS** consists of a collection of standard Internet protocol servers such as HTTP and FTP in addition to common infrastructures that are used by other Microsoft Internet protocol servers such as SMTP, NNTP, and so on. **IIS** has been part of the Windows operating system in some versions and a separate install package in others. **IIS** version 5.0 shipped as part of Windows 2000 operating system, **IIS** version 5.1 as part of Windows XP operating system, **IIS** version 6.0 as part of Windows Server 2003 operating system, and **IIS** version 7.0 as part of Windows Vista operating system and Windows Server 2008 operating system.

little-endian: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

machine group: A grouping of WSRM servers. Pre-authored policy can be exported to the servers in a machine group simultaneously.

management client: An application that uses the WSRM Protocol interfaces for the purpose of presenting a user interface that allows a user to perform the functions exposed by the WSRM Protocol.

management service: An agent that implements the WSRM Protocol on a given computer by applying specified resource policies, returning requested **accounting** information, and storing the **accounting** data dumped by the other **management services** running on remote servers.

management state: A state switch with two values, running and stopped, that tells whether the **management service** can be active or inactive. "Running" means the service will perform all **resource management** and **accounting** functions according to its current policies. "Stopped" means that it will remain in an inactive state, doing nothing except making configuration changes that will take effect when the **management service** becomes active again; for example, import, export, creation, deletion, or modification of **resource allocation policy**.

NetBIOS: A particular network transport that is part of the LAN Manager protocol suite. **NetBIOS** uses a broadcast communication style that was applicable to early segmented local area networks. The LAN Manager protocols were the default in Windows NT operating system environments prior to Windows 2000. A protocol family including name resolution, datagram, and connection services. For more information, see [RFC1001] and [RFC1002].

process matching criteria (PMC): A **resource policy object** that selects a subset of currently executing processes. Since processes are dynamically created and terminated by the operating system in the course of running workloads, the WSRM Protocol uses PMCs as a means of identifying processes for **resource management** purposes. PMCs specify partial or full values to be matched against process property fields. Each PMC includes a name and a nonempty set of matching values and can also include a nonempty set of exclusion values. All running processes under management whose path and the associated user name match the values provided in a PMC are selected by that PMC, provided that they are not already selected by another PMC and do not match the exclusion values. Processes selected by a PMC specification at any given time are said to match, or be in, the PMC. A process can be selected by only one PMC at a time. The term **resource group**" and PMC are used interchangeably.

processor affinity: An element of **process matching criteria (PMC)**, the association between a task or process and a specific processor needed to execute that task. Processor affinity takes advantage of the fact that some remnants of a process might remain in one processor's state (in particular, in its cache) from the last time the process ran, and so scheduling it to run on the same processor the next time could make the process run more efficiently than if it were to run on another processor.

registry: A local system-defined database in which applications and system components store and retrieve configuration data. It is a hierarchical data store with lightly typed elements that are logically stored in tree format. Applications use the registry API to retrieve, modify, or delete registry data. The data stored in the registry varies according to the version of Windows.

Remote Administration Protocol (RAP): A synchronous request/response protocol, used prior to the development of the remote procedure call (RPC) protocol, for marshaling and unmarshaling procedure call input and output arguments into messages and for reliably transporting messages to and from clients and servers.

remote procedure call (RPC): A context-dependent term commonly overloaded with three meanings. Note that much of the industry literature concerning RPC technologies uses this term interchangeably for any of the three meanings. Following are the three definitions: (*) The runtime environment providing remote procedure call facilities. The preferred usage for this meaning is "RPC runtime". (*) The pattern of request and response message exchange between two parties (typically, a client and a server). The preferred usage for this meaning is "RPC exchange". (*) A single message from an exchange as defined in the previous definition. The preferred usage for this term is "RPC message". For more information about RPC, see [C706].

residual PMC: The residual PMC matches all of the processes that are not included in the **exclusion list** or do not match another **process matching criteria**. The residual **PMC** gets an unallocated percentage of CPU in a **RAP** that can never be less than 1 percent. Other **resource allocation**, such as a **processor affinity** mask, an amount of physical memory, or an amount of virtual memory, do not apply to residual **PMC**.

resource allocation: The part of a **RAP** that specifies the part of a computer's hardware resources that can be allocated to the processes in a **PMC**. A resource allocation can specify any of the following: a percentage of processor bandwidth, a **processor affinity** mask, an amount of physical memory, or an amount of virtual memory. A resource allocation includes a specification of at least one resource.

resource allocation policy (RAP): A named specification for allocating computer resources to all of the managed processes on a computer. A **RAP** specifies how the managed resources of a computer can be divided among managed processes running on the computer by providing an ordered list of **PMC** names, each with an associated **resource allocation**. All processes not matched by any **PMC** named in the list and not included in the **exclusion list** are selected into an implicit "residual" **PMC**.

resource group: Used interchangeably with **process matching criteria (PMC)** in Windows System Resource Manager (WSRM) Protocol [MS-WSRM].

resource management: A method of allocating the hardware resources of a computer to tasks being performed on that computer. It includes a system of **accounting** for hardware resources by task. The purpose of the WSRM Protocol is to control the resource management of a computer.

resource policy: A declarative **resource allocation** rule set used by the WSRM Protocol to specify the desired **resource management** behavior of a computer.

resource policy object: A persistent object that is maintained by the **management service**, created by a **management client**, or built-in to the **management service**. **Resource policy** objects specify the desired **resource management** behavior of the computer whose resources are under management.

schema: The set of attributes and object classes that govern the creation and update of objects.

schema-validity assessment: The determination of whether the name and namespace of an information element matches a definition in a specified **XML schema**.

service: A process or agent that is available on the network, offering resources or services for clients. Examples of services include file servers, web servers, and so on.

session identifier: Unique identifier that an operating system generates when a session is created. A session spans the period of time from logon until logoff from a specific system.

Simple Mail Transfer Protocol (SMTP): A member of the TCP/IP suite of protocols that is used to transport Internet messages, as described in [RFC5321].

smart import-conflict resolution: An import of **conflicting objects** in which one of the conflicting objects is assigned a new unique name to resolve the conflict. All references to the renamed object are updated accordingly.

terminal services (TS): A service on a server computer that allows delivery of applications, or the desktop itself, to various computing devices. When a user runs an application on a terminal server, the application execution takes place on the server computer and only keyboard, mouse, and display information is transmitted over the network. Each user sees only his or her individual session, which is managed transparently by the server operating system and is independent of any other client session.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

universally unique identifier (UUID): A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and **RPC** objects. UUIDs are highly likely to be unique. UUIDs are also known as globally unique identifiers (GUIDs) and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the UUID.

Windows Internal Database: Built-in relational database component of Windows Server 2008 for use by other Windows components.

workgroup: A collection of computers that share a name. In the absence of a **domain**, a workgroup allows a convenient means for browser clients to limit the scope of a search.

working set: The set of memory pages recently touched by the threads of a process. If free memory in the computer is above a threshold, pages are left in the working set of a process even if they are not being used. When free memory falls below a threshold, pages are trimmed from the working set.

WSRM configuration: The current **PMC**, **RAP**, **calendar** events, and **conditions** of the WSRM system.

WSRM management service: See **management service**.

XML: The Extensible Markup Language, as described in [XML1.0].

XML schema definition (XSD): The World Wide Web Consortium (W3C) standard language that is used in defining XML schemas. Schemas are useful for enforcing structure and constraining the types of data that can be used validly within other XML documents. XML schema definition refers to the fully specified and currently recommended standard for use in authoring XML schemas.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, https://www2.opengroup.org/ogsys/catalog/c706

[MS-ADA3] Microsoft Corporation, "Active Directory Schema Attributes N-Z".

[MS-CMRP] Microsoft Corporation, "Failover Cluster: Management API (ClusAPI) Protocol".

[MS-DCOM] Microsoft Corporation, "Distributed Component Object Model (DCOM) Remote Protocol".

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

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

[MS-OAUT] Microsoft Corporation, "OLE Automation Protocol".

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions".

[MS-SAMR] Microsoft Corporation, "Security Account Manager (SAM) Remote Protocol (Client-to-Server)".

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

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

[UNICODE] The Unicode Consortium, "The Unicode Consortium Home Page", 2006, http://www.unicode.org/

[W3C-XSD] World Wide Web Consortium, "XML Schema Part 2: Datatypes Second Edition", October 2004, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028

[XML1.0] Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E., "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, October 2000, http://www.w3.org/TR/2000/REC-xml-20001006

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1/2] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures Second Edition", W3C Recommendation, October 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

1.2.2 Informative References

[MSDN-WSRM1] Microsoft Corporation, "Windows System Resource Manager, Windows Server 2008 R2", http://technet.microsoft.com/en-us/library/cc755056.aspx

1.3 Overview

The Windows System Resource Manager (WSRM) Protocol provides tools for managing processor and memory resources on a computer. With WSRM, administrators can control how CPU resources are allocated to applications, services, and processes, and prevent applications from consuming more than their share, thereby preventing one application from starving other applications of CPU resources and memory.

WSRM policies can be applied according to a date/time schedule. This allows administrators to free up the CPU and memory for maintenance applications during nonpeak hours and for mission-critical applications during peak hours.

The WSRM accounting feature allows administrators to generate, store, view, and export resource utilization reports for systems management, as well as service-level agreement tracking and billing information.

WSRM exposes a set of DCOM Protocol [MS-DCOM] interfaces that perform the following functions:

- **Resource Management**: Performs administrative tasks and retrieves system information from the computer being managed.
- **Accounting Management**: Performs configuration tasks on accounting data, including the following:
  - Specifying the database server.
  - Changing the location of database files.
  - Controlling how often the server creates accounting records.
  - Managing accounting data records.
- **Calendar Management**: Schedules resource management operations.

- **Configuration**: Manages the global configuration of server resources.
- **Machine Group Management**: Creates, manages, and deletes **machine groups**.
- **Object Management**: Imports and exports objects with **smart import-conflict resolution**.
- **Policy Management**: Creates, edits, enumerates, and deletes objects that control the allocation of resources to user tasks.
- **Protocol Management**: Manages client versions supported on the server.
- **Remote Session Management**: Manages CPU quota allocation to different remote sessions.
- **Resource Group Management**: Retrieves, modifies, and deletes **resource groups**.

1.4 Relationship to Other Protocols

The WSRM Protocol depends on the DCOM Protocol [MS-DCOM], which uses remote procedure call (RPC) ([C706] and [MS-RPCE]) as its transport.

There are no protocols that depend on the WSRM Protocol. The WSRM Protocol ~~may~~can be used by applications directly.

1.5 Prerequisites/Preconditions

The WSRM Protocol requires a working, correctly configured DCOM Protocol [MS-DCOM] infrastructure.

1.6 Applicability Statement

The WSRM Protocol is applicable when a programmatic interface is required that allows a user to remotely limit and monitor the allocation of resources to tasks running on a computer. The WSRM Protocol can be used where **Terminal Services** is installed to manage multiple applications on a single computer or multiple users connecting to a server.

Managing resources with the WSRM Protocol can improve system performance and reduce the chance that applications, services, or processes will interfere with the rest of the system. The WSRM Protocol creates a more consistent and predictable experience for users of applications and services running on the computer.

1.7 Versioning and Capability Negotiation

This specification covers versioning issues in the following areas:

Supported Transports: The WSRM Protocol supports the transports supported by the DCOM Protocol [MS-DCOM].

Protocol Versions: The following table shows the DCOM interfaces that are defined in the WSRM Protocol and their versions.

Interface	Version
IResourceManager	1.0
IResourceManager2	1.0
IWRMAccounting	1.0
IWRMCalendar	1.0

Interface	Version
IWRMConfig	1.0
IWRMMachineGroup	1.0
IWRMPolicy	1.0
IWRMProtocol	1.0
IWRMRemoteSessionMgmt	1.0
IWRMResourceGroup	1.0

Security and Authentication Methods: The WSRM Protocol relies upon the underlying RPC Protocol to obtain the identity of users making method calls ([MS-RPCE] section 3.3.3.4.3). The WSRM server uses this identity to perform access checks, as described in Security Considerations for Implementers (section 5.1).

1.8 Vendor-Extensible Fields

The WSRM Protocol uses **HRESULT** method return values as specified in [MS-ERREF]. Vendors MAY<1> specify additional error codes, which MUST have the C bit (0x20000000) set, indicating they are extensions.

A **management client** SHOULD<2> be implemented to provide a user interface for accounting and resource management operations.

1.9 Standards Assignments

The WSRM Protocol uses the following **UUIDs** to uniquely identify its interfaces.

Interface	UUID	Reference
RPC Interface UUID for IResourceManager	C5CEBEE2-9DF5-4CDD-A08C-C2471BC144B4	As specified in [C706], Appendix A; for more information, see section 3.2.4.1.
RPC Interface UUID for IResourceManager2	2A3EB639-D134-422d-90D8-AAA1B5216202	As specified in [C706], Appendix A; for more information, see section 3.2.4.2.
RPC Interface UUID for IWRMAccounting	4F7CA01C-A9E5-45B6-B142-2332A1339C1D	As specified in [C706], Appendix A; for more information, see section 3.2.4.3.
RPC Interface UUID for IWRMCalendar	481E06CF-AB04-4498-8FFE-124A0A34296D	As specified in [C706], Appendix A; for more information, see section 3.2.4.4.
RPC Interface UUID for IWRMConfig	21546AE8-4DA5-445E-987F-627FEA39C5E8	As specified in [C706], Appendix A; for more information, see section 3.2.4.5.
RPC Interface UUID for IWRMMachineGroup	943991A5-B3FE-41FA-9696-7F7B656EE34B	As specified in [C706], Appendix A; for more information, see section 3.2.4.6.
RPC Interface UUID for IWRMPolicy	59602EB6-57B0-4FD8-AA4B-EBF06971FE15	As specified in [C706], Appendix A; for more information, see section 3.2.4.7.
RPC Interface UUID for IWRMProtocol	F31931A9-832D-481C-9503-887A0E6A79F0	As specified in [C706], Appendix A; for more information, see section 3.2.4.8.
RPC Interface UUID for IWRMRemoteSessionMgmt	FC910418-55CA-45EF-B264-83D4CE7D30E0	As specified in [C706], Appendix A; for more information, see section 3.2.4.9.

Interface	UUID	Reference
RPC Interface UUID for IWRMResourceGroup	BC681469-9DD9-4BF4-9B3D-709F69EFE431	As specified in [C706], Appendix A; for more information, see section 3.2.4.10.

2 Messages

2.1 Transport

Message transport in the Windows System Resource Manager (WSRM) Protocol uses the DCOM Protocol [MS-DCOM], which uses Remote Procedure Call (RPC) ([C706] and [MS-RPCE]) as its transport.

The WSRM server requires clients to request an RPC **authentication level** of **RPC_C_AUTHN_LEVEL_PKT_PRIVACY** (see [MS-RPCE] section 2.2.1.1.8) for method calls to protect the privacy of client-server communication.

2.2 Common Data Types

This section specifies common data types in the WSRM Protocol. These data types are used in messages and method parameters.

Name	Description
ABNF data formats	Specifies **Augmented Backus-Naur Form (ABNF)** syntax for string data.
Constants	Specifies literal constants.
Enumerations	Specifies enumeration constants.
XML data formats	Specifies XSD **schemas** for **XML** string data.

Note Unless otherwise indicated, the internal representation of all numerical values is in **little-endian** order.

2.2.1 ABNF Data Formats

The WSRM Protocol specifies Augmented Backus-Naur Form (ABNF) data formats for strings that are used in messages and method parameters. These formats are defined using syntax specified in the ABNF standard [RFC5234]. Basic rules that are defined in the standard appear in upper case. Literal alphabetic characters in ABNF are case-insensitive unless specified otherwise.

The following table lists the ABNF data formats that are defined in this section.

Name	Description
Affinity	Specifies the format of **processor affinity** values.
Boolean	Specifies the format of Boolean values.
Date and Time	Specifies date and time values in various formats.
Timestamp	Specifies the format of timestamp values.

2.2.1.1 Affinity

This section specifies the ABNF rules for affinity values, which are used to specify processors on a server. Processor affinity values can be specified in strings in WSRM Protocol method parameters and XML data elements.

```
cpu-number        = 1*DIGIT
cpu-range         = cpu-number "-" cpu-number

cpu-sequence-elem = cpu-number / cpu-range
cpu-sequence      = cpu-sequence-elem *("," cpu-sequence-elem)

affinity          = cpu-sequence / "all"
```

2.2.1.2 Boolean

This section specifies the ABNF rules for Boolean values. These values can be specified in strings in WSRM Protocol method parameters and XML data elements.

```
boolean-true    = "true"
boolean-false   = "false"

boolean         = boolean-true / boolean-false
```

2.2.1.3 Date and Time

This section specifies the ABNF rules for date and time values. These values can be specified in strings in WSRM Protocol method parameters and XML data elements.

```
date-fullyear    = 4DIGIT
date-month       = 2DIGIT  ; 01-12
date-month-day   = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31 based on
                           ; date-month

date             = date-fullyear "-" date-month "-" date-month-day

time-24-hour     = 2DIGIT  ; 00-23

time-minute      = 2DIGIT  ; 00-59
time-second      = 2DIGIT  ; 00-58, 00-59, 00-60 based on leap
                           ; second rules

time-secfrac     = 3DIGIT

time             = time-24-hour ":" time-minute ":" time-second

time-and-secfraq = time-24-hour ":" time-minute ":" time-second "."
                   time-secfrac

date-colon-time  = date-fullyear ":" date-month ":" date-month-day ":"
                   time-24-hour ":" time-minute ":" time-second ":"
                   time-secfrac

date-and-time    = date " " time-and-secfraq

date-T-time      = date "T" time
```

date-colon-time values are specified in Machine XML elements (section 2.2.5.17). See the Machine example (section 4.2.14).<3>

date-and-time values are specified in parameters to some IWRMAccounting interface (section 3.2.4.3) methods. See the ExecuteAccountingQuery method (section 3.2.4.3.3).

date-T-time values are specified in Events XML elements (section 2.2.5.15). See the Events example (section 4.2.12).

2.2.1.4 Timestamp

This section specifies the ABNF rules for timestamp values. A timestamp can be specified inside a common node at the root level of an XML element.

```
timestamp_high  = 1*DIGIT
timestamp_low   = 1*DIGIT

timestamp       = ":" timestamp_high ":" timestamp_low ":"
```

timestamp_high: A string representation of a decimal integer that corresponds to the value of the **dwHighDateTime** member of a FILETIME structure ([MS-DTYP] section 2.3.3).

timestamp_low: A string representation of a decimal integer that corresponds to the value of the **dwLowDateTime** member of a FILETIME structure.

A **timestamp** value MUST be defined at the root level of Calendar elements (section 2.2.5.7). See the Calendar example (section 4.2.6).

2.2.2 Constants

The WSRM Protocol specifies literal constants that are used in messages and method parameters.

The following table lists the types of literal constants that are defined in this section.

Name	Description
Boolean values	Specify binary decisions.
Category Types	Specify priorities of **resource policies**.
Day Modifiers	Specify a monthly instance of a day of the week.
Day Options	Specify a calendar day of the week for a scheduled event.
Frequency Options	Specify the frequency of occurrence of a scheduled event.
Memory Options	Specify an action to take when a process exceeds its maximum committed memory.
Object Types	Specify types of **conflicting objects**.
Resource Allocation Options	Specify how resources such as the CPU and memory are allocated to processes.
Supported Client Options	Specify the level of support for clients on the WSRM server.
User Types	Specify the type of a user.

2.2.2.1 Boolean Values

This section defines integer constants for Boolean values, which specify binary decisions. These values are used in WSRM protocol method parameters and XML data elements.

Constant/value	Description
FALSE 0	The integer value for "false".
TRUE 1	The integer value for "true".

2.2.2.2 Category Types

This section defines literal constants for category types, which specify priorities of resource policies. These values are used in WSRM protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
Premium "Premium"	Highest priority.
Standard "Standard"	Medium priority.
Basic "Basic"	Lowest priority.

2.2.2.3 Day Modifiers

This section defines literal constants for day modifiers, which specify the monthly instances of a day of the week for a scheduled event. They are intended to be used with day options (section 2.2.2.4). These values are used in WSRM Protocol method parameters and XML data elements.

Constant/value	Description
First 1	The first instance of a day in a month.
Second 2	The second instance of a day in a month.
Third 3	The third instance of a day in a month.
Fourth 4	The fourth instance of a day in a month.
Last -1	The last instance of a day in a month.

2.2.2.4 Day Options

This section defines literal constants for day options, which specify a calendar day of the week for a scheduled event. These values are used in WSRM Protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
Monday "mo"	The event is scheduled for a Monday.
Tuesday "tu"	The event is scheduled for a Tuesday.
Wednesday "we"	The event is scheduled for a Wednesday.
Thursday "th"	The event is scheduled for a Thursday.
Friday "fr"	The event is scheduled for a Friday.
Saturday "sa"	The event is scheduled for a Saturday.
Sunday "su"	The event is scheduled for a Sunday.
Day "day"	The event is scheduled for any day.

2.2.2.5 Frequency Options

This section defines literal constants for frequency options, which specify the frequency of occurrence of a scheduled event. These values are used in WSRM Protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
Daily "daily"	The event happens every day.
Weekly "weekly"	The event happens once a week.
Monthly "monthly"	The event happens once a month.

Constant/value	Description
Yearly "yearly"	The event happens once a year.

2.2.2.6 Memory Options

This section defines literal constants for memory options, which specify an action to take when a process exceeds its maximum committed memory. These values are used in WSRM Protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
LogEvent "LogEvent"	An event is written to the event log.
TerminateApp "TerminateApp"	The process is terminated.

2.2.2.7 Object Types

This section defines literal constants for object types, which specify types of conflicting objects. These values are used in WSRM protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
OBJECT_SELECTION_CRITERIA "PMC"	The object is a **process matching criteria (PMC)** object (section 2.2.5.24).
OBJECT_POLICY "Policy"	The object is a policy object (section 2.2.5.21).
OBJECT_SCHEDULE "Schedule"	The object is a schedule object (section 2.2.5.26).
OBJECT_CALENDAR "Calendar"	The object is a **calendar** object (section 2.2.5.7).

2.2.2.8 Resource Allocation Options

This section defines literal constants for resource allocation options, which specify how resources such as the CPU and memory are allocated to processes. These values are used in WSRM protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
EqualPerIISAppPool "EqualPerIISAppPool"	The resource is allocated in equal shares among members of an **Internet Information Services (IIS) application pool**.<4>
EqualPerProcess "EqualPerProcess"	The resource is allocated in equal shares among processes.
EqualPerSession "EqualPerSession"	The resource is allocated in equal shares among user sessions.<5>
EqualPerUser "EqualPerUser"	The resource is allocated in equal shares among users.
Standard "Standard"	No attempt is made to control how the resource is allocated among processes.

2.2.2.9 Supported Client Options

This section defines literal constants for supported client options, which specify the level of support for clients on the WSRM server. These values are used in WSRM Protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
Version1 "V1"	The level of client support provided by version 1 WSRM servers.<6>
Version2 "V2"	The level of client support provided by version 2 WSRM servers.<7>
Version3 "V3"	The level of client support provided by version 3 WSRM servers.<8>

2.2.2.10 User Types

This section defines literal constants for user types, which specify the type of a user. These values are used in WSRM protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
Group "Group"	A group of users.
Individual "Inidividual"	An individual user.

2.2.2.11 Column Types

This section defines integer constants for Column Types, which specify the type of the columns in accounting databases. These values are used in XML data elements.

Constant/value	Description
CT_INT64 "0"	The integer designating column type as integer number.
CT_DATETIME "1"	The integer designating column type as date-time field.
CT_LONG_TEXT "2"	The integer designating column type as text of maximum length 510 characters.
CT_SHORT_TEXT "3"	The integer designating column type as text of maximum length 255 characters.

2.2.2.12 Condition Category

This section defines literal constants for Condition Category, which specifies the category of a **condition**. These values are used in WSRM protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
Processor "Processor"	Category to represent hot addition of processor or processors event.
Memory "Memory"	Category to represent hot addition of memory event.
MSCS "MSCS"	Category to represent events of: cluster node goes up, cluster node goes down, cluster resource group goes online, or cluster resource group goes offline, as defined in [MS-CMRP] sections 3.1.4.1.18, 3.1.4.1.19, 3.1.4.1.50, and 3.1.4.1.51, respectively.
ANY "ANY"	A special condition category. A condition with this category will be evaluated if no other condition matches for a **conditional event**.

2.2.2.13 Condition Name

This section defines literal constants for Condition Name, which specifies the name of a condition. These values are used in WSRM protocol method parameters and XML data elements.

Note All string values are case-insensitive.

Constant/value	Description
ProcessorNew "ProcessorNew"	Name to represent condition associated with "Processor hot add" conditional event.

Constant/value	Description
MemoryNew "MemoryNew"	Name to represent condition associated with "Memory hot add" conditional event.
MSCSNodeUp "MSCSNodeUp"	Name to represent condition associated with "Cluster node goes up" conditional event.
MSCSNodeDown "MSCSNodeDown"	Name to represent condition associated with "Cluster node goes down" conditional event.
MSCSRGOnline "MSCSRGOnline"	Name to represent condition associated with "Cluster resource group goes online" conditional event.
MSCSRGOffline "MSCSRGOffline"	Name to represent condition associated with "Cluster resource group goes offline" conditional event.
ANY "ANY"	Name to represent condition, which is not associated with any of the conditional event.

2.2.3 Enumerations

The WSRM Protocol specifies enumeration constants that are used in messages and method parameters.

The following table lists the enumerations that are defined in this section.

Name	Description
CONFIGTYPE	Defines types of configuration objects on which operations can be performed.
EXCLUSIONLIST_TYPE	Defines types of server exclusion lists.
IMPORT_TYPE	Defines import modes for handling conflicting objects.
MACHINE_GROUP_MERGE_OPTIONS	Defines options for machine group modification.
MANAGEMENT_TYPE	Defines server management modes.
OBJECT_TYPE	Defines types of objects on which operations can be performed.
RESTORE_MODE	Defines states that objects can have when they are restored.

2.2.3.1 CONFIGTYPE Enumeration

The CONFIGTYPE enumeration defines types of **WSRM configuration** objects on which operations can be performed. WSRM configuration objects are used in the IWRMConfig Interface (section 3.2.4.5).

```
typedef [v1_enum] enum
{
  CONFIGTYPE_ACCOUNTING = 1,
  CONFIGTYPE_NOTIFICATION = 2,
  CONFIGTYPE_CALENDARING = 3
```

```
} CONFIGTYPE;
```

CONFIGTYPE_ACCOUNTING: This operation targets the accounting configuration.

CONFIGTYPE_NOTIFICATION: This operation targets the notification configuration.

CONFIGTYPE_CALENDARING: This operation targets the calendar configuration.

2.2.3.2 EXCLUSIONLIST_TYPE Enumeration

The EXCLUSIONLIST_TYPE enumeration defines types of server **exclusion list**. A WSRM exclusion list is in the form of an ExclusionList element (section 2.2.5.16) in XML. They are used by IWRMConfig Interface (section 3.2.4.5).

```
typedef [v1_enum] enum
{
  SYSTEM_EXCLUSION_LIST = 1,
  USER_EXCLUSION_LIST = 2,
  DEFAULT_USER_EXCLUSION_LIST = 4
} EXCLUSIONLIST_TYPE;
```

SYSTEM_EXCLUSION_LIST: The exclusion list is system-defined.

USER_EXCLUSION_LIST: The exclusion list is user-defined.

DEFAULT_USER_EXCLUSION_LIST: The exclusion list is the default user-defined exclusion list.<9>

2.2.3.3 IMPORT_TYPE Enumeration

The IMPORT_TYPE enumeration defines modes for importing objects when conflicting objects are present in the configuration.

```
typedef [v1_enum] enum
{
  OVERWRITE_IMPORT = 1,
  IGNORE_EXISTING_IMPORT = 2,
  OVERRIDE_EXISTING_IMPORT = 3,
  SMART_MERGE_RENAME_EXISTING_IMPORT = 4,
  SMART_MERGE_RENAME_IMPORTED_IMPORT = 5,
} IMPORT_TYPE;
```

OVERWRITE_IMPORT: Specifies that objects of a given type SHOULD be removed prior to importing more objects of that type.

IGNORE_EXISTING_IMPORT: Specifies that the import of objects that conflict with existing objects in the configuration SHOULD be ignored.

OVERRIDE_EXISTING_IMPORT: Specifies that the import of objects that conflict with existing objects in the configuration SHOULD cause the existing objects to be deleted.

SMART_MERGE_RENAME_EXISTING_IMPORT: Specifies that existing objects in the configuration that conflict with imported objects SHOULD be renamed prior to importing.

SMART_MERGE_RENAME_IMPORTED_IMPORT: Specifies that imported objects that conflict with existing objects in the configuration SHOULD be renamed prior to importing.

2.2.3.4 MACHINE_GROUP_MERGE_OPTIONS Enumeration

The MACHINE_GROUP_MERGE_OPTIONS enumeration defines options for machine group modification.

```
typedef [v1_enum] enum
{
  OVERWRITE_MG_MERGE_OPTION = 1,
  OVERRIDE_MG_MERGE_OPTION = 2,
  APPEND_MG_MERGE_OPTION = 3,
  SMART_MG_MERGE_OPTION = 4,
} MACHINE_GROUP_MERGE_OPTIONS;
```

OVERWRITE_MG_MERGE_OPTION: Specifies that the machine group configuration SHOULD be overwritten.

OVERRIDE_MG_MERGE_OPTION: Specifies that the machine group configuration SHOULD be overridden. That means that if the same machine group name exists with the same hierarchy then this (along with its children) SHOULD be replaced by the new machine group node. Nonconflicting new nodes SHOULD be imported as is. Nonconflicting existing nodes SHOULD remain as is.

APPEND_MG_MERGE_OPTION: Specifies that the machine group configuration SHOULD be appended. This means that nonconflicting nodes SHOULD remain as is in the existing configuration. Nonconflicting new nodes SHOULD be imported as is. Conflicting nodes SHOULD not be modified.

SMART_MG_MERGE_OPTION: Specifies that the machine group configuration SHOULD be handled smartly. This means that nonconflicting nodes SHOULD remain as is in the existing configuration. Nonconflicting new nodes SHOULD be imported as is. Conflicting nodes SHOULD be merged in a way that the name of the imported node will be changed to avoid name conflict. The new name of the conflicting object, which is being imported, is built by appending its original name with the string "##@" followed by a number. The numbers used are in the range 1-16384 and a number, once used, is not reused until the object it was used in is either deleted or renamed. For example, an object name "ObjectName" might become "ObjectName##@1" after using this option. This range is sufficiently large and is not consumed completely because, in the WSRM configuration, the maximum allowed number of objects of each type is 128. In addition, whenever an object with a smart name is deleted or renamed, the number that was used in its name is available for reuse.

2.2.3.5 MANAGEMENT_TYPE Enumeration

The MANAGEMENT_TYPE enumeration defines management modes for the WSRM Protocol server. Management modes are used by IWRMPolicy methods (section 3.2.4.7).

```
typedef [v1_enum] enum
{
  MANUAL_ACTIVE_POLICY = 1,
  CALENDAR_POLICY = 2,
  PROFILING = 3
} MANAGEMENT_TYPE;
```

MANUAL_ACTIVE_POLICY: The server is in manual active policy mode. In this mode, WSRM manages the CPU and memory allocation for different processes according to the PMCs that are defined by the current resource policy. The current resource policy is selected using the SetCurrentPolicy method (section 3.2.4.7.12).

CALENDAR_POLICY: The server is in calendar mode. In this mode, a **resource policy** is managing the CPU and memory allocation for different processes according to the PMCs defined by the policy. The policy was set and the management mode was set to calendar mode as a result of a calendar event, which is created and managed by IWRMCalendar methods (section 3.2.4.4).

PROFILING: The server is in profiling mode. In this mode, WSRM does not manage the CPU and memory allocation for processes. If accounting is enabled, process properties of all running processes are logged into the accounting database.

2.2.3.6 OBJECT_TYPE Enumeration

The OBJECT_TYPE enumeration defines types of objects on which an operation can be performed. The WSRM objects are used by the GetDependencies (section 3.2.4.1.6) and ExportObjects (section 3.2.4.2.1) methods.

```
typedef [v1_enum] enum
{
  OBJECT_SELECTION_CRITERIA = 1,
  OBJECT_POLICY = 2,
  OBJECT_SCHEDULE = 3
} OBJECT_TYPE;
```

OBJECT_SELECTION_CRITERIA: The target object is a process matching criteria (PMC).

OBJECT_POLICY: The target object is a policy.

OBJECT_SCHEDULE: The target object is a calendar or a schedule object.

2.2.3.7 RESTORE_MODE Enumeration

The RESTORE_MODE enumeration defines states that the WSRM configuration can have when it is restored. The WSRM configuration is restored by the RestoreXMLFiles method (section 3.2.4.1.5).

```
typedef [v1_enum] enum
{
  RESTORE_LAST_GOOD_STATE = 1,
  RESTORE_EMPTY_FILES = 2
} RESTORE_MODE;
```

RESTORE_LAST_GOOD_STATE: This value targets the last saved good state of the WSRM configuration, which is defined as a consistent set of **WSRM** objects. **WSRM** objects are stored in the XML files after successful completion of any method call of this protocol.

RESTORE_EMPTY_FILES: This value targets the initial state of the WSRM configuration with which the product is shipped. This means that only the out-of-box policies, empty calendars, and conditional policies will be used in the restored state of the object.

2.2.4 Namespaces

This specification defines and references various XML namespaces using the mechanisms specified in [XMLNS]. Although this specification associates a particular prefix for each namespace that is used, the choice of a particular prefix is implementation-specific and is not significant for interoperability.

Prefix	Namespace URI	Reference
xs	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1]
xsd	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1]

2.2.5 XML Data Formats

The WSRM Protocol specifies XML data formats for strings that are used in messages and method parameters. These formats are defined using **XSD** schema syntax specified in [W3C-XSD].

The following table lists the XML data formats that are defined in this section.

Name	Description
AccountingClientList (section 2.2.5.1)	Specifies a list of **accounting clients**.
AccountingConfigInfo (section 2.2.5.2)	Specifies configuration information concerning the accounting database.
AccountingMetaData (section 2.2.5.1)	Specifies accounting metadata, including database properties.
AccountingProcessList (section 2.2.5.4)	Specifies a list of **accounting processes** and their properties.
AccountingQueryCondition (section 2.2.5.5)	Specifies a query on the accounting database.
AppPoolList (section 2.2.5.6)	Specifies an IIS application pool list structure.
Calendar (section 2.2.5.7)	Controls the scheduling of **resource allocation**.
CalendarEvent (section 2.2.5.8)	Specifies the scheduling parameters for a calendar event.
CalendarRule (section 2.2.5.9)	Specifies the calendar parameters for a single calendar event
Calendars (section 2.2.5.10)	Specifies one or more calendar objects.
CalendarsCollection (section 2.2.5.11)	Specifies a collection of calendar and schedule objects.
ConditionalPolicy (section 2.2.5.12)	Controls the selection of **resource allocation policies (RAPs)**.
ConfigurationFiles (section 2.2.5.13)	Specifies configurations used for the import and export of user information.
DependencyList (section 2.2.5.14)	Specifies dependent policies and calendar events.
Events (section 2.2.5.15)	Specifies a list of scheduled calendar events.
ExclusionList (section 2.2.5.16)	Specifies processes to exclude from management.
Machine (section 2.2.5.17)	Specifies parameters for managing machines within machine groups .
MachineGroup (section 2.2.5.18)	Specifies parameters for managing machine groups.
NotificationConfigInfo (section 2.2.5.19)	Specifies the notification configuration.
ObjectIds (section 2.2.5.20)	Specifies objects on the server.
Policy (section 2.2.5.21)	Specifies parameters for managing resource allocation.
PolicyCollection (section 2.2.5.22)	Specifies a collection of policy objects.
ProcessList (section 2.2.5.23)	Specifies a list of processes and their properties for a **resource allocation policy (RAP)**.
ProcessMatchingCriteria (section 2.2.5.24)	Specifies parameters for resource management.

Name	Description
ProcessMatchingCriteriaCollection (section 2.2.5.25)	Specifies a collection of process matching criteria objects.
Schedule (section 2.2.5.26)	Specifies a schedule for a calendar event.
Schedules (section 2.2.5.27)	Specifies one or more schedule objects
ServiceList (section 2.2.5.28)	Specifies services that are registered with the server.
SupportedClients (section 2.2.5.29)	Specifies the level of support for clients on the WSRM server.
Users (section 2.2.5.30)	Specifies categories of remote session users.

The XML data contained within messages of the WSRM Protocol MUST obey the syntax of well-formed XML 1.0 documents ([XML1.0] section 2.1). **Schema-validity assessment** of each document's root element MUST result in a value of "valid" for the **validity** property ([XMLSCHEMA1/2] section 3.3.4).

Samples of XML data formats are provided in Sample XML Data (section 4.2).

2.2.5.1 AccountingClientList Element

The AccountingClientList XML element is used to specify a list of accounting clients.

```
<xs:element name="AccountingClientList">
  <xs:complexType>
    <xs:choice
      minOccurs="0"
      maxOccurs="unbounded"
    >
      <xs:element name="AccountingClient"
        nillable="true"
      >
        <xs:complexType name="AccountingClient">
          <xs:simpleContent>
            <xs:extension
              base="xs:string"
            >
              <xs:attribute name="Enabled"
                type="xs:string"
               />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
AccountingClient	AccountingClient	A specification of an accounting client machine.

Attributes

Name	Type	Description
Enabled	xs:string	A string that specifies whether the accounting functionality of the accounting client is

Name	Type	Description
		currently enabled, in Boolean format (section 2.2.1.2).

2.2.5.2 AccountingConfigInfo Element

The AccountingConfigInfo XML element is used to specify configuration information concerning the accounting database. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="AccountingConfigInfo">
  <xs:complexType name="AccountingConfigInfoType">
    <xs:sequence>
      <xs:element name="AccountingEnabled"
        type="xsd:string"
        minOccurs="1"
        maxOccurs="1"
       />
      <xs:element name="RecordWriteInterval"
        type="xsd:double"
        minOccurs="1"
        maxOccurs="1"
       />
      <xs:element name="DatabaseLocation"
        type="xsd:string"
        minOccurs="1"
        maxOccurs="1"
       />
      <xs:element name="DatabaseServer"
        type="xsd:string"
        minOccurs="1"
        maxOccurs="1"
       />
      <xs:element name="DatabaseInstance"
        type="xsd:string"
        minOccurs="1"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
AccountingEnabled	xsd:string	Whether the accounting database is currently enabled for data collection, in Boolean format (section 2.2.1.2).
RecordWriteInterval	xsd:double	The time interval for data collection in minutes.
DatabaseLocation	xsd:string	The file system path where the accounting database is located.
DatabaseServer	xsd:string	The name of the server on which the accounting database is located. Its value "." denotes the name of the WSRM server the client is connected to.
DatabaseInstance	xsd:string	The name of the database server instance for the accounting database.

Additional XML data formats are specified in section 2.2.5.

2.2.5.3 AccountingMetaData Element

The AccountingMetaData XML element is used to specify metadata for the accounting database.

```
<xs:element name="AccountingMetaData">
  <xs:complexType>
    <xs:choice
      minOccurs="0"
      maxOccurs="unbounded"
    >
      <xs:element name="Column">
        <xs:complexType name="Column">
          <xs:sequence>
            <xs:element name="BitmaskPosition"
              type="xs:integer"
              minOccurs="0"
             />
            <xs:element name="ColumnName"
              type="xs:string"
              minOccurs="0"
             />
            <xs:element name="IsVisible"
              type="xs:integer"
              minOccurs="0"
             />
            <xs:element name="IsArchivable"
              type="xs:integer"
              minOccurs="0"
             />
            <xs:element name="IsPivotable"
              type="xs:integer"
              minOccurs="0"
             />
            <xs:element name="ColumnType"
              type="xs:integer"
              minOccurs="0"
             />
            <xs:element name="Units"
              type="xs:string"
              minOccurs="0"
              nillable="true"
             />
            <xs:element name="AggregationCollection"
              minOccurs="0"
              maxOccurs="unbounded"
            >
              <xs:complexType name="AggregationCollection">
                <xs:sequence>
                  <xs:element name="Aggregation"
                    minOccurs="0"
                    maxOccurs="unbounded"
                  >
                    <xs:complexType name="Aggregation">
                      <xs:sequence>
                        <xs:element name="Pivot"
                          type="xs:string"
                          minOccurs="0"
                         />
                        <xs:element name="AggregateFunction"
                          type="xs:string"
                          minOccurs="0"
                         />
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
```

```
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Column	Column	A specification of the properties of a column of the database.
BitmaskPosition	xs:integer	An integer designating the position assigned to the column. Its value will vary from 1 to the maximum number of columns. Two columns cannot have the same BitmaskPosition value.
ColumnName	xs:string	A string that specifies the name of a column in the database.
IsVisible	xs:integer	An integer that specifies whether the column can be viewed, expressed as a Boolean value (section 2.2.2.1).
IsArchivable	xs:integer	An integer that specifies whether the column can be archived, expressed as a Boolean value.
IsPivotable	xs:integer	An integer that specifies whether the column can be rotated to rows, expressed as a Boolean value.
ColumnType	xs:integer	An integer that specifies the type of column. Its value will be one of the constants defined in Column Types (section 2.2.2.11).
Units	xs:string	A string that specifies units, or the NULL value. The string can be either "ms", for milliseconds, or "KB", for kilobytes.
AggregationCollection	AggregationCollection	A specification of a group of aggregations.
Aggregation	Aggregation	A specification of an aggregation, which is a set of objects in the database.
Pivot	xs:string	A string specifying a column name, which can be used for grouping of rows. If used, the associated AggregateFunction will be applied to the values of the current column.
AggregateFunction	xs:string	A string that specifies an aggregate function, which performs a calculation on rows in the column.

2.2.5.4 AccountingProcessList Element

The AccountingProcessList XML element is used to specify a list of accounting processes and their properties. These properties correspond to data items in the accounting database. See section 3.2.1.2 for descriptions of these data items.

```
<xs:element name="AccountingProcessList">
  <xs:complexType>
    <xs:choice
      minOccurs="0"
      maxOccurs="unbounded"
    >
      <xs:element name="Process">
        <xs:complexType name="Process">
          <xs:sequence>
```

```
<xs:element name="EventType"
  type="xs:string"
  minOccurs="0"
 />
<xs:element name="UserModeTime"
  type="xs:unsignedLong"
  minOccurs="0"
 />
<xs:element name="KernelModeTime"
  type="xs: unsignedLong"
  minOccurs="0"
 />
<xs:element name="ReadOperationCount"
  type="xs: unsignedLong"
  minOccurs="0"
 />
<xs:element name="WriteOperationCount"
  type="xs: unsignedLong"
  minOccurs="0"
 />
<xs:element name="OtherOperationCount"
  type="xs: unsignedLong"
  minOccurs="0"
 />
<xs:element name="ReadTransferCount"
  type="xs: unsignedLong"
  minOccurs="0"
 />
<xs:element name="WriteTransferCount"
  type="xs: unsignedLong"
  minOccurs="0"
 />
<xs:element name="OtherTransferCount"
  type="xs: unsignedLong"
  minOccurs="0"
 />
<xs:element name="WorkingSetSize"
  type="xs: unsignedLong"
  minOccurs="0"
 />
<xs:element name="VirtualSize"
  type="xs:double"
  minOccurs="0"
 />
<xs:element name="PrivatePageCount"
  type="xs: unsignedLong"
  minOccurs="0"
 />
<xs:element name="ImageName"
  type="xs:string"
  minOccurs="0"
 />
<xs:element name="ResourceGroupName"
  type="xs:string"
  minOccurs="0"
 />
<xs:element name="UserName"
  type="xs:string"
  minOccurs="0"
 />
<xs:element name="DomainName"
  type="xs:string"
  minOccurs="0"
 />
<xs:element name="ImagePath"
  type="xs:string"
  minOccurs="0"
 />
<xs:element name="ProcessCommandLine"
  type="xs:string"
```

```
              minOccurs="0"
             />
            <xs:element name="PolicyName"
              type="xs:string"
              minOccurs="0"
             />
            <xs:element name="CreationTime"
              type="xs:unsignedLong"
              minOccurs="0"
             />
            <xs:element name="CreationSystemTime"
              type="xs:unsignedLong"
              minOccurs="0"
             />
            <xs:element name="PolicySetTime"
              type="xs:unsignedLong"
              minOccurs="0"
             />
            <xs:element name="ProcessId"
              type="xs:unsignedInt"
              minOccurs="0"
             />
            <xs:element name="ParentProcessId"
              type="xs: unsignedInt"
              minOccurs="0"
             />
            <xs:element name="SessionId"
              type="xs: unsignedInt"
              minOccurs="0"
             />
            <xs:element name="ThreadCount"
              type="xs: unsignedInt"
              minOccurs="0"
             />
            <xs:element name="PageFaultCount"
              type="xs: unsignedInt"
              minOccurs="0"
             />
            <xs:element name="PageFileUsage"
              type="xs: unsignedInt"
              minOccurs="0"
             />
            <xs:element name="PeakPageFileUsage"
              type="xs: unsignedInt"
              minOccurs="0"
             />
            <xs:element name="QuotaNonPagedPoolUsage"
              type="xs: unsignedInt"
              minOccurs="0"
             />
            <xs:element name="QuotaPagedPoolUsage"
              type="xs: unsignedInt"
              minOccurs="0"
             />
            <xs:element name="QuotaPeakNonPagedPoolUsage"
              type="xs: unsignedInt"
              minOccurs="0"
             />
            <xs:element name="QuotaPeakPagedPoolUsage"
              type="xs: unsignedInt"
              minOccurs="0"
             />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Process	Process	A specification of the properties of an accounting process.
EventType	xs:string	A string that specifies the event upon which properties of a process are recorded. The string can be "C" for process creation, "D" for process termination, and "L" for regular logging.
UserModeTime	xs:unsignedLong	A 64-bit unsigned integer value that specifies the length of time, in 100-nanosecond units, the process has executed in user mode.
KernelModeTime	xs: unsignedLong	A 64-bit unsigned integer value that specifies the length of time, in 100-nanosecond units, the process has executed in kernel mode.
ReadOperationCount	xs: unsignedLong	A 64-bit unsigned integer value that specifies the number of read operations that have been performed by the process.
WriteOperationCount	xs: unsignedLong	A 64-bit unsigned integer value that specifies the number of write operations that have been performed by the process.
OtherOperationCount	xs: unsignedLong	A 64-bit unsigned integer value that specifies the number of non-read and non-write operations that have been performed by the process.
ReadTransferCount	xs: unsignedLong	A 64-bit unsigned integer value that specifies the amount of data, in bytes, that has been read by the process.
WriteTransferCount	xs: unsignedLong	A 64-bit unsigned integer value that specifies the amount of data, in bytes, that has been written by the process.
OtherTransferCount	xs: unsignedLong	A 64-bit unsigned integer value that specifies the amount of data, in bytes, that has been transferred by the process.
WorkingSetSize	xs: unsignedLong	A 64-bit unsigned integer value that specifies the working set size, in bytes, for the process.
VirtualSize	xs:double	A 64-bit unsigned integer value that specifies the size of virtual memory, in bytes, for the process.
PrivatePageCount	xs: unsignedLong	A 64-bit unsigned integer value that specifies the number of private memory pages used by the process.
ImageName	xs:string	A string that specifies the image name associated with the process.
ResourceGroupName	xs:string	A string that specifies the resource group name associated with the process.
UserName	xs:string	A string that specifies the user name associated with the process.
DomainName	xs:string	A string that specifies the **domain name** associated with the process.
ImagePath	xs:string	A string that specifies the image path.
ProcessCommandLine	xs:string	A string that specifies the contents of the command line for invoking the process.
PolicyName	xs:string	A string that specifies the name of the resource allocation

Element	Type	Description
		policy (RAP) associated with the process.
CreationTime	xs:unsignedLong	A 64-bit unsigned integer that specifies the creation time of the process. Its lower 32 bits represent the **dwLowDateTime** field, and its higher 32 bits represent the **dwHighDateTime** field of the **FILETIME** structure as specified in [MS-DTYP] section 2.3.3.
CreationSystemTime	xs:unsignedLong	A 64-bit unsigned integer that specifies the time the data was entered into the database. Its lower 32 bits represent the **dwLowDateTime** field, and its higher 32 bits represent the **dwHighDateTime** field of the **FILETIME** structure as specified in [MS-DTYP] section 2.3.3.
PolicySetTime	xs:unsignedLong	A 64-bit unsigned integer that specifies the time that the RAP was set as the current resource policy. Its lower 32 bits represent the **dwLowDateTime** field, and its higher 32 bits represent the **dwHighDateTime** field of the **FILETIME** structure as specified in [MS-DTYP] section 2.3.3.
ProcessId	xs:unsignedInt	A 32-bit integer value that uniquely distinguishes a process while it runs.
ParentProcessId	xs: unsignedInt	A 32-bit integer value that specifies the <ProcessId> of the parent process.
SessionId	xs: unsignedInt	A 32-bit integer value that specifies the identifier of the session to which this process belongs.
ThreadCount	xs: unsignedInt	A 32-bit integer value that specifies the number of active threads for the process.
PageFaultCount	xs: unsignedInt	A 32-bit integer value that specifies the number of page faults generated by the process.
PageFileUsage	xs: unsignedInt	A 32-bit integer value that specifies the amount of the page file, in kilobytes, used by the process.
PeakPageFileUsage	xs: unsignedInt	A 32-bit integer value that specifies the maximum usage of the page file, in kilobytes, by the process.
QuotaNonPagedPoolUsage	xs: unsignedInt	A 32-bit integer value that specifies the quota amount of non-paged pool usage, in kilobytes, for the process.
QuotaPagedPoolUsage	xs: unsignedInt	A 32-bit integer value that specifies the quota amount of paged pool usage, in kilobytes, for the process.
QuotaPeakNonPagedPoolUsage	xs: unsignedInt	A 32-bit integer value that specifies the maximum quota amount of non-paged pool usage, in kilobytes, for the process.
QuotaPeakPagedPoolUsage	xs: unsignedInt	A 32-bit integer value that specifies the maximum quota amount of non-paged pool usage, in kilobytes, for the process.

Additional XML data formats are specified in section 2.2.5.

2.2.5.5 AccountingQueryCondition Element

The AccountingQueryCondition XML element is used to specify a query on the accounting database. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="AccountingQueryCondition">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SelectFieldCollection">
        <xs:complexType name="SelectFieldCollection">
          <xs:sequence>
            <xs:element name="Column"
              type="xs:string"
              maxOccurs="unbounded"
             />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="GroupColumnCollection">
        <xs:complexType name="GroupColumnCollection">
          <xs:sequence>
            <xs:element name="Column"
              type="xs:string"
              maxOccurs="unbounded"
             />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="OrderColumnCollection">
        <xs:complexType name="OrderColumnCollection">
          <xs:sequence>
            <xs:element name="OrderInfo"
              maxOccurs="unbounded"
            >
              <xs:complexType name="OrderInfo">
                <xs:sequence>
                  <xs:element name="Column"
                    type="xs:string"
                    maxOccurs="unbounded"
                   />
                  <xs:element name="IsAscending"
                    type="xs:unsignedByte"
                   />
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="WhereClause"
        type="xs:string"
       />
      <xs:element name="HavingClause"
        type="xs:string"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
SelectFieldCollection	SelectFieldCollection	A specification of a collection of field selections for an accounting database query.
Column	xs:string	The column names of the fields to select.
GroupColumnCollection	GroupColumnCollection	A specification of a collection of column groupings for a data query.

Element	Type	Description
Column	xs:string	The column names of the fields to group.
OrderColumnCollection	OrderColumnCollection	A specification of a collection of column orderings for a data query.
OrderInfo	OrderInfo	A specification of a column ordering.
Column	xs:string	The column names of the fields to order.
IsAscending	xs:unsignedByte	The sort order, expressed as a **Boolean** value (section 2.2.2.1).
WhereClause	xs:string	Conditions to filter the data in the result set of the query.
HavingClause	xs:string	Conditions to filter the data in the result set of the query. This element SHOULD be used only with the <GroupColumnCollection> element.

The <WhereClause> and <HavingClause> elements are used as **WHERE** and **HAVING** clauses, respectively, of SQL stored procedures. They SHOULD follow SQL format, except that ("\n") SHOULD be used instead of apostrophe ("'").

For example:

```
[ComputerName] = N\nNISWIN7AMD3\n AND [ProcessName] LIKE N\n%A%\n
```

"N" is needed for **Unicode** [UNICODE] support.

Additional XML data formats are specified in section 2.2.5.

2.2.5.6 AppPoolList Element

The AppPoolList XML element specifies an IIS application pool list structure. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="AppPoolList">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AppPool"
        type="xs:string"
        minOccurs="0"
        maxOccurs="unbounded"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
AppPool	xs:string	A string that specifies the application pool.

The following schema shows a reference to the AppPoolList element from a public XML namespace [XMLNS]:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="AppPoolList" />
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Additional XML data formats are specified in section 2.2.5.

2.2.5.7 Calendar Element

The Calendar XML element specifies a calendar object, which controls the scheduling of resource allocation. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="Calendar"
  minOccurs="1"
  maxOccurs="unbounded"
>
  <xs:complexType name="Calendar">
    <xs:sequence>
      <xs:element name="CalendarName"
        type="xsd:string"
        minOccurs="1"
        maxOccurs="1"
       />
      <xs:element name="CalendarDate"
        type="xsd:string"
        minOccurs="0"
        maxOccurs="1"
       />
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        ref="CalendarRule"
       />
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        ref="CalendarEvent"
       />
      <xs:element name="ScheduleName"
        type="xsd:string"
        minOccurs="0"
        maxOccurs="1"
       />
      <xs:element name="ScheduleDtStart"
        type="xsd:string"
        minOccurs="0"
        maxOccurs="1"
       />
      <xs:element name="ScheduleDtEnd"
        type="xsd:string"
        minOccurs="0"
        maxOccurs="1"
       />
      <xs:element name="Description"
        type="xsd:string"
        minOccurs="0"
        maxOccurs="1"
```

```
        />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Child Elements

Element	Type	Description
CalendarName	xsd:string	A string that specifies the name associated with the calendar object.
CalendarDate	xsd:string	A string that specifies the calendar date, in date format (section 2.2.1.3).
CalendarRule	CalendarRule	A specification of the calendar rule, in the form of a CalendarRule element (section 2.2.5.9).
CalendarEvent	CalendarEvent	A specification of a calendar event, in the form of a CalendarEvent element (section 2.2.5.8).
ScheduleName	xsd:string	A string that specifies the name of the event schedule associated with the calendar object.
ScheduleDtStart	xsd:string	A string that specifies the start date of the event schedule, in date format.
ScheduleDtEnd	xsd:string	A string that specifies the end date of the event schedule, in date format.
Description	xsd:string	A text description of the calendar object.

Additional XML data formats are specified in section 2.2.5.

2.2.5.8 CalendarEvent Element

The CalendarEvent XML element specifies the scheduling parameters for a calendar event that is associated with a resource allocation policy (RAP). It is used in the definition of Calendar and Schedule elements (section 2.2.5.7 and 2.2.5.26).

```
<xs:element name="CalendarEvent"
  minOccurs="0"
  maxOccurs="1"
>
  <xs:complexType name="CalendarEvent">
    <xs:choice
      minOccurs="1"
      maxOccurs="unbounded"
    >
      <xs:element name="PolicyName"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1"
       />
      <xs:element name="TmStart"
        type="xs:string"
        minOccurs="1"
        maxOccurs="1"
       />
      <xs:element name="DurationDays"
        type="xs:unsignedByte"
        minOccurs="0"
        maxOccurs="1"
       />
      <xs:element name="DurationHours"
        type="xs:unsignedByte"
        minOccurs="0"
```

```
        maxOccurs="1"
       />
      <xs:element name="DurationMinutes"
        type="xs:unsignedByte"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
PolicyName	xs:string	A string that specifies the name of the policy associated with the event.
TmStart	xs:string	A string that specifies the start time of the event, in time format (section 2.2.1.3).
DurationDays	xs:unsignedByte	An 8-bit unsigned integer that specifies the duration in days of the event.
DurationHours	xs:unsignedByte	An 8-bit unsigned integer that specifies the duration in hours of the event.
DurationMinutes	xs:unsignedByte	An 8-bit unsigned integer that specifies the duration in minutes of the event.

Additional XML data formats are specified in section 2.2.5.

2.2.5.9 CalendarRule Element

The CalendarRule XML element specifies the calendar parameters for a single calendar event. It is the root element of XML documents that are used in WSRM method parameters, including methods of the IWRMCalendar interface (section 3.2.4.4).

```
<xs:element name="CalendarRule"
  minOccurs="0"
  maxOccurs="1"
>
  <xs:complexType name="CalendarRule">
    <xs:choice
      minOccurs="1"
      maxOccurs="unbounded"
    >
      <xs:element name="DtStart"
        type="xsd:string"
        minOccurs="1"
        maxOccurs="1"
       />
      <xs:element name="DtEnd"
        type="xsd:string"
        minOccurs="0"
        maxOccurs="1"
       />
      <xs:element name="Freq"
        type="xsd:string"
        minOccurs="1"
        maxOccurs="1"
       />
      <xs:element name="Interval"
        type="xsd:double"
        minOccurs="1"
        maxOccurs="1"
       />
```

```
<xs:element name="ByDay"
  minOccurs="0"
  maxOccurs="1"
>
  <xs:complexType name="ByDay">
    <xs:sequence
      maxOccurs="unbounded"
      minOccurs="0"
    >
      <xs:element name="Day"
        type="xsd:string"
        minOccurs="0"
        maxOccurs="unbounded"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ByMonthDay"
  minOccurs="0"
  maxOccurs="1"
>
  <xs:complexType name="ByMonthDay">
    <xs:sequence>
      <xs:element name="MonthDay"
        type="xsd:double"
        minOccurs="0"
        maxOccurs="unbounded"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ByYearDay"
  minOccurs="0"
  maxOccurs="1"
>
  <xs:complexType name="ByYearDay">
    <xs:sequence>
      <xs:element name="YearDay"
        type="xsd:double"
        minOccurs="0"
        maxOccurs="unbounded"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ByWeekNo"
  minOccurs="0"
  maxOccurs="1"
>
  <xs:complexType name="ByWeekNo">
    <xs:sequence>
      <xs:element name="WeekNo"
        type="xsd:double"
        minOccurs="0"
        maxOccurs="unbounded"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ByMonth"
  minOccurs="0"
  maxOccurs="1"
>
  <xs:complexType name="ByMonth">
    <xs:sequence>
      <xs:element name="Month"
        type="xsd:double"
        minOccurs="0"
        maxOccurs="unbounded"
       />
```

```
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="BySetPos"
          minOccurs="0"
          maxOccurs="1"
        >
          <xs:complexType name="BySetPos">
            <xs:sequence>
              <xs:element name="SetPos"
                type="xsd:double"
                minOccurs="0"
                maxOccurs="unbounded"
               />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
```

Child Elements

Element	Type	Description
DtStart	xsd:string	A string that specifies the calendar start date, in date format (section 2.2.1.3).
DtEnd	xsd:string	A string that specifies the calendar end date, in date format.
Freq	xsd:string	A string that specifies the calendar frequency, expressed as a frequency option (section 2.2.2.5).
Interval	xsd:double	A floating-point value that specifies the interval of the frequency. For example, if the <Freq> value is "weekly" and the <Interval> is "2.0", the frequency is biweekly.
ByDay	ByDay	A specification of calendar occurrence by day of the week.
Day	xsd:string	A string that specifies the day of the week, expressed as a day option (section 2.2.2.4). <Day> is used in weekly recurring calendars to specify that on a particular day of the week, a calendar event SHOULD be fired.
ByMonthDay	ByMonthDay	A specification of calendar occurrence by day of the month.
MonthDay	xsd:double	A floating-point value that specifies the day of the month, expressed as a value from 1 to 31. <MonthDay> is used in monthly recurring calendars to specify that on a particular day of the month, a calendar event SHOULD be fired.
ByYearDay	ByYearDay	A specification of calendar occurrence by day of the year.
YearDay	xsd:double	A floating-point value that specifies the day of the year, expressed as a value from 1 to 366. <YearDay> is used in annually recurring calendars to specify that on a particular day of the year, a calendar event SHOULD be fired.<10>
ByWeekNo	ByWeekNo	A specification of calendar occurrence by week of the year.
WeekNo	xsd:double	A floating-point value that specifies the week of the year, expressed as a value from 1 to 53. <WeekNo> is used in annually recurring calendars to specify that in a particular week of the year, a calendar event SHOULD be fired.<11>

Element	Type	Description
ByMonth	ByMonth	A specification of calendar occurrence by month of the year.
Month	xsd:double	A floating-point value that specifies the month of the year, expressed as a value from 1 to 12. <Month> is used in annually recurring calendars to specify that in a particular month of the year, a calendar event SHOULD be fired.
BySetPos	BySetPos	A specification of calendar occurrence by monthly instance of a day of the week.
SetPos	xsd:double	A floating-point value that specifies the monthly instance of a day of the week, expressed as a day modifier (section 2.2.2.3). <BySetPos> is used with <Day> and <Month> elements in annually recurring calendars to specify that on a particular monthly instance of a day of the week, a calendar event SHOULD be fired. For example, the following elements specify the second Sunday in January: <ByDay><Day>Su</Day></ByDay> <ByMonth><Month>1</Month></ByMonth> <BySetPos><SetPos>2</SetPos></BySetPos> For another example, the following elements specify the last Monday in March: <ByDay><Day>Mo</Day></ByDay> <ByMonth><Month>3</Month></ByMonth> <BySetPos><SetPos>-1</SetPos></BySetPos>

A calendar event can be triggered by a schedule, such as whenever a specified scheduled event occurs. In such cases, the policy defined by the schedule is assigned the **PROFILING** value from the MANAGEMENT_TYPE enumeration (section 2.2.3.5).

Additional XML data formats are specified in section 2.2.5.

2.2.5.10 Calendars Element

The Calendars XML element specifies one or more calendar objects. It is the root element of XML documents that are used in WSRM method parameters, including methods of the IWRMCalendar interface (section 3.2.4.4).

```
<xs:element name="Calendars">
  <xs:complexType>
    <xs:sequence>
      <xs:element
        maxOccurs="unbounded"
        ref="Calendar"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Calendar	Calendar	A specification of a Calendar element (section 2.2.5.7) that defines an individual calendar object.

Additional XML data formats are specified in section 2.2.5.

2.2.5.11 CalendarsCollection Element

The CalendarsCollection XML element specifies a collection of calendar and schedule objects. It is the root element of XML documents that are used in WSRM method parameters, including methods of the IResourceManager interface (section 3.2.4.1).<12>

```
<xs:element name="CalendarsCollection">
  <xs:complexType>
    <xs:choice
      minOccurs="0"
      maxOccurs="unbounded"
    >
      <xs:element
        ref="CalendarEvent"
       />
      <xs:element
        ref="Calendars"
       />
      <xs:element
        ref="Schedules"
       />
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
CalendarEvent	CalendarEvent	A specification of a calendar event, in the form of a CalendarEvent element (section 2.2.5.8).
Calendars	Calendars	A specification of a Calendars element (section 2.2.5.10) that defines one or more calendar objects.
Schedules	Schedules	A specification of a Schedules element (section 2.2.5.27) that defines one or more schedule objects.

Additional XML data formats are specified in section 2.2.5.

2.2.5.12 ConditionalPolicy Element

The ConditionalPolicy XML element is used to control the selection of RAP. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="ConditionalPolicy">
  <xs:complexType name="ConditionalPolicy">
    <xs:sequence>
      <xs:element name="ConditionsList"
        minOccurs="1"
        maxOccurs="unbounded"
      >
        <xs:complexType name="ConditionsList">
```

```
<xs:sequence>
  <xs:element name="Condition"
    minOccurs="0"
    maxOccurs="unbounded"
  >
    <xs:complexType name="Condition">
      <xs:choice
        maxOccurs="unbounded"
      >
        <xs:element name="ConditionString">
          <xs:complexType name="ConditionString">
            <xs:attribute name="ConditionState"
              type="xs:string"
              use="required"
             />
            <xs:attribute name="Operator"
              type="xs:string"
              use="optional"
             />
            <xs:attribute name="Num"
              type="xs:integer"
              use="optional"
             />
            <xs:attribute name="Value"
              type="xs:string"
              use="optional"
             />
          </xs:complexType>
        </xs:element>
        <xs:element name="ConditionRelationship">
          <xs:complexType name="ConditionRelationship">
            <xs:attribute name="Operator"
              type="xs:string"
              use="required"
             />
            <xs:attribute name="NextNode"
              type="xs:string"
              use="required"
             />
          </xs:complexType>
        </xs:element>
        <xs:element name="ConditionEvaluation">
          <xs:complexType name="ConditionEvaluation">
            <xs:attribute name="Result"
              type="xs:boolean"
              use="required"
             />
          </xs:complexType>
        </xs:element>
        <xs:element name="Action">
          <xs:complexType name="Action">
            <xs:sequence>
              <xs:element name="SwitchToPolicy"
                type="xs:string"
               />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:attribute name="Category"
        type="xs:string"
        use="required"
       />
      <xs:attribute name="Name"
        type="xs:string"
        use="required"
       />
      <xs:attribute name="Active"
        type="xs:boolean"
        use="required"
```

```
                  />
                 <xs:attribute name="ID"
                   type="xs:unsignedByte"
                   use="required"
                  />
               </xs:complexType>
             </xs:element>
           </xs:sequence>
         </xs:complexType>
       </xs:element>
     </xs:sequence>
     <xs:attribute name="Name"
       type="xsd:string"
       use="required"
      />
   </xs:complexType>
 </xs:element>
```

Child Elements

Element	Type	Description
ConditionsList	ConditionsList	A specification of a set of conditions for filtering RAP.
Condition	Condition	A specification of a condition for filtering RAP.
ConditionString	ConditionString	A string that contains an argument that defines the condition.
ConditionRelationship	ConditionRelationship	A specification of a conditional relationship.
ConditionEvaluation	ConditionEvaluation	A specification of conditional evaluation.
Action	Action	A specification of an action based on the evaluation of the condition.
SwitchToPolicy	xs:string	The name of the policy to switch to, or "no action".

Attributes

Name	Type	Description
ConditionState	xs:string	A description of the **condition state**.
Operator	xs:string	An operator associated with the condition.
Num	xs:integer	A numerical argument to associate with the condition.
Value	xs:string	A value associated with the condition.
Operator	xs:string	The logical operator associated with the conditional relationship.
NextNode	xs:string	A link between conditions.
Result	xs:boolean	The result of evaluating the condition, in Boolean format (section 2.2.1.2).
Category	xs:string	The category of the condition defined in Condition Category (section 2.2.2.12).
Name	xs:string	The name of the condition. It can have one of the values from Condition Name (section 2.2.2.13).
Active	xs:boolean	An indication of whether the condition is active, in Boolean format.
ID	xs:unsignedByte	A numeric identifier of the conditional event.

Name	Type	Description
Name	xsd:string	The name of the conditional policy.

Additional XML data formats are specified in section 2.2.5.

2.2.5.13 ConfigurationFiles Element

The ConfigurationFiles XML element is used to specify configurations used for the import and export of user information. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="ConfigurationFiles">
  <xs:complexType name="ConfigurationFiles">
    <xs:choice
      minOccurs="0"
      maxOccurs="unbounded"
    >
      <xs:element name="ConfigurationFile">
        <xs:complexType name="ConfigurationFile">
          <xs:sequence>
            <xs:element
              minOccurs="0"
              maxOccurs="unbounded"
              ref="Users"
             />
          </xs:sequence>
          <xs:attribute name="FileName"
            type="xs:string"
           />
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
ConfigurationFile	ConfigurationFile	The data file for the object being imported or exported.
Users	Users	The categories of remote session users, in the form of a Users element (section 2.2.5.30).

Attributes

Name	Type	Description
FileName	xs:string	The name of the data file that is being imported or exported in the child element. It is always set to "UserCategories.Xml".

Additional XML data formats are specified in section 2.2.5.

2.2.5.14 DependencyList Element

The DependencyList XML element is used to specify dependent policies, calendar events, schedules, and conditions. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="DependencyList">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PolicyName"
          type="xs:string"
          minOccurs="0"
          maxOccurs="unbounded"
         />
        <xs:element name="ScheduleName"
          type="xs:string"
          minOccurs="0"
          maxOccurs="unbounded"
         />
        <xs:element name="CalendarName"
          type="xs:string"
          minOccurs="0"
          maxOccurs="unbounded"
         />
        <xs:element name="Condition"
          minOccurs="0"
          maxOccurs="unbounded"
          nillable="true"
        >
          <xs:complexType name="Condition">
            <xs:simpleContent>
              <xs:extension
                base="xs:string"
              >
                <xs:attribute name="Num"
                  type="xs:string"
                 />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Child Elements

Element	Type	Description
PolicyName	xs:string	The name of the dependent policy.
ScheduleName	xs:string	The name of the dependent schedule.
CalendarName	xs:string	The name of the dependent calendar.
Condition	Condition	The name of the dependent condition.

Attributes

Name	Type	Description
Num	xs:string	If the Category of the dependent condition (section 2.2.2.12) is "Processor", "Memory", or "ANY", it is "0". Otherwise, if the Category of the dependent condition is "MSCS", it is the value of the **Value** attribute of the first <ConditionString> element of the dependent condition.

The following schema shows the database column name and position specified in the type for the <Condition> element.

```
    <?xml version="1.0"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-
com:xml-msdata"
   elementFormDefault="qualified">
  <xs:element name="DependencyList">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PolicyName" type="xs:string" minOccurs="0"
           maxOccurs="unbounded"/>
        <xs:element name="ScheduleName" type="xs:string" minOccurs="0"
           maxOccurs="unbounded"/>
        <xs:element name="CalendarName" type="xs:string" minOccurs="0"
           maxOccurs="unbounded"/>
        <xs:element name="Condition" nillable="true" minOccurs="0"
           maxOccurs="unbounded">
          <xs:complexType>
            <xs:simpleContent msdata:ColumnName="Condition_Text"
               msdata:Ordinal="1">
              <xs:extension base="xs:string">
                <xs:attribute name="Num" type="xs:string" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Additional XML data formats are specified in section 2.2.5.

2.2.5.15 Events Element

The Events XML element is used to specify a list of scheduled calendar events. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="Events">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="EventData"
        maxOccurs="unbounded"
      >
        <xs:complexType name="EventData">
          <xs:sequence>
            <xs:element name="DtTmStart"
              type="xs:string"
             />
            <xs:element name="DtTmEnd"
              type="xs:string"
             />
            <xs:element name="PolicyName"
              type="xs:string"
             />
            <xs:element name="CalendarName"
              type="xs:string"
             />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
EventData	EventData	A specification of a scheduled event.
DtTmStart	xs:string	A string that specifies the start date and time of the event, in date-T-time format (section 2.2.1.3).
DtTmEnd	xs:string	A string that specifies the end date and time of the event, in date-T-time format.
PolicyName	xs:string	The name of the policy to which this event is linked.
CalendarName	xs:string	The name of the calendar to which this event is linked.

Additional XML data formats are specified in section 2.2.5.

2.2.5.16 ExclusionList Element

The ExclusionList XML element is used to specify an exclusion list of processes to exclude from management. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="ExclusionList">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Process"
        type="xs:string"
        maxOccurs="unbounded"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Process	xs:string	The name of a process to be excluded.

Additional XML data formats are specified in section 2.2.5.

2.2.5.17 Machine Element

The Machine XML element is used to specify parameters for managing machines within machine groups. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="Machine">
  <xs:complexType>
    <xs:attribute name="Name"
      type="xs:string"
     />
    <xs:attribute name="Description"
      type="xs:string"
     />
    <xs:attribute name="FQDN"
      type="xs:string"
     />
    <xs:attribute name="OS"
      type="xs:string"
     />
    <xs:attribute name="OSVersion"
      type="xs:string"
```

```
      />
     <xs:attribute name="LastUpdatedTime"
       type="xs:string"
      />
   </xs:complexType>
 </xs:element>
```

Attributes

Name	Type	Description
Name	xs:string	The name of the machine.
Description	xs:string	A description of the machine.
FQDN	xs:string	The **fully qualified domain name (FQDN)** of the WSRM server.
OS	xs:string	The operating system on which the WSRM server is running.
OSVersion	xs:string	The version of the operating system on which the WSRM server is running.
LastUpdatedTime	xs:string	The last time this machine information was updated, in date-colon-time format (section 2.2.1.3).

Additional XML data formats are specified in section 2.2.5.

2.2.5.18 MachineGroup Element

The MachineGroup XML element is used to specify parameters for managing machine groups. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="MachineGroup">
  <xs:complexType>
    <xs:sequence>
      <xs:element
        minOccurs="0"
        maxOccurs="unbounded"
        ref="Machine"
       />
      <xs:element
        minOccurs="0"
        maxOccurs="unbounded"
        ref="MachineGroup"
       />
    </xs:sequence>
    <xs:attribute name="Name"
      type="xs:string"
     />
    <xs:attribute name="Description"
      type="xs:string"
     />
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Machine	Machine	The parameters that define a machine.
MachineGroup	MachineGroup	The parameters that define a machine group.

Attributes

Name	Type	Description
Name	xs:string	The name of the machine group.
Description	xs:string	A description of the machine group.

Additional XML data formats are specified in section 2.2.5.

2.2.5.19 NotificationConfigInfo Element

The NotificationConfigInfo XML element is used to specify the notification configuration.

```
<xs:element name="NotificationConfigInfo">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="NotificationEnabled"
        type="xs:string"
        minOccurs="0"
       />
      <xs:element name="SMTPServer"
        type="xs:string"
        minOccurs="0"
       />
      <xs:element name="EmailIds"
        type="xs:string"
        minOccurs="0"
       />
      <xs:element name="EventList"
        type="xs:string"
        minOccurs="0"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
NotificationEnabled	xs:string	A string that specifies whether the notification configuration is currently enabled, in Boolean format (section 2.2.1.2).
SMTPServer	xs:string	A string that specifies the **NetBIOS** name or fully qualified domain name (FQDN) of the **Simple Mail Transfer Protocol (SMTP)** server.
EmailIds	xs:string	A string that specifies one or more email identifiers to be notified, separated by semicolons.
EventList	xs:string	A string that specifies a list of numeric **event identifiers**, separated by commas.

2.2.5.20 ObjectIds Element

The ObjectIds XML element is used to specify objects on the server. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="ObjectIds">
  <xs:complexType>
    <xs:choice
      minOccurs="0"
      maxOccurs="unbounded"
    >
      <xs:element name="ObjectId"
        nillable="true"
      >
        <xs:complexType name="ObjectId">
          <xs:simpleContent>
            <xs:extension
              base="xs:string"
            >
              <xs:attribute name="Type"
                type="xs:string"
               />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
ObjectId	ObjectId	The name of the object.

Attributes

Name	Type	Description
Type	xs:string	The type of the object, expressed as an object type (section 2.2.2.7).

Additional XML data formats are specified in section 2.2.5.

2.2.5.21 Policy Element

The Policy XML element specifies parameters for managing resource allocation. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="Policy">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AllocationCriteria"
        maxOccurs="unbounded"
      >
        <xs:complexType name="AllocationCriteria">
          <xs:sequence>
            <xs:element name="ProcessMatchingCriteria">
              <xs:complexType name="ProcessMatchingCriteria">
                <xs:attribute name="RefName"
                  type="xs:string"
                  use="required"
                 />
              </xs:complexType>
            </xs:element>
            <xs:element name="Affinity"
              type="xs:string"
```

```
              minOccurs="0"
             />
            <xs:element name="CPUAllocation"
              type="xs:unsignedByte"
             />
            <xs:element name="ManagementRule"
              type="xs:string"
              minOccurs="0"
             />
            <xs:element name="MaximumWorkingSet"
              type="xs:unsignedInt"
              minOccurs="0"
             />
            <xs:element name="MaximumCommittedMemory"
              type="xs:unsignedShort"
              minOccurs="0"
             />
            <xs:element name="CommittedMemoryExceededOption"
              type="xs:string"
              minOccurs="0"
             />
          </xs:sequence>
          <xs:attribute name="Name"
            type="xs:string"
            use="required"
           />
        </xs:complexType>
      </xs:element>
      <xs:element name="Description"
        type="xs:string"
        minOccurs="0"
       />
    </xs:sequence>
    <xs:attribute name="Name"
      type="xs:string"
      use="required"
     />
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
AllocationCriteria	AllocationCriteria	A specification of the allocation criteria for a policy, including the process matching criteria (PMC), resources and management regulations.
ProcessMatchingCriteria	ProcessMatchingCriteria	Specifies the name of the PMC being used.
Affinity	xs:string	The processor affinity on a multi-processor computer, in affinity format (section 2.2.1.1).
CPUAllocation	xs:unsignedByte	The percentage of CPU usage for this PMC.
ManagementRule	xs:string	The management rule for CPU allocation, expressed as a resource allocation option (section 2.2.2.8).
MaximumWorkingSet	xs:unsignedInt	The maximum amount of physical memory, in megabytes, that can be assigned to a process by the operating system.
MaximumCommittedMemory	xs:unsignedShort	An upper limit on the **committed memory**, in megabytes, that a process consumes. If this limit is exceeded, WSRM will take action as specified in

Element	Type	Description
		<CommittedMemoryExceededOption>.
CommittedMemoryExceededOption	xs:string	The action that will be taken if the <MaximumCommittedMemory> limit is exceeded, expressed as a memory option (section 2.2.2.6).
Description	xs:string	A text description of the policy.

Attributes

Name	Type	Description
RefName	xs:string	The name of the PMC.
Name	xs:string	The name of the <AllocationCriteria> element.<13>
Name	xs:string	The name of the policy.

Additional XML data formats are specified in section 2.2.5.

2.2.5.22 PolicyCollection Element

The PolicyCollection XML element specifies a collection of policy objects for managing resource allocation. It is the root element of XML documents that are used in WSRM method parameters, including methods of the IWRMPolicy interface (section 3.2.4.7).<14>

```
<xs:element name="PolicyCollection">
  <xs:complexType>
    <xs:sequence>
      <xs:element
        maxOccurs="unbounded"
        ref="Policy"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Policy	Policy	A specification of a Policy element (section 2.2.5.21) that defines a resource allocation policy (RAP).

Additional XML data formats are specified in section 2.2.5.

2.2.5.23 ProcessList Element

The ProcessList XML element specifies a list of processes and their properties and is used in the GetProcessList (Opnum 16) (section 3.2.4.7.10) method to return processes matched by the PMCs of a resource allocation policy (RAP). It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="ProcessList">
  <xs:complexType name="ProcessList">
```

```
<xs:sequence>
  <xs:element name="Process"
    minOccurs="0"
    maxOccurs="unbounded"
  >
    <xs:complexType name="Process">
      <xs:sequence>
        <xs:element name="Name"
          type="xs:unsignedInt"
         />
        <xs:element name="Image"
          type="xs:string"
         />
        <xs:element name="Path"
          type="xs:string"
         />
        <xs:element name="CommandLine"
          type="xs:string"
         />
        <xs:element name="User"
          type="xs:string"
         />
        <xs:element name="Domain"
          type="xs:string"
         />
        <xs:element name="CreationTime"
          type="xs:unsignedLong"
         />
        <xs:element name="UserTime"
          type="xs:unsignedLong"
         />
        <xs:element name="KernelTime"
          type="xs:unsignedLong"
         />
        <xs:element name="HandleCount"
          type="xs:unsignedShort"
         />
        <xs:element name="SessionId"
          type="xs:unsignedInt"
         />
        <xs:element name="NumberOfThreads"
          type="xs:unsignedByte"
         />
        <xs:element name="PeakVirtualSize"
          type="xs:unsignedInt"
         />
        <xs:element name="VirtualSize"
          type="xs:unsignedLong"
         />
        <xs:element name="PageFaultCount"
          type="xs:unsignedInt"
         />
        <xs:element name="PeakWorkingSetSize"
          type="xs:unsignedInt"
         />
        <xs:element name="WorkingSetSize"
          type="xs:unsignedLong"
         />
        <xs:element name="QuotaPeakPagedPoolUsage"
          type="xs:unsignedInt"
         />
        <xs:element name="QuotaPagedPoolUsage"
          type="xs:unsignedInt"
         />
        <xs:element name="QuotaPeakNonPagedPoolUsage"
          type="xs:unsignedInt"
         />
        <xs:element name="QuotaNonPagedPoolUsage"
          type="xs:unsignedInt"
         />
```

```
            <xs:element name="PageFileUsage"
              type="xs:unsignedInt"
             />
            <xs:element name="PeakPageFileUsage"
              type="xs:unsignedInt"
             />
            <xs:element name="PrivatePageCount"
              type="xs:unsignedLong"
             />
            <xs:element name="ReadOperationCount"
              type="xs:unsignedLong"
             />
            <xs:element name="WriteOperationCount"
              type="xs:unsignedLong"
             />
            <xs:element name="OtherOperationCount"
              type="xs:unsignedLong"
             />
            <xs:element name="ReadTransferCount"
              type="xs:unsignedLong"
             />
            <xs:element name="WriteTransferCount"
              type="xs:unsignedLong"
             />
            <xs:element name="OtherTransferCount"
              type="xs:unsignedLong"
             />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Process	Process	A specification of the properties of a process.
Name	xs:unsignedInt	A number specifying the process identifier of the process.
Image	xs:string	A string that specifies the image name.
Path	xs:string	A string that specifies the image path.
CommandLine	xs:string	A string that specifies the contents of the command line for invoking the process.
User	xs:string	A string that specifies the user name associated with the process.
Domain	xs:string	A string that specifies the domain name associated with the process.
CreationTime	xs:unsignedLong	A 64-bit unsigned integer that specifies the creation time of the process. Its lower 32 bits represent the **dwLowDateTime** field, and its higher 32 bits represent the **dwHighDateTime** field of the **FILETIME** structure as specified in [MS-DTYP] section 2.3.3.
UserTime	xs:unsignedLong	A 64-bit unsigned integer that specifies the length of time, in milliseconds, the process has executed in user mode.
KernelTime	xs:unsignedLong	A 64-bit unsigned integer that specifies the length of time,

Element	Type	Description
		in milliseconds, the process has executed in kernel mode.
HandleCount	xs:unsignedShort	A 16-bit unsigned integer that specifies the number of handles opened by the process.
SessionId	xs:unsignedInt	A32-bit unsigned integer that specifies a session id for the process.
NumberOfThreads	xs:unsignedByte	An 8-bit unsigned integer that specifies the number of threads associated with the process.
PeakVirtualSize	xs:unsignedInt	A 32-bit unsigned integer that specifies the maximum virtual memory size, in bytes, attained by the process.
VirtualSize	xs:unsignedLong	A 64-bit unsigned integer that specifies the virtual memory size, in bytes, attained by the process when properties of the process were recorded.
PageFaultCount	xs:unsignedInt	A 32-bit unsigned integer that specifies the number of page faults generated by the process.
PeakWorkingSetSize	xs:unsignedInt	A 32-bit unsigned integer that specifies the maximum working set size, in bytes, attained by the process.
WorkingSetSize	xs:unsignedLong	A 64-bit unsigned integer that specifies the working set size, in bytes, attained by the process.
QuotaPeakPagedPoolUsage	xs:unsignedInt	A 32-bit unsigned integer that specifies the maximum quota amount of paged pool usage, in kilobytes, for the process.
QuotaPagedPoolUsage	xs:unsignedInt	A 32-bit unsigned integer that specifies the quota amount of paged pool usage, in kilobytes, for the process.
QuotaPeakNonPagedPoolUsage	xs:unsignedInt	A 32-bit unsigned integer that specifies the maximum quota amount of non-paged pool usage, in kilobytes, for the process.
QuotaNonPagedPoolUsage	xs:unsignedInt	A 32-bit unsigned integer that specifies the quota amount of non-paged pool usage, in kilobytes, for the process.
PageFileUsage	xs:unsignedInt	A 32-bit unsigned integer that specifies the amount of the page file used, in kilobytes, by the process.
PeakPageFileUsage	xs:unsignedInt	A 32-bit unsigned integer that specifies the maximum amount of the page file used, in kilobytes, by the process.
PrivatePageCount	xs:unsignedLong	A 64-bit unsigned integer that specifies the size of private virtual memory pages used by the process.
ReadOperationCount	xs:unsignedLong	A 64-bit unsigned integer that specifies the number of read operations that have been performed by the process.
WriteOperationCount	xs:unsignedLong	A 64-bit unsigned integer that specifies the number of write operations that have been performed by the process.
OtherOperationCount	xs:unsignedLong	A 64-bit unsigned integer that specifies the number of non-read and non-write I/O operations--for example, I/O control functions--that have been performed by the process.
ReadTransferCount	xs:unsignedLong	A 64-bit unsigned integer that specifies the amount of data, in bytes, that has been read by the process.
WriteTransferCount	xs:unsignedLong	A 64-bit unsigned integer that specifies the amount of data,

Element	Type	Description
		in bytes, that has been written by the process.
OtherTransferCount	xs:unsignedLong	A 64-bit unsigned integer that specifies the amount of data, in bytes, that has been transferred by the process through non-read or non-write data transfers; an example would be data transferred for I/O control functions.

Additional XML data formats are specified in section 2.2.5.

2.2.5.24 ProcessMatchingCriteria Element

The ProcessMatchingCriteria XML element specifies parameters for resource management. It is the root element of XML documents that are used to specify process matching criteria (PMC) in WSRM method parameters.

```
<xs:element name="ProcessMatchingCriteria">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Rule">
        <xs:complexType name="Rule">
          <xs:sequence>
            <xs:element name="Path"
              type="xs:string"
             />
            <xs:element name="User"
              type="xs:string"
             />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Description"
        type="xs:string"
        minOccurs="0"
       />
    </xs:sequence>
    <xs:attribute name="Name"
      type="xs:string"
      use="required"
     />
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Rule	Rule	A tuple of the *Path* and *User* elements of the PMC.
Path	xs:string	A regular expression matching the path of a process's image file.
User	xs:string	A user or group name.
Description	xs:string	A text description of the PMC.

Attributes

Name	Type	Description
Name	xs:string	The name of the PMC.

Additional XML data formats are specified in section 2.2.5.

2.2.5.25 ProcessMatchingCriteriaCollection Element

The ProcessMatchingCriteriaCollection XML element specifies a collection of process matching criteria (PMC) objects for resource management. It is the root element of XML documents that are used to specify process matching criteria (PMC) in WSRM method parameters.<15>

```
<xs:element name="ProcessMatchingCriteriaCollection">
  <xs:complexType>
    <xs:sequence>
      <xs:element
        maxOccurs="unbounded"
        ref="ProcessMatchingCriteria"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
ProcessMatchingCriteria	ProcessMatchingCriteria	A specification of a ProcessMatchingCriteria element (section 2.2.5.24) that defines parameters for resource management.

Additional XML data formats are specified in section 2.2.5.

2.2.5.26 Schedule Element

The Schedule XML element specifies a schedule for a calendar event. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="Schedule">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ScheduleName"
        type="xs:string"
        minOccurs="1"
        maxOccurs="1"
       />
      <xs:element
        maxOccurs="unbounded"
        ref="CalendarEvent"
       />
      <xs:element name="Description"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
ScheduleName	xs:string	The name of the schedule.
CalendarEvent	CalendarEvent	A specification of a calendar event, in the form of a CalendarEvent element (section 2.2.5.8).
Description	xs:string	A text description of the schedule.

Additional XML data formats are specified in section 2.2.5.

2.2.5.27 Schedules Element

The Schedules XML element specifies one or more schedule objects. It is the root element of XML documents that are used in WSRM method parameters, including methods of the IWRMCalendar interface (section 3.2.4.4).

```
<xs:element name="Schedules">
  <xs:complexType>
    <xs:sequence>
      <xs:element
        maxOccurs="unbounded"
        ref="Schedule"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Schedule	Schedule	A specification of a Schedule element (section 2.2.5.26) that defines an individual schedule object.

Additional XML data formats are specified in section 2.2.5.

2.2.5.28 ServiceList Element

The ServiceList XML element is used to specify **services** that are registered with the server. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="ServiceList">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Service"
        maxOccurs="unbounded"
      >
        <xs:complexType name="Service">
          <xs:sequence>
            <xs:element name="Name"
              type="xs:string"
             />
            <xs:element name="StartName"
              type="xs:string"
             />
            <xs:element name="Path"
              type="xs:string"
             />
            <xs:element name="Description"
```

```
              type="xs:string"
             />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Service	Service	A specification of a service.
Name	xs:string	The name of the service.
StartName	xs:string	The name of the user account in which the service is started.
Path	xs:string	The path to the service executable binary, along with the command line for invoking the service.
Description	xs:string	A text description of the service.

Additional XML data formats are specified in section 2.2.5.

2.2.5.29 SupportedClients Element

The SupportedClients XML element is used to specify the level of support for clients on the WSRM server. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="SupportedClients">
  <xs:complexType>
    <xs:choice
      maxOccurs="unbounded"
      minOccurs="0"
    >
      <xs:element name="Client"
        nillable="true"
      >
        <xs:complexType name="Client">
          <xs:simpleContent>
            <xs:extension
              base="xs:string"
             />
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Client	Client	A string that identifies the level of support for clients, expressed as a supported clients option (section 2.2.2.9).

2.2.5.30 Users Element

The Users XML element is used to specify categories of remote session users. It is the root element of XML documents that are used in WSRM method parameters.

```
<xs:element name="Users">
  <xs:complexType name="Users">
    <xs:sequence>
      <xs:element name="Category"
        minOccurs="0"
        maxOccurs="unbounded"
      >
        <xs:complexType name="Category">
          <xs:sequence>
            <xs:element name="User"
              maxOccurs="unbounded"
              minOccurs="0"
            >
              <xs:complexType name="User">
                <xs:attribute name="DisplayName"
                  type="xs:string"
                 />
                <xs:attribute name="ID"
                  type="xs:string"
                 />
                <xs:attribute name="Domain"
                  type="xs:string"
                 />
                <xs:attribute name="Type"
                  type="xs:string"
                 />
                <xs:attribute name="Description"
                  type="xs:string"
                 />
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="Type"
            type="xs:string"
           />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type	Description
Category	Category	A specification of a group of users.
User	User	A specification of a set of attributes that describe a user.

Attributes

Name	Type	Description
DisplayName	xs:string	The display name of the user.
ID	xs:string	The **Active Directory** value for **sAMAccountName** of the user, as defined in [MS-ADA3] section 2.222.
Domain	xs:string	The **domain** of the user.

Name	Type	Description
Type	xs:string	The type of the user, expressed as a user type (section 2.2.2.10).
Description	xs:string	
Type	xs:string	The type of the category, expressed as a category type (section 2.2.2.2).

Additional XML data formats are specified in section 2.2.5.

3 Protocol Details

3.1 Client Role Details

3.1.1 Abstract Data Model

No abstract data model is defined.

The WSRM Protocol does not support notification callback, so no notification callback objects are defined.

3.1.2 Timers

No timers are required.

3.1.3 Initialization

The client MUST instantiate a **ResourceManager Component Object Model (COM)** object. A description of how to create COM objects is specified in [MS-DCOM] section 1.3.1.

After instantiating the ResourceManager object and acquiring pointers to its interfaces, a client calls methods of interest.

3.1.4 Message Processing Events and Sequencing Rules

3.1.4.1 Higher-Layer Triggered Events

All method invocations are triggered by higher-layer events, including commands issued within administrative and diagnostic applications. Method invocations are specified in Message Processing Events and Sequencing Rules (section 3.2.4) for the WSRM Protocol server.

3.1.4.2 Opening and Closing a Session

To open a session, the client MUST do the following, in sequence:

1. Create an object of **ResourceManager** COM class.
2. Acquire pointers to WSRM interfaces by using the IUnknown::QueryInterface method, as described in [MS-DCOM] section 3.1.1.5.8.
3. Call methods of interest in the interfaces.

To close a session, the client MUST release all of its acquired interface pointers by calling their **Release** methods.

3.1.4.3 Processing Server Replies

Upon receiving a reply from the server in response to a method call, the client MUST validate the return code. If the return value is 0x00000000,success is indicated, and the client can assume that all output parameters are present and valid. If the value returned from the server is a negative HRESULT value, an error is indicated.

The client MUST release any DCOM interfaces returned by the server when the client no longer has any use for them.

3.1.4.4 Processing Server Notifications

WSRM does not support notification callback; that is, the client receives no callback events issued by the server.

3.1.5 Timer Events

No timer events are used.

3.1.6 Other Local Events

No other local events require special processing on the client.

3.2 Server Role Details

The following types of tasks are executed on the WSRM Protocol server:

- Mapping processes to the PMC of the active policy.
- Calculating and managing the resources consumed by each process.
- Performing a configured action when the **working set** size crosses a specified limit.
- Switching management policies based on calendar, schedules, or other specified conditions.
- Periodically writing process counter records into the accounting database.
- Listening for commands to query, modify, configure, or create accounting database objects.
- Checking for the triggering of a conditional policy.
- Assigning resources to remote sessions according to configuration settings.

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation of the WSRM Protocol could maintain in order to participate in this protocol. This description is provided to facilitate the explanation of how the protocol behaves.

This specification does not mandate that implementations adhere to this model as long as their external behavior is consistent with that defined in this specification.

3.2.1.1 Data Structures

The WSRM Protocol maintains types of information as described by the following data structures:

- **Application pool list**: An information structure returned by the GetIISAppPoolNames (section 3.2.4.1.8) method. It contains a list of URLs in the Internet Information Services (IIS) application pool.<16>
- **Process list**: A list of processes that match a given set of PMCs.
- **Processor affinity mask**: A representation of processor affinity as a bit vector in which each bit represents one of the processors that are configured into a system.
- **Resource policy store**: The entire set of resource allocation policies (RAPs) and all objects that refer to or are referred to by those RAPs maintained by the **management service**. The resource policy store is persistent across invocations of the management service.

- **System directory**: An information structure returned by the GetIISAppPoolNames method. It defines the system directory used by Terminal Services.<17>

3.2.1.2 Accounting Database

The WSRM Protocol maintains information for each process it manages in an accounting database. It SHOULD<18> contain the following information. The "Name" column specifies the name of the AccountingProcessList element (section 2.2.5.4) to which the data corresponds.

Name	Description
CreationSystemTime	A 64-bit unsigned integer that specifies the time the data was entered into the database. Its lower 32 bits represent the **dwLowDateTime** field, and its higher 32 bits represent the **dwHighDateTime** field of the **FILETIME** structure as specified in [MS-DTYP] section 2.3.3.
CreationTime	A 64-bit unsigned integer that specifies the creation time of the process. Its lower 32 bits represent the **dwLowDateTime** field, and its higher 32 bits represent the **dwHighDateTime** field of the **FILETIME** structure as specified in [MS-DTYP] section 2.3.3.
DomainName	The Active Directory domain name of the user who created the process. If the computer is not in an Active Directory domain, the computer's **workgroup** is stored.
EventType	A string that specifies the event upon which properties of a process are recorded. The string can be "C", for process creation; "D", for process termination; and "L", for regular logging.
ImageName	The image name of the process associated with the accounting record.
ImagePath	The path to the directory where the executable file of the process is located.
KernelModeTime	The elapsed time, in 100-nanosecond units, that the process has spent in kernel mode.
OtherOperationCount	The number of input/output (I/O) operations that are neither read nor write operations (for example, a control function). This counter counts all I/O activity generated by the process, including file, network, and device operations.
OtherTransferCount	The number of bytes transferred in I/O operations that are neither read nor write (for example, a control function), including file, network, and device operations.
PageFaultCount	The number of page faults. A page fault occurs when a thread refers to a virtual memory page that is not in its working set in main memory and ~~must~~needs to be retrieved from the disk. If the page is being used by another process that shares the page, or if the page is already in main memory but is on the standby list, the page might not be found on the disk.
PageFileUsage	The current amount of virtual memory, in kilobytes, that this process has reserved for use in paging files. Paging files store pages of memory that are used by the process and that are not contained in other files. Paging files are shared by all processes. The lack of space in paging files can prevent other processes from allocating memory. If there is no paging file, this counter reflects the current amount of virtual memory that the process has reserved for use in physical memory.
ParentProcessId	The <ProcessId> of the parent process.
PeakPageFileUsage	The maximum amount of virtual memory, in kilobytes, that this process has reserved for use in paging files.

Name	Description
PeakVirtualSize	Maximum virtual address space, in bytes, that a process uses at any one time.
PeakWorkingSetSize	Peak working set size, in bytes, of a process.
PolicyName	The name of the resource allocation policy (RAP) associated with the accounting data.
PolicySetTime	A 64-bit unsigned integer that specifies the time that the RAP was set as the current resource policy. Its lower 32 bits represent the **dwLowDateTime** field, and its higher 32 bits represent the **dwHighDateTime** field of the **FILETIME** structure as specified in [MS-DTYP] section 2.3.3.
PrivatePageCount	The current number of pages allocated that are accessible only to this process.
ProcessCommandLine	The command used to start the process. This is a string of text written in the command language and passed to the command interpreter for execution. This is usually, but not always, the same as the program path.
ProcessId	A numerical identifier that uniquely distinguishes a process while it runs.
QuotaNonPagedPoolUsage	Current nonpaged pool usage for the process, in kilobytes.
QuotaPagedPoolUsage	Current paged pool usage for the process, in kilobytes.
QuotaPeakNonPagedPoolUsage	Peak nonpaged pool usage for the process, in kilobytes.
QuotaPeakPagedPoolUsage	Peak paged pool usage for the process, in kilobytes.
ReadOperationCount	The number of read I/O operations generated by the process, including file, network, and device operations.
ReadTransferCount	The number of bytes read in I/O operations, including file, network, and device operations.
ResourceGroupName	The resource group that was in use when the process started.
SessionId	The **session identifier** of the session that owns the process.
ThreadCount	The number of threads currently active in this process. An instruction is the basic unit of execution in a processor. A thread is the object that executes instructions. Every running process has at least one thread.
UserModeTime	The elapsed time, in 100-nanosecond units, that the processor has spent in user mode. User mode is a restricted processing mode, designed for applications, environment subsystems, and integral subsystems. The alternative is kernel mode, which is designed for operating system components. Kernel mode allows direct access to hardware and all memory. The operating system switches application threads to kernel mode in order to access operating system services. This value counts the average busy time as a percentage of the sample time.
UserName	The name of the user who started the process.
VirtualSize	The current size, in bytes, of the virtual address space the process is using. Use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages. Virtual space is finite, and the process can limit its ability to load libraries.
WorkingSetSize	The current size, in bytes, of the working set of this process.
WriteOperationCount	The number of write I/O operations generated by the process, including file, network, and device operations.

Name	Description
WriteTransferCount	The number of bytes written in write I/O operations generated by a process, including file, network, and device operations.

The following database fields are not used in the AccountingProcessList element.

Name	Description
ComputerName	The name of the server where data was collected.
EndTime	A 64-bit unsigned integer that specifies the time when the process terminated. Its lower 32 bits represent the **dwLowDateTime** field, and its higher 32 bits represent the **dwHighDateTime** field of the **FILETIME** structure as specified in [MS-DTYP] section 2.3.3.
TotalCPU	A 64-bit unsigned integer value that specifies the length of time, in 100-nanosecond units, that the process has executed in kernel mode or user mode.
ElapsedTime	A 64-bit unsigned integer value that specifies the length of time, in 100-nanosecond units, that the process has been executing.
GroupId	A numerical identifier that uniquely identifies a raw accounting data record.

3.2.2 Timers

Timers are not required for this interface.

3.2.3 Initialization

At startup, the server registers the Component Object Model (COM) interfaces. At the initialization, the **WSRM management service** MUST attempt the following tasks:

- Restore its prior **management state**; failing to do so SHOULD default to the active or running management state.
- The WSRM management service MAY have built-in policies. The WSRM server SHOULD load built-in policies, which can be returned in methods of IWRMPolicy; for example, GetPolicyInfo, SetCurrentPolicy, and so on.<19>
- If there is no machine group in the configuration, the management service MUST create the first machine group with the name "Default" and the description "Default machine group", without any machine group or machine as its children.

3.2.4 Message Processing Events and Sequencing Rules

The WSRM Protocol server exposes DCOM Protocol [MS-DCOM] interfaces for interacting with clients<20>. The server MUST implement the following interfaces:

Interface	Description
IResourceManager	Performs administrative tasks and retrieves system information from the computer being managed.
IResourceManager2	Imports and exports objects with smart import-conflict resolution.<21>
IWRMAccounting	Performs configuration tasks on accounting data.

Interface	Description
IWRMCalendar	Schedules resource management operations.
IWRMConfig	Manages the WSRM configuration of server resources.
IWRMMachineGroup	Creates, manages, and deletes machine groups.<22>
IWRMPolicy	Manages objects that control the allocation of resources to user tasks.
IWRMProtocol	Manages client versions supported on the server.
IWRMRemoteSessionMgmt	Manages user category and remote session information.<23>
IWRMResourceGroup	Retrieves, modifies, and deletes resource groups.

These interfaces are specified in the sections that follow.

3.2.4.1 IResourceManager Interface

The IResourceManager interface defines the following methods.

Methods in RPC Opnum Order

Method	Description
RetrieveEventList	Is not implemented and always returns an error. Opnum: 7
GetSystemAffinity	Get the processor affinity of the system. Opnum: 8
ImportXMLFiles	Loads a specified WSRM configuration. Opnum: 9
ExportXMLFiles	Saves the current WSRM configuration. Opnum: 10
RestoreXMLFiles	Restores the WSRM configuration to a specified state. Opnum: 11
GetDependencies	Provides a list of WSRM objects that are being used or that depend on the specified object. Opnum: 12
GetServiceList	Provides a list of registered services that are registered with the server that are not excluded by the exclusion list. Opnum: 13
GetIISAppPoolNames	Returns a list of all the application pools on the WSRM server that are defined by and known to the local IIS server.<24> Opnum: 14
GetServerName	Returns the server name.<25> Opnum: 15
GetCurrentMemory	Determines the total amount of physical memory in the system.<26> Opnum: 16

3.2.4.1.1 RetrieveEventList (Opnum 7)

The RetrieveEventList method is not implemented and always returns an error.

```
[id(1), helpstring("method RetrieveEventList")] HRESULT RetrieveEventList(
  [out] BSTR~~* bstrEventList~~ *pbstrEventList
);
```

~~bstrEventList~~pbstrEventList: Not used.

Return Values: This method returns **E_NOTIMPL**, unless a parameter error is found.

Return value/code	Description
0x80004001 E_NOTIMPL	This function is not implemented.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.1.2 GetSystemAffinity (Opnum 8)

The GetSystemAffinity method gets the processor affinity of the system.

```
[id(2), helpstring("method GetSystemAffinity")] HRESULT GetSystemAffinity(
  [out] DWORD64* pdwSysAffinity
);
```

pdwSysAffinity: A pointer to a 64-bit unsigned integer that returns a processor affinity information structure, which is a bit vector where each bit represents the processors that are configured into a system.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.1.3 ImportXMLFiles (Opnum 9)

The **ImportXMLFiles** method loads a specified WSRM configuration.

```
[id(3), helpstring("method ImportXMLFiles")] HRESULT ImportXMLFiles(
  [in] BSTR bstrPMCXml,
  [in] BSTR bstrPolicyXml,
  [in] BSTR bstrCalendarXml,
  [in] BSTR bstrConditionalXml
);
```

bstrPMCXml: A string that specifies the PMCs to be loaded by the server, in the form of a ProcessMatchingCriteriaCollection element (section 2.2.5.25). For an example, see ProcessMatchingCriteriaCollection example (section 4.2.21). This parameter is ignored if it is NULL.

bstrPolicyXml: A string that specifies resource policies to be loaded by the server, in the form of a PolicyCollection element (section 2.2.5.22). This parameter is ignored if it is NULL.

bstrCalendarXml: A string that specifies calendars to be loaded by the server, in the form of a CalendarsCollection element (section 2.2.5.11). For an example, see CalendarsCollection example (section 4.2.8). This parameter is ignored if it is NULL.

bstrConditionalXml: A string that specifies conditions to be loaded by the server, in the form of a ConditionalPolicy element (section 2.2.5.12). For an example, see section 4.2.9. This parameter is ignored if it is NULL.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<27>

The ImportXMLFiles method can be used to manage system resources by importing a valid RAP.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.1.4 ExportXMLFiles (Opnum 10)

The ExportXMLFiles method saves the current WSRM configuration.<28>

```
[id(4), helpstring("method ExportXMLFiles")] HRESULT ExportXMLFiles(
  [out] BSTR* pbstrPMCXml,
  [out] BSTR* pbstrPolicyXml,
  [out] BSTR* pbstrCalendarXml,
  [out] BSTR* pbstrConditionalXml
);
```

pbstrPMCXml: A pointer to a string that returns process matching criteria (PMC) in the form of a ProcessMatchingCriteriaCollection element (section 2.2.5.25). For an example, see ProcessMatchingCriteriaCollection example (section 4.2.21).

pbstrPolicyXml: A pointer to a string that returns a resource policy in the form of a PolicyCollection element (section 2.2.5.22).

pbstrCalendarXml: A pointer to a string that returns a calendar in the form of a CalendarsCollection element (section 2.2.5.11). For an example, see CalendarsCollection Example (section 4.2.8).

pbstrConditionalXml: A pointer to a string that returns a condition in the form of a ConditionalPolicy element (section 2.2.5.12). For an example, see section 4.2.9.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<29>
0x00000000 S_OK	Operation successful.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.1.5 RestoreXMLFiles (Opnum 11)

The RestoreXMLFiles method restores the WSRM configuration to a specified state.

```
[id(5), helpstring("method RestoreXMLFiles")] HRESULT RestoreXMLFiles(
  [in] RESTORE_MODE enumRestore
);
```

enumRestore: A value that specifies the state to restore.

This parameter MUST be defined in the RESTORE_MODE enumeration (section 2.2.3.7); otherwise, **E_INVALIDARG** MUST be returned.<30>

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<31>
0x80070057 E_INVALIDARG	One or more arguments are invalid.

This method can be used to reset the WSRM server state to a known value in case some data becomes corrupted and the server cannot proceed.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.1.6 GetDependencies (Opnum 12)

The GetDependencies method returns a list of WSRM objects that are being used or that depend on a specified object.

```
[id(6), helpstring("method GetDependencies")] HRESULT GetDependencies(
  [in] BSTR bstrObjectName,
  [in] OBJECT_TYPE enumObject,
  [out] BSTR* pbstrDependencyList
);
```

bstrObjectName: A string that specifies the name of the object whose dependencies are to be returned.

enumObject: An OBJECT_TYPE enumeration (section 2.2.3.6) value that specifies the type of object specified in *bstrObjectName*.

pbstrDependencyList: A pointer to a string that returns the dependencies for the object specified in *bstrObjectName*, in the form of a DependencyList element (section 2.2.5.14). For an example, see section 4.2.11.

A DependencyList element can specify policies, schedules, calendars, and conditional policy events. The content of the structure that is returned by this method is determined by the value of the *enumObject* parameter.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-") and cannot contain spaces or any of the following characters: `\ / ? * | : < > " , ;.`
0x80004005 E_FAIL	Either an object with the specified name and type was not found or its dependency list could not be created.
0xC1FF0271 WRM_ERR_CAL_UNKNOWN_SCHEDULE	A calendar name or an invalid schedule name was passed in *bstrObjectName* for OBJECT_SCHEDULE enumObject.
0xC1FF013A WRM_ERR_DEPENDENCIES_FOR_RESIDUAL	This is a **residual PMC** and is a part of all policies. All processes that do not match any of the PMC specified by a user in a policy automatically match to residual PMC.

The server SHOULD process this method call as follows:

- If a PMC object is specified, this method MUST return a list of policies that make use of the PMC.
- If a **resource policy object** is specified, this method MUST return a list of calendar events, conditions, and schedules that make use of the resource policy.
- If a schedule object is specified, this method MUST return a list of calendar events making use of the schedule.
- If a calendar object is specified, this method MUST return WRM_ERR_CAL_UNKNOWN_SCHEDULE.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.1.7 GetServiceList (Opnum 13)

The GetServiceList method provides a list of services that are registered with the server and are not excluded by the exclusion list. This list of services is expected to be used for defining process matching criteria (PMC).

```
[id(7), helpstring("method GetServiceList")] HRESULT GetServiceList(
```

```
  [out] BSTR* pbstrServiceList
);
```

pbstrServiceList: A pointer to a string that returns a list of services, in the form of a ServiceList element (section 2.2.5.28). For an example, see section 4.2.23.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.1.8 GetIISAppPoolNames (Opnum 14)

The GetIISAppPoolNames method returns a list of all the application pools on the WSRM server that are defined by and known to the IIS server.

```
[id(8), helpstring("method GetIISAppPoolNames")] HRESULT GetIISAppPoolNames(
  [out] BSTR* pbstrIISAppPoolList,
  [out] BSTR* pbstrSystemDirectory
);
```

pbstrIISAppPoolList: A pointer to a string in the format of the AppPoolList element (section 2.2.5.6) that returns an IIS application pool list structure.

pbstrSystemDirectory: A pointer to a string that returns implementation-specific system directory information.<32>

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.1.9 GetServerName (Opnum 15)

The GetServerName method returns the server name.

```
[id(9), helpstring("method GetServerName")] HRESULT GetServerName(
  [out] BSTR* pbstrServerName
);
```

pbstrServerName: A pointer to a string that returns the server name.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.<33>

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.1.10 GetCurrentMemory (Opnum 16)

The GetCurrentMemory method determines the total amount of physical memory in the system.

```
[id(10), helpstring("method GetCurrentMemory")] HRESULT GetCurrentMemory(
  [out] DWORD64* pdwCurrMemory
);
```

pdwCurrMemory: A pointer to a 64-bit unsigned integer that returns the current system memory size, in bytes.

Return Values: This method returns 0x00000001 for success or 0x00000000 if an error occurs.

Return value/code	Description
0x00000000	An error occurred.
0x00000001 S_OK	Operation successful.

Additional IResourceManager interface methods are specified in section 3.2.4.1.

3.2.4.2 IResourceManager2 Interface

The IResourceManager2 interface defines the following methods.<34>

Methods in RPC Opnum Order

Method	Description
ExportObjects	Creates XML for exporting objects. Opnum: 7
GetImportConflicts	Finds conflicts between import objects and existing objects. Opnum: 8
ImportXml	Imports objects into the configuration. Opnum: 9
ExportXml	Exports objects from the configuration. Opnum: 10

3.2.4.2.1 ExportObjects (Opnum 7)

The ExportObjects method creates XML for exporting objects.

```
[id(1), helpstring("method ExportObjects")] HRESULT ExportObjects(
```

```
    [in] BSTR bstrObjectIds,
    [in] OBJECT_TYPE enumObjectType,
    [out] BSTR* pbstrObjectXml
);
```

bstrObjectIds: A string that identifies the objects to be exported, in the format of an ObjectIds element (section 2.2.5.20).

enumObjectType: An OBJECT_TYPE enumeration value (section 2.2.3.6) that specifies the type of the objects to be exported. This determines the format of XML object returned in the *pbstrObjectXml* parameter, as follows.

Object type	XML object
OBJECT_SELECTION_CRITERIA	ProcessMatchingCriteria element (section 2.2.5.24)
OBJECT_POLICY	Policy element (section 2.2.5.21)
OBJECT_SCHEDULE	Calendar element (section 2.2.5.7) or Schedule element (section 2.2.5.26)

pbstrObjectXml: A pointer to a string that returns the XML for the objects to be exported. The format of the XML depends on the type of exported objects specified by the *enumObjectType* parameter.<35>

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Additional IResourceManager2 interface methods are specified in section 3.2.4.2.

The server SHOULD process this method call as follows.

- If one or more ObjectIds XML elements included in the *bstrObjectIds* parameter cannot be found, they MUST be ignored, and the call MUST proceed for other ObjectIds.
- If a PMC object is specified in the *enumObjectType* parameter, this method MUST return an XML string containing the requested PMCs in the configuration. The "Type" attributes of individual objects in the ObjectIds XML element MUST be ignored.
- If a resource policy object is specified in the *enumObjectType* parameter, this method MUST return an XML string containing the requested RAPs in the configuration. The "Type" attributes of individual objects in the ObjectIds XML element MUST be ignored.
- If a schedule object is specified in the *enumObjectType* parameter and the "Type" attribute of the object requested in the ObjectIds XML element is "Calendar", this method MUST return an XML string containing the requested calendar objects in the configuration.
- If a schedule object is specified in the *enumObjectType* parameter and the "Type" attribute of the object requested in the ObjectIds XML element is not "Calendar", this method MUST return an XML string containing the requested schedule objects in the configuration.

3.2.4.2.2 GetImportConflicts (Opnum 8)

The GetImportConflicts method finds conflicts between import objects and existing objects.

```
[id(2), helpstring("method GetImportConflicts")] HRESULT GetImportConflicts(
  [in] BSTR bstrPMCXml,
  [in] BSTR bstrPolicyXml,
  [in] BSTR bstrCalendarXml,
  [in] BSTR bstrConditionalXml,
  [in] BSTR bstrMachineGroupXml,
  [in] BSTR ~~bstrConfigurationXml~~bstrConfigurationXmls,
  [out] BSTR~~*~~ *pbstrConflictingObjects
);
```

bstrPMCXml: An optional string that specifies process matching criteria (PMC), in the form of a ProcessMatchingCriteriaCollection element (section 2.2.5.25).

bstrPolicyXml: An optional string that specifies resource policies, in the form of a PolicyCollection element (section 2.2.5.22).

bstrCalendarXml: An optional string that specifies calendar elements, in the form of a CalendarsCollection element (section 2.2.5.11).

bstrConditionalXml: An optional string that specifies a conditional policy, in the form of a ConditionalPolicy element (section 2.2.5.12).

bstrMachineGroupXml: An optional string that specifies a machine group, in the form of a MachineGroup element (section 2.2.5.18).

~~bstrConfigurationXml~~bstrConfigurationXmls: Not used.

pbstrConflictingObjects: A pointer to a string that SHOULD identify the conflicting objects, in the form of an ObjectIds element (section 2.2.5.20).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<36>

Additional IResourceManager2 interface methods are specified in section 3.2.4.2.

3.2.4.2.3 ImportXml (Opnum 9)

The ImportXml method imports objects into the configuration.

```
[id(3), helpstring("method ImportXml")] HRESULT ImportXml(
  [in] BSTR bstrPMCXml,
  [in] BSTR bstrPolicyXml,
  [in] BSTR bstrCalendarXml,
  [in] BSTR bstrConditionalXml,
  [in] BSTR bstrMachineGroupXml,
```

```
    [in] BSTR bstrConfigurationXmlbstrConfigurationXmls,
    [in] IMPORT_TYPE enumImportType
);
```

bstrPMCXml: A string that specifies process matching criteria (PMC), in the form of a ProcessMatchingCriteriaCollection element (section 2.2.5.25). For an example, see ProcessMatchingCriteriaCollection example (section 4.2.21).

bstrPolicyXml: A string that specifies a resource policy, in the form of a PolicyCollection element (section 2.2.5.22).

bstrCalendarXml: A string that specifies a calendar, in the form of a CalendarsCollection element (section 2.2.5.11). For an example, see CalendarsCollection example (section 4.2.8).<37>

bstrConditionalXml: A string that specifies a conditional policy, in the form of a ConditionalPolicy element (section 2.2.5.12).

bstrMachineGroupXml: A string that specifies a machine group, in the form of a MachineGroup element (section 2.2.5.17).

bstrConfigurationXmlbstrConfigurationXmls: A string that specifies a configuration to be loaded by the WSRM server, in the form of a ConfigurationFiles element (section 2.2.5.13).

enumImportType: An IMPORT_TYPE enumeration value (section 2.2.3.6) that specifies the mode in which to handle conflicting objects.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0272 WRM_ERR_CAL_SCHEDULE_IN_USE	An existing schedule with the same name as the one in the supplied CalendarsCollection is currently in use with an existing calendar. The complete import process is canceled.<38>
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<39>

Additional IResourceManager2 interface methods are specified in section 3.2.4.2.

3.2.4.2.4 ExportXml (Opnum 10)

The ExportXml method exports objects from the configuration.

```
[id(4), helpstring("method ExportXml")] HRESULT ExportXml(
    [out] BSTR* bstrPMCXml *pbstrPMCXml,
    [out] BSTR* bstrPolicyXml *pbstrPolicyXml,
    [out] BSTR* bstrCalendarXml *pbstrCalendarXml,
    [out] BSTR* bstrConditionalXml *pbstrConditionalXml,
    [out] BSTR* bstrMachineGroupXml *pbstrMachineGroupXml,
    [out] BSTR* bstrConfigurationXml *pbstrConfigurationXmls
);
```

~~bstrPMCXml~~pbstrPMCXml: A pointer to a string that returns process matching criteria (PMC), in the form of a ProcessMatchingCriteriaCollection element (section 2.2.5.25). For an example, see ProcessMatchingCriteriaCollection example (section 4.2.21).

~~bstrPolicyXml~~pbstrPolicyXml: A pointer to a string that returns a resource policy, in the form of a PolicyCollection element (section 2.2.5.22).

~~bstrCalendarXml~~pbstrCalendarXml: A pointer to a string that returns a calendar, in the form of a CalendarsCollection element (section 2.2.5.11). For an example, see CalendarsCollection example (section 4.2.8).

~~bstrConditionalXml~~pbstrConditionalXml: A pointer to a string that SHOULD return a conditional policy, in the format of a ConditionalPolicy element (section 2.2.5.12).

~~bstrMachineGroupXml~~pbstrMachineGroupXml: A pointer to a string that SHOULD return a machine group, in the format of a MachineGroup element (section 2.2.5.17).

~~bstrConfigurationXml~~pbstrConfigurationXmls: A pointer to a string that SHOULD return a configuration, in the format of a ConfigurationFiles element (section 2.2.5.13).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IResourceManager2 interface methods are specified in section 3.2.4.2.

3.2.4.3 IWRMAccounting Interface

The IWRMAccounting interface defines the following methods.

Methods in RPC Opnum Order

Method	Description
CreateAccountingDb	Creates the accounting database on a specified machine at runtime. Opnum: 7
GetAccountingMetadata	Retrieves accounting metadata. Opnum: 8
ExecuteAccountingQuery	Executes an accounting query. Opnum: 9
GetRawAccountingData	Returns raw accounting data. Opnum: 10
GetNextAccountingDataBatch	Gets a batch of accounting data. Opnum: 11
DeleteAccountingData	Deletes accounting data within a specified time period. Opnum: 12
DefragmentDB	Not implemented. Opnum: 13

Method	Description
CancelAccountingQuery	Cancels a query currently being made to the accounting database. Opnum: 14
RegisterAccountingClient	Registers a client for remote accounting on a server. Opnum: 15
DumpAccountingData	Dumps the accounting data of the server. Opnum: 16
GetAccountingClients	Gets the names of all servers that are acting as accounting clients on this server. Opnum: 17
SetAccountingClientStatus	Sets the status of servers that are in the remote client role. Opnum: 18
CheckAccountingConnection	Determines the status of the connection to the **accounting server**. Opnum: 19
SetClientPermissions	Adds or removes a specified client for remote accounting on a server and makes the required changes for DCOM permissions. Opnum: 20

With the WSRM Protocol, it is possible for one WSRM server, acting as an accounting client, to log accounting data on another WSRM server, acting as the accounting server. The accounting client dumps the accounting data to the database by calling the WSRM management service on the accounting server.

Methods in the **IWRMAccounting Interface** can be called by the WSRM management service to manage remote accounting as follows:

- RegisterAccountingClient can register the remote server as an accounting client when accounting is enabled for the first time.
- DumpAccountingData can pass accounting data from a remote server acting as an accounting client to the current accounting server.
- CheckAccountingConnection can check the connection to the accounting server.
- SetClientPermissions can add or remove an accounting client from the DCOM users group on the accounting server.

Note Remote accounting is not supported in a workgroup environment.

3.2.4.3.1 CreateAccountingDb (Opnum 7)

The CreateAccountingDb method creates the database for accounting data.<40>

```
[id(1), helpstring("method CreateAccountingDb")] HRESULT CreateAccountingDb(
  [in] BSTR bstrServerName,
  [in] BOOL bWindowsAuth,
  [in] BSTR bstrUserName,
  [in] BSTR bstrPasswd
);
```

bstrServerName: Name of the server where the accounting database MUST be created.

bWindowsAuth: This parameter MUST be ignored.

bstrUserName: User name for creating the database.

bstrPasswd: Password of the user.

Return Values: This method returns 0x00000000 for success, or a negative **HRESULT** value (shown in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation is successful.
0xC1FF01F7 WRM_ERR_ACCOUNTING_FAILED	WSRM encountered an error in accounting.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.2 GetAccountingMetadata (Opnum 8)

The GetAccountingMetadata method retrieves accounting metadata, which includes column names, types, and other attributes of the accounting tables.

```
[id(2), helpstring("method GetAccountingMetadata")] HRESULT GetAccountingMetadata(
  [out] BSTR* pbstrMetaData
);
```

pbstrMetaData: A pointer to a string that returns accounting metadata in the form of an AccountingMetaData XML element (section 2.2.5.3).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (int the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.3 ExecuteAccountingQuery (Opnum 9)

The ExecuteAccountingQuery method executes an accounting query.

```
[id(3), helpstring("method ExecuteAccountingQuery")] HRESULT ExecuteAccountingQuery(
  [in] BSTR bstrAccountingQuery,
  [in] BSTR bstrStartingDate,
  [in] BSTR bstrEndingDate,
  [out] BSTR* pbstrResult,
  [out] BOOL* pbIsThereMoreData
);
```

bstrAccountingQuery: A string that specifies an AccountingQueryCondition element (section 2.2.5.5) in XML. For an example, see section 4.2.5.

bstrStartingDate: A string that specifies the starting date for the query, in date-and-time format (section 2.2.1.3). If this value is not in the correct format, the date range is ignored and the complete set of accounting data is returned.

bstrEndingDate: A string that specifies the ending date for the query, in date-and-time format. If this value is not in the correct format, the date range is ignored and the complete set of accounting data is returned.

pbstrResult: A pointer to a string that returns the requested data.

This string is formatted as a set of rows representing accounting process entries. Rows are delimited by carriage return escape characters (\r). Each row is a set of columns delimited by newline escape characters (\n). The columns correspond in number and order to the columns specified in the AccountingQueryCondition element in the *bstrAccountingQuery* parameter.

pbIsThereMoreData: A pointer to a Boolean value that specifies whether more data is available.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF01FA WRM_ERR_WYUKON_NOT_CONNECTABLE	Cannot establish a connection to the accounting database.<41>
0xC1FF01FE WRM_ERR_JET_INVALID_COLUMN_NAME	The query has been canceled. One or more of the column names specified in the accounting query are invalid.
0xC1FF0200 WRM_ERR_JET_PIVOTABLE_COLUMN_NOT_GROUPED_BY	One or more SQL **SELECT** columns cannot be selected because of the current SQL **GROUP BY** settings. These columns MUST be grouped to be selected.
0xC1FF0201 WRM_ERR_JET_INVALID_GROUP_BY_COL	One or more columns specified for SQL **GROUP BY** is either invalid or cannot be grouped on.
0xC1FF0203 WRM_ERR_JET_SERVER_TOO_BUSY	The server can service only one accounting request at a time.<42>
0xC1FF0207 WRM_ERR_JET_SERVICE_BEING_SHUT_DOWN	The query has been aborted since the management service is being shut down.

The error **WRM_ERR_JET_PIVOTABLE_COLUMN_NOT_GROUPED_BY** is returned in cases where a column with the **IsVisible** attribute set to FALSE is included in the SQL **SELECT** column while there are some columns in the group column collection. The following sample AccountingQueryCondition XML (section 2.2.5.5) SHOULD return this error:

```
<AccountingQueryCondition>
    <SelectFieldCollection>
        <Column>ProcessName</Column>
        <Column>ProcessId</Column>
    </SelectFieldCollection>
    <GroupColumnCollection>
        <Column>ProcessName</Column>
    </GroupColumnCollection>
    <OrderColumnCollection />
    <HavingClause />
```

```
</AccountingQueryCondition>
```

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.4 GetRawAccountingData (Opnum 10)

The GetRawAccountingData method returns raw accounting data from the accounting database (section 3.2.1.2).

```
[id(4), helpstring("method GetRawAccountingData")] HRESULT GetRawAccountingData(
  [in] BSTR bstrStartingDate,
  [in] BSTR bstrEndingDate,
  [in] BSTR bstrMachineName,
  [out] BSTR* pbstrResult,
  [out] BOOL* pbIsThereMoreData
);
```

bstrStartingDate: A string that specifies the starting date of the data, in date-and-time format (section 2.2.1.3). If this value is not in the correct format, the date range is ignored and the complete set of accounting data is returned.

bstrEndingDate: A string that specifies the ending date of the data, in date-and-time format. If this value is not in the correct format, the date range is ignored and the complete set of accounting data is returned.

bstrMachineName: A string that specifies the machine name of the accounting database server. A default accounting database SHOULD<43> be defined.

pbstrResult: A pointer to a string that returns raw accounting data.

The string is formatted as a set of rows representing accounting process entries. Rows are delimited by carriage return escape characters (\r). Each row of raw accounting data contains 38 columns delimited by newline escape characters (\n).

The following table lists the 38 columns in order of their position in the row.

Column	Description
TimeStamp	See CreationSystemTime of accounting database in section 3.2.1.2.
ComputerName	See ComputerName of accounting database in section 3.2.1.2.
ProcessId	See ProcessId of accounting database in section 3.2.1.2.
ProcessName	See ImageName of accounting database in section 3.2.1.2.
Domain	See DomainName of accounting database in section 3.2.1.2.
User	See UserName of accounting database in section 3.2.1.2.
PolicyName	See PolicyName of accounting database in section 3.2.1.2.
PolicySetTime	See PolicySetTime of accounting database in section 3.2.1.2.
ProcessMatchingCriteria	See ResourceGroupName of accounting database in section 3.2.1.2.
CreationTime	See CreationTime of accounting database in section 3.2.1.2.
EndTime	See EndTime of accounting database in section 3.2.1.2.

Column	Description
ProgramPath	See ImagePath of accounting database in section 3.2.1.2.
CommandLine	See ProcessCommandLine of accounting database in section 3.2.1.2.
SessionId	See SessionId of accounting database in section 3.2.1.2.
ThreadCount	See ThreadCount of accounting database in section 3.2.1.2.
TotalCPU	See TotalCPU of accounting database in section 3.2.1.2.
ElapsedTime	See ElapsedTime of accounting database in section 3.2.1.2.
KernelModeTime	See KernelModeTime of accounting database in section 3.2.1.2.
UserModeTime	See UserModeTime of accounting database in section 3.2.1.2.
PageFileUsage	See PageFileUsage of accounting database in section 3.2.1.2.
PeakPageFileUsage	See PeakPageFileUsage of accounting database in section 3.2.1.2.
PageFaultCount	See PageFaultCount of accounting database in section 3.2.1.2.
VirtualSize	See VirtualSize of accounting database in section 3.2.1.2.
PeakVirtualSize	See PeakVirtualSize of accounting database in section 3.2.1.2.
WorkingSetSize	See WorkingSetSize of accounting database in section 3.2.1.2.
PeakWorkingSetSize	See PeakWorkingSetSize of accounting database in section 3.2.1.2.
PrivatePageCount	See PrivatePageCount of accounting database in section 3.2.1.2.
QuotaPagedPoolUsage	See QuotaPagedPoolUsage of accounting database in section 3.2.1.2.
QuotaPeakPagedPoolUsage	See QuotaPeakPagedPoolUsage of accounting database in section 3.2.1.2.
QuotaNonPagedPoolUsage	See QuotaNonPagedPoolUsage of accounting database in section 3.2.1.2.
QuotaPeakNonPagedPoolUsage	See QuotaPeakNonPagedPoolUsage of accounting database in section 3.2.1.2.
ReadOperationCount	See ReadOperationCount of accounting database in section 3.2.1.2.
ReadTransferCount	See ReadTransferCount of accounting database in section 3.2.1.2.
WriteOperationCount	See WriteOperationCount of accounting database in section 3.2.1.2.
WriteTransferCount	See WriteTransferCount of accounting database in section 3.2.1.2.
OtherOperationCount	See OtherOperationCount of accounting database in section 3.2.1.2.
OtherTransferCount	See OtherTransferCount of accounting database in section 3.2.1.2.
GroupId	See GroupId of accounting database in section 3.2.1.2.

pbIsThereMoreData: A pointer to a Boolean value that returns whether more data is available.

Return Values: This method returns 0x00000000 for success, or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF01F7 WRM_ERR_ACCOUNTING_FAILED	WSRM encountered an error in accounting.
0xC1FF01FA WRM_ERR_WYUKON_NOT_CONNECTABLE	Cannot establish a connection to the accounting database due to an error other than one of the errors of the WRM_ERR_WYUKON_CORRUPTED return value.
0xC1FF01FB WRM_ERR_WYUKON_CORRUPTED	Cannot establish a connection to the accounting database; either the database is in single-user mode and already connected, or it is in an invalid or corrupted state.
0xC1FF01FC WRM_ERR_WYUKON_NOT_INSTALLED	The accounting database is not installed on the specified server.
0xC1FF0203 WRM_ERR_JET_SERVER_TOO_BUSY	The server can service only one accounting request at a time.<44>

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.5 GetNextAccountingDataBatch (Opnum 11)

The GetNextAccountingDataBatch method gets the next batch of data in a previously initiated query to the accounting database.<45>

```
[id(5), helpstring("method GetNextAccountingDataBatch")] HRESULT GetNextAccountingDataBatch(
  [out] BSTR* pbstrResult,
  [out] BOOL* pbIsThereMoreData
);
```

pbstrResult: A pointer to a string that returns the requested data.

pbIsThereMoreData: A pointer to a Boolean value that specifies whether more data is available.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x81FF0217 WRM_WRN_WSRM_INCOMPLETE_FETCH	Data was fetched incompletely; the connection to the database might have terminated.
0xC1FF01FA WRM_ERR_WYUKON_NOT_CONNECTABLE	Cannot establish a connection to the accounting database.
0xC1FF0203 WRM_ERR_JET_SERVER_TOO_BUSY	The server can service only one accounting request at a time.<46>

The method GetNextAccountingDataBatch returns data from the accounting database if all the data was not retrieved by previous calls to this method or either of the methods ExecuteAccountingQuery (section 3.2.4.3.3) or GetRawAccountingData (section 3.2.4.3.4). The availability of additional database data is indicated by the value returned in the *pbIsThereMoreData* parameter of each of these methods.

If ExecuteAccountingQuery or GetRawAccountingData had returned indicating no more accounting data to be retrieved, and still GetNextAccountingDataBatch is called, *pbstrResult* is returned as NULL.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.6 DeleteAccountingData (Opnum 12)

The DeleteAccountingData method deletes accounting data within a specified time period from the accounting database (section 3.2.1.2). If there is no accounting data present between the specified dates, the functions returns SUCCESS while no accounting data is deleted.

```
[id(6), helpstring("method DeleteAccountingData")] HRESULT DeleteAccountingData(
  [in] BSTR bstrStartingDate,
  [in] BSTR bstrEndingDate,
  [in] BSTR bstrMachineName
);
```

bstrStartingDate: A string that specifies the starting date for the deletion, in date-and-time format (section 2.2.1.3). If this value is not in the correct format, the date range is ignored and the complete set of accounting data is deleted.

bstrEndingDate: A string that specifies the ending date for the deletion, in date-and-time format. If this value is not in the correct format, the date range is ignored and the complete set of accounting data is deleted.

bstrMachineName: A string that specifies the name of the machine whose accounting data is to be deleted. A default accounting database SHOULD<47> be defined.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.<48>
0xC1FF01FA WRM_ERR_WYUKON_NOT_CONNECTABLE	Cannot establish a connection to the accounting database.
0xC1FF01FB WRM_ERR_WYUKON_CORRUPTED	Cannot establish a connection to the accounting database; either the database is in single user mode and already connected or it is in an invalid or corrupted state.
0xC1FF0203 WRM_ERR_JET_SERVER_TOO_BUSY	The server can service only one accounting request at a time.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.7 DefragmentDB (Opnum 13)

The DefragmentDB method is not implemented. It MUST return a success code.

```
[id(7), helpstring("method DefragmentDB")] HRESULT DefragmentDB();
```

This method has no parameters.

Return Values: This method returns 0x00000000 for success.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.8 CancelAccountingQuery (Opnum 14)

The CancelAccountingQuery method cancels a previously-initiated query to the accounting database.

```
[id(8), helpstring("method CancelAccountingQuery")] HRESULT CancelAccountingQuery(
  [in] BOOL flag
);
```

flag: A Boolean value that specifies FALSE to stop reading data from the database and release the connection, and TRUE to read all remaining data and release the connection.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0xC1FF01FA WRM_ERR_WYUKON_NOT_CONNECTABLE	Cannot establish a connection to the accounting database.<49>

The method CancelAccountingQuery cancels a query to the accounting database after previous calls to either of the methods ExecuteAccountingQuery (section 3.2.4.3.3) or GetRawAccountingData (section 3.2.4.3.4) and before a call to the method GetNextAccountingDataBatch (section 3.2.4.3.5). The availability of additional database data is indicated by the value returned in the *pbIsThereMoreData* parameter of each method.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.9 RegisterAccountingClient (Opnum 15)

The RegisterAccountingClient method registers an accounting client for remote accounting on an accounting server. A default accounting database SHOULD<50> be defined.

Note This method is expected to be called remotely by the WSRM management service that is acting as an accounting client. A client SHOULD NOT call this method.

```
[id(9), helpstring("method RegisterAccountingClient")] HRESULT RegisterAccountingClient(
  [in] BSTR bstrClientId
```

```
);
```

bstrClientId: A string that specifies a client machine name.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success, or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0209 WRM_ERR_ACC_DISABLED_FOR_REMOTE_CLIENT	WSRM encountered an error in accounting.
0xC1FF020C WRM_ERR_REMOTE_SERVICE_NOT_SETUP_FOR_REMOTING	Connection to the remote server could not be established. The server is not set up for remote accounting. This error is returned when an accounting client is trying to register itself on an accounting server machine which itself is an accounting client.<51>
0xC1FF0212 WRM_ERR_INVALID_OPERATION	The operation is invalid. This error is returned when a WSRM management service acting as an accounting client tries to register itself twice, as when this method is called with the same DCOM interface instance.
0xC1FF0216 WRM_ERR_DBSERVER_CANNOT_BE_REMOTE	Data cannot be logged on the remote system. This error is returned when an accounting client calls this method on the accounting server but accounting is not yet initialized.<52>

It is possible for multiple WSRM servers to act as accounting clients and have a common accounting database server.

Note Remote accounting is not supported in a workgroup environment.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.10 DumpAccountingData (Opnum 16)

The DumpAccountingData method dumps accounting data from a remote server acting as an accounting client to the server currently acting as its accounting database server. The time interval of dumping data SHOULD be set by the client by using the **SetConfig** method call of the management service running on the accounting client.

Note This method is expected to be called remotely by the WSRM management service that is acting as an accounting client. A client SHOULD NOT call this method.

```
[id(10), helpstring("method DumpAccountingData")] HRESULT DumpAccountingData(
  [in] BSTR bstrAccountingData
);
```

bstrAccountingData: A string that specifies the accounting data to be dumped, in the form of an AccountingProcessList element (section 2.2.5.4).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0209 WRM_ERR_ACC_DISABLED_FOR_REMOTE_CLIENT	Accounting is disabled on the remote accounting server.
0xC1FF020A WRM_ERR_ACC_CLIENT_CONN_TERMINATED	Connection to remote server failed. The remote connection has been terminated by the accounting server because the accounting functionality of the accounting client was disabled.
0xC1FF020E WRM_ERR_ACC_NO_CONNECTION	The operation failed. There is no existing connection to the remote database server.

It is possible for multiple WSRM servers to have a common accounting database server.

Note Remote accounting is not supported in a workgroup environment.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.11 GetAccountingClients (Opnum 17)

The GetAccountingClients method gets the names of all servers that are acting as accounting clients on the current server.

```
[id(11), helpstring("method GetAccountingClients")] HRESULT GetAccountingClients(
  [out] BSTR* pbstrClientIds
);
```

pbstrClientIds: A pointer to a string that returns the machine names of all accounting clients, in the form of an AccountingClientList element (section 2.2.5.1).

Return Values: This method returns 0x00000000 for success, or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Return value/code	Description
0xC1FF0210 WRM_ERR_CLIENTSTATUS_ACC_OFF	Accounting functionality is currently turned off on the accounting server. There are no clients logging data to this machine.
0xC1FF0211 WRM_ERR_CLIENTSTATUS_ACC_REMOTE	Accounting is currently being done on a remote machine. There are no clients logging data to this machine.

The GetAccountingClients method can be used to find the names of WSRM servers, acting as accounting clients, for which the current server is acting as an accounting server.

Note Remote accounting is not supported in a workgroup environment.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.12 SetAccountingClientStatus (Opnum 18)

The SetAccountingClientStatus method sets the status of accounting functionality of servers that are in the remote accounting client role. Status of accounting functionality is controlled by the **Enabled** attribute in *bstrClientIds* XML. If the **Enabled** attribute in the **bstrClientIds** XML is specified as "true", the accounting functionality for the WSRM server, whose name is specified as the value of the respective AccountingClient node, is enabled; otherwise, if it is specified as "false", the accounting functionality is disabled. The DumpAccountingData method does not store accounting data in the database if the status of the accounting functionality of the respective server is disabled using the SetAccountingClientStatus method.<53>

```
[id(12), helpstring("method SetAccountingClientStatus")] HRESULT SetAccountingClientStatus(
  [in] BSTR bstrClientIds
);
```

bstrClientIds: A string that specifies the machine names of all accounting clients, in the form of an AccountingClientList element (section 2.2.5.1). For an example, see section 4.2.1. The value of the **Enabled** attribute specifies the accounting functionality status of the accounting clients.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success, or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF020F WRM_ERR_SETCLIENTSTATUS_XML_INVALID	The operation failed. XML data provided is invalid.
0xC1FF0210 WRM_ERR_CLIENTSTATUS_ACC_OFF	Accounting is currently turned off. There are no clients logging data to this machine.
0xC1FF0211 WRM_ERR_CLIENTSTATUS_ACC_REMOTE	Accounting is currently being done on a remote machine. There are no clients logging data to this machine.

Note Remote accounting is not supported in a workgroup environment.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.13 CheckAccountingConnection (Opnum 19)

The CheckAccountingConnection method determines the status of the connection to the accounting server.

Note This method is expected to be called remotely by the WSRM management service. In a remote accounting scenario, this method is called by the accounting client's WSRM service to check the connection status. A WSRM client SHOULD NOT call this method.

```
[id(13), helpstring("method CheckAccountingConnection")] HRESULT CheckAccountingConnection();
```

This method has no parameters.

Return Values: This method returns 0x00000000 for success, or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0xC1FF0209 WRM_ERR_ACC_DISABLED_FOR_REMOTE_CLIENT	Accounting is disabled on the remote accounting server.
0xC1FF020A WRM_ERR_ACC_CLIENT_CONN_TERMINATED	Connection to remote accounting server failed. The remote connection has been terminated by the accounting server because the accounting functionality of the accounting client was disabled.<54>
0xC1FF020E WRM_ERR_ACC_NO_CONNECTION	The operation failed. There is no existing connection to the remote database server.

It is possible for multiple WSRM servers to have a common accounting database server. <55>

Note Remote accounting is not supported in a workgroup environment.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.3.14 SetClientPermissions (Opnum 20)

The SetClientPermissions method adds or removes a specified client for remote accounting on a server and makes the required changes for DCOM permissions; that is, it adds the machine account of the accounting client to the Distributed COM Users group (see [MS-SAMR] section 3.1.4.2) of the accounting server.

```
[id(14), helpstring("method SetClientPermissions")] HRESULT SetClientPermissions(
  [in] BSTR bstrClientId,
  [in] BOOL fAddPermissions
);
```

bstrClientId: A string that identifies the client machine name whose account is to be added or removed.

fAddPermissions: TRUE to add and FALSE to remove.

Return Values: This method returns 0x00000000 for success, or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0xC1FF0213 WRM_ERR_INVALID_DBSERVER_NAME	Could not establish a connection to the database server. Either the server name is invalid or the machine executing this method does not have permission to perform an account lookup.
0xC1FF0214 WRM_ERR_PERMISSION_GRANT_UNSUCCESSFUL	Could not grant permissions to the current machine to log data remotely. This error is returned when a server machine denies access to a requesting machine.

It is possible for multiple WSRM servers to have a common accounting server.

Note Remote accounting is not supported in a workgroup environment.

Additional IWRMAccounting interface methods are specified in section 3.2.4.3.

3.2.4.4 IWRMCalendar Interface

The IWRMCalendar interface defines the following methods.

Methods in RPC Opnum Order

Method	Description
GetCalendarInfo	Gets information about one or all calendar events. Opnum: 7
CreateCalendar	Creates a new calendar event. Opnum: 8
ModifyCalendar	Modifies an existing calendar event. Opnum: 9
DeleteCalendar	Deletes a specified calendar event. Opnum: 10
RenameCalendar	Renames a specified calendar event. Opnum: 11
ComputeEvents	Computes the calendar events in a specified time interval. Opnum: 12
GetScheduleInfo	Gets information about a specified schedule. Opnum: 13
CreateSchedule	Creates a new schedule object. Opnum: 14
ModifySchedule	Modifies the specified schedule of the calendar. Opnum: 15

Method	Description
DeleteSchedule	Deletes an existing schedule. Opnum: 16
RenameSchedule	Renames a specified schedule object of the calendar. Opnum: 17
MoveBeforeCalendar	Moves the calendar event to before the specified reference calendar. Opnum: 18
MoveAfterCalendar	Moves the calendar event to after the specified reference calendar. Opnum: 19
GetServerTimeZone	Gets the current time zone setting of the server. Opnum: 20

3.2.4.4.1 GetCalendarInfo (Opnum 7)

The GetCalendarInfo method gets information about one or all calendar events.

```
[id(1), helpstring("method GetCalendarInfo")] HRESULT GetCalendarInfo(
  [in] BSTR bstrCalendarName,
  [out] BSTR* pbstrCalendarXML
);
```

bstrCalendarName: A string that specifies the name of the calendar event for which information will be returned. If the string is "\", all calendar events are specified.

pbstrCalendarXML: A pointer to a string that returns the specified calendar information, in the form of a Calendar element (section 2.2.5.7). For an example, see the Calendar example (section 4.2.6).

If **bstrCalendarName** is "\", all calendar events MUST be returned, in the form of a Calendars element (section 2.2.5.10). For an example, see the Calendars example (section 4.2.7).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.2 CreateCalendar (Opnum 8)

The CreateCalendar method creates a new calendar event.

```
[id(2), helpstring("method CreateCalendar")] HRESULT CreateCalendar(
  [in] BSTR bstrCalendarXML,
```

```
  [in] BOOL bChangeActivePolicy
);
```

bstrCalendarXML: A string that specifies the new calendar event, in the form of a Calendar element (section 2.2.5.7). For an example, see Calendar Example (section 4.2.8).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bChangeActivePolicy: A Boolean value that specifies whether the configuration changes made by this method call ~~should~~SHOULD change the current active policy, if applicable, of the system.<56>

Value	Meaning
FALSE 0x00000000	The current active policy of the system SHOULD NOT be changed by the method call.
TRUE 0x00000001	If applicable, the current active policy of the system SHOULD be changed by the method call.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<57>
0xC1FF0258 WRM_ERR_CAL_DUPLICATE_CALENDAR	A calendar event with the specified name already exists.
0xC1FF025B WRM_ERR_CAL_MAX_CAL_EXCEEDED	The number of calendar events that exist at one time has exceeded an implementation-defined limit.<58>

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.3 ModifyCalendar (Opnum 9)

The ModifyCalendar method modifies an existing calendar event.

```
[id(3), helpstring("method ModifyCalendar")] HRESULT ModifyCalendar(
  [in] BSTR bstrCalendarXML,
  [in] BOOL bOverwrite,
  [in] BOOL bChangeActivePolicy
);
```

bstrCalendarXML: A string that specifies the calendar event, in the form of a Calendar element (section 2.2.5.7).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bOverwrite: A Boolean value that specifies whether to ignore the timestamp of the specified calendar event when validating.

A timestamp MUST be defined inside a common node at the root level of an XML element, as shown in the Calendar example (section 4.2.6). The format of a timestamp is specified in section 2.2.1.4.

Value	Meaning
FALSE 0x00000000	The timestamp of the new calendar event MUST specify a time that is later than or equal to the timestamp of any modifications made to a calendar or schedule object on the server. Otherwise, the modification SHOULD fail, and **WRM_ERR_OLD_INFORMATION** SHOULD be returned.
TRUE 0x00000001	The calendar event is validated and modified without checking the timestamp.

bChangeActivePolicy: A Boolean value that specifies whether the configuration changes made by this method call ~~should~~SHOULD change the current active policy, if applicable, of the system.<59>

Value	Meaning
FALSE 0x00000000	The current active policy of the system SHOULD NOT be changed by the method call.
TRUE 0x00000001	If applicable, the current active policy of the system SHOULD be changed by the method call.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0069 WRM_ERR_OLD_INFORMATION	The XML timestamp is out of date.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<60>
0xC1FF0260 WRM_ERR_CAL_INTERVAL_TOO_LONG	The calendar recurrence interval value MUST be from 1 to 999.
0xC1FF0263 WRM_ERR_CAL_INVALID_MONTH	The month value MUST be from 1 to 12.
0xC1FF0264 WRM_ERR_CAL_INVALID_MONTHDAY	The days of the month MUST have a value from 1 to 31.
0xC1FF0265 WRM_ERR_CAL_INVALID_YEARDAY	The day of the year cannot be zero and SHOULD NOT exceed the number of days in the selected year.<61>

Return value/code	Description
0xC1FF0266 WRM_ERR_CAL_INVALID_WEEKNO	The weeks of the year MUST have a value from 1 to 53.
0xC1FF0267 WRM_ERR_CAL_INVALID_SETPOS	The value of the monthly instance of a day of the week is invalid.
0xC1FF0269 WRM_ERR_CAL_SCHEDULE_NAME_INVALID	The specified schedule name does not exist.
0xC1FF0275 WRM_ERR_CAL_INVALID_DURATION	The specified calendar event duration is invalid. Specify a duration using the format dd:hh:mm.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.4 DeleteCalendar (Opnum 10)

The DeleteCalendar method deletes a specified calendar event.

```
[id(4), helpstring("method DeleteCalendar")] HRESULT DeleteCalendar(
  [in] BSTR bstrCalendarName,
  [in] BOOL bChangeActivePolicy
);
```

bstrCalendarName: A string that specifies the name of the calendar event.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bChangeActivePolicy: A Boolean value that specifies whether the configuration changes made by this method call ~~should~~SHOULD change the current active policy, if applicable, of the system.<62>

Value	Meaning
FALSE 0x00000000	The current active policy of the system SHOULD NOT be changed by the method call.
TRUE 0x00000001	If applicable, the current active policy of the system SHOULD be changed by the method call.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-") and cannot contain spaces or any of the following characters:

Return value/code	Description
	\ / ? * | : < > " , ;.
0xC1FF0259 WRM_ERR_CAL_UNKNOWN_CALENDAR	The specified calendar event does not exist.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.5 RenameCalendar (Opnum 11)

The RenameCalendar method renames a specified calendar event.

```
[id(5), helpstring("method RenameCalendar")] HRESULT RenameCalendar(
  [in] BSTR bstrOldCalendarName,
  [in] BSTR bstrNewCalendarName
);
```

bstrOldCalendarName: A string that specifies the current name of the calendar event to be renamed.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrNewCalendarName: A string that specifies the new name of the calendar event.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-") and cannot contain spaces or any of the following characters: \ / ? * | : < > " , ;.
0xC1FF0258 WRM_ERR_CAL_DUPLICATE_CALENDAR	A calendar event with the specified name already exists.
0xC1FF0259 WRM_ERR_CAL_UNKNOWN_CALENDAR	The specified calendar event does not exist.
0xC1FF025A WRM_ERR_CAL_NAME_TOO_LONG	The calendar name has exceeded an implementation-defined limit.<63>.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.6 ComputeEvents (Opnum 12)

The ComputeEvents method computes the calendar events in a specified time interval. It returns the list of all calendar events from the system in *pbstrEvents* with a start time greater than or equal to the one specified in *szStartTime* and with an end time less than or equal to the one specified in *szEndTime*. If the method is called with *fMergeEvents* set to TRUE, conflicting events are merged and returned in *pbstrConflicts*.

```
[id(6), helpstring("method ComputeEvents")] HRESULT ComputeEvents(
  [in] BSTR szStartTime,
  [in] BSTR szEndTime,
  [in] BOOL fMergeEvents,
  [out] BSTR* pbstrEvents,
  [out] BSTR* pbstrConflicts
);
```

szStartTime: A string that specifies the start time to compute calendar events, in time format (section 2.2.1.3).

szEndTime: A string that specifies the end time to compute calendar events, in time format.

fMergeEvents: A Boolean value that specifies whether to merge calendar events. All conflicting events, that is, those whose start time and end time overlap, are returned in *pbstrConflicts*. If the *fMergeEvents* value is TRUE, conflicting events are merged into a new event with the start-time **DtTmStart** element picked from the event with the earliest start time and the end-time **DtTmEnd**, **PolicyName**, and **CalendarName** elements picked from the event with the latest end time.

Value	Meaning
FALSE 0x00000000	The server SHOULD NOT merge calendar events.
TRUE 0x00000001	The server SHOULD merge calendar events.

pbstrEvents: A pointer to a string that returns a list of events that are computed during execution, in the form of an Events element (section 2.2.5.15). For an example, see Events Example (section 4.2.12).

pbstrConflicts: A pointer to a string that returns a list of conflicting events during the specified time interval, in the form of an Events element (section 2.2.5.15). For an example, see Events Example (section 4.2.12).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.7 GetScheduleInfo (Opnum 13)

The GetScheduleInfo method gets information about a specified schedule object.

```
[id(7), helpstring("method GetScheduleInfo")] HRESULT GetScheduleInfo(
  [in] BSTR bstrScheduleName,
  [out] BSTR* pbstrScheduleXML
);
```

bstrScheduleName: A string that specifies the name of the schedule object for which information is required.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

pbstrScheduleXML: A pointer to a string that returns the specified schedule object information structure, in the form of a Schedule element (section 2.2.5.26). Sample XML is provided in Schedule XML Example (section 4.2.22).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0271 WRM_ERR_CAL_UNKNOWN_SCHEDULE	The specified schedule object does not exist.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.8 CreateSchedule (Opnum 14)

The CreateSchedule method creates a new schedule object.

```
[id(8), helpstring("method CreateSchedule")] HRESULT CreateSchedule(
  [in] BSTR bstrScheduleXML
);
```

bstrScheduleXML: A string that specifies the new schedule, in the form of a Schedule element (section 2.2.5.26). Sample XML is provided in Schedule Example (section 4.2.22).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<64>

Return value/code	Description
0xC1FF025C WRM_ERR_CAL_MAX_SCHED_EXCEEDED	The number of schedules has exceeded an implementation-defined limit.<65>
0xC1FF0270 WRM_ERR_CAL_DUPLICATE_SCHEDULE	The specified schedule object already exists.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.9 ModifySchedule (Opnum 15)

The ModifySchedule method modifies the specified schedule of the calendar.

```
[id(9), helpstring("method ModifySchedule")] HRESULT ModifySchedule(
  [in] BSTR bstrScheduleXML,
  [in] BOOL bOverwrite,
  [in] BOOL bChangeActivePolicy
);
```

bstrScheduleXML: A string that specifies the modified schedule, in the form of a Schedule element (section 2.2.5.26). Sample XML is provided in Schedule Example (section 4.2.22).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bOverwrite: A Boolean value that specifies whether to ignore the timestamp of the specified schedule object when validating.

A timestamp MUST be defined inside a common node at the root level of an XML element, as shown in the Calendar example (section 4.2.6). The format of a timestamp is specified in section 2.2.1.4.

Value	Meaning
FALSE 0x00000000	The timestamp of the new schedule object MUST specify a time that is later than or equal to the timestamp of any modifications made to a calendar or schedule object on the server. Otherwise, the modification SHOULD fail, and **WRM_ERR_OLD_INFORMATION** SHOULD be returned.
TRUE 0x00000001	The schedule object is validated and modified without checking the timestamp.

bChangeActivePolicy: A Boolean value that specifies whether the configuration changes made by this method call ~~should~~SHOULD change the current active policy, if applicable, of the system.<66>

Value	Meaning
FALSE 0x00000000	The current active policy of the system SHOULD NOT be changed by the method call.
TRUE 0x00000001	If applicable, the current active policy of the system SHOULD be changed by the method call.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0069 WRM_ERR_OLD_INFORMATION	The XML timestamp is out of date.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<67>
0xC1FF0271 WRM_ERR_CAL_UNKNOWN_SCHEDULE	The specified schedule object does not exist.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.10 DeleteSchedule (Opnum 16)

The DeleteSchedule method deletes an existing schedule object.

```
[id(10), helpstring("method DeleteSchedule")] HRESULT DeleteSchedule(
  [in] BSTR bstrScheduleName
);
```

bstrScheduleName: A string that specifies the name of the schedule object to be deleted.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0xC1FF0272 WRM_ERR_CAL_SCHEDULE_IN_USE	The selected schedule is currently in use with a calendar. Delete or edit the calendar first.
0xC1FF0271 WRM_ERR_CAL_UNKNOWN_SCHEDULE	The specified schedule object does not exist.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.11 RenameSchedule (Opnum 17)

The RenameSchedule method renames a specified schedule object. If the schedule object is being referenced by some calendar object, then the calendar object is also updated with the new name.

```
[id(11), helpstring("method RenameSchedule")] HRESULT RenameSchedule(
  [in] BSTR bstrOldScheduleName,
```

```
  [in] BSTR bstrNewScheduleName
);
```

bstrOldScheduleName: A string that specifies the current name of the schedule object to be renamed.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrNewScheduleName: A string that specifies the new name of the schedule object.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-") and cannot contain spaces or any of the following characters: \ / ? * | : < > " , ;.
0xC1FF0270 WRM_ERR_CAL_DUPLICATE_SCHEDULE	The new schedule name is already taken by an existing schedule object.
0xC1FF0271 WRM_ERR_CAL_UNKNOWN_SCHEDULE	The specified schedule object does not exist.
0xC1FF0273 WRM_ERR_CAL_SCHEDULE_NAME_TOOLONG	The schedule object name has exceeded an implementation-defined limit.<68>

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.12 MoveBeforeCalendar (Opnum 18)

The MoveBeforeCalendar method moves a calendar event before the specified reference event. The caller can choose to control the move depending on whether the current resource policy is affected.

```
[id(12), helpstring("method MoveBeforeCalendar")] HRESULT MoveBeforeCalendar(
  [in] BSTR bstrCalendarName,
  [in] BSTR bstrRefCalendarName,
  [in] BOOL bChangeActivePolicy
);
```

bstrCalendarName: A string that specifies the name of the calendar event to be moved.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrRefCalendarName: A string that specifies the name of a reference calendar event, before which the specified event is to be moved.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bChangeActivePolicy: A Boolean value that specifies whether the configuration changes made by this method call ~~should~~SHOULD change the current active policy, if applicable, of the system.<69>

Value	Meaning
FALSE 0x00000000	The current active policy of the system SHOULD NOT be changed by the method call.
TRUE 0x00000001	If applicable, the current active policy of the system SHOULD be changed by the method call.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0259 WRM_ERR_CAL_UNKNOWN_CALENDAR	The specified calendar event does not exist.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.13 MoveAfterCalendar (Opnum 19)

The MoveAfterCalendar method moves a calendar event after the specified reference event. The caller can choose to control the move depending on whether the current resource policy is affected.

```
[id(13), helpstring("method MoveAfterCalendar")] HRESULT MoveAfterCalendar(
  [in] BSTR bstrCalendarName,
  [in] BSTR bstrRefCalendarName,
  [in] BOOL bChangeActivePolicy
);
```

bstrCalendarName: A string that specifies the name of the calendar event to be moved.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrRefCalendarName: A string that specifies the name of a reference calendar event, after which the specified event is to be moved.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bChangeActivePolicy: A Boolean value that specifies whether the configuration changes made by this method call ~~should~~SHOULD change the current active policy, if applicable, of the system.<70>

Value	Meaning
FALSE 0x00000000	The current active policy of the system SHOULD NOT be changed by the method call.

Value	Meaning
TRUE 0x00000001	If applicable, the current active policy of the system SHOULD be changed by the method call.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0259 WRM_ERR_CAL_UNKNOWN_CALENDAR	The specified calendar event does not exist.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.4.14 GetServerTimeZone (Opnum 20)

The GetServerTimeZone method gets the current time zone setting of the server.

```
[id(14), helpstring("method GetServerTimeZone")] HRESULT GetServerTimeZone(
  [out] int* pnServerTimeZone
);
```

pnServerTimeZone: A pointer to a 32-bit signed integer that returns the server time zone.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Additional IWRMCalendar interface methods are specified in section 3.2.4.4.

3.2.4.5 IWRMConfig Interface

The IWRMConfig interface defines the following methods.

Methods in RPC Opnum Order

Method	Description
GetConfig	Gets WSRM configuration information concerning accounting, calendars, and notifications. Opnum: 7

Method	Description
SetConfig	Sets WSRM configuration information concerning accounting, calendars, and notifications. Opnum: 8
IsEnabled	Returns whether the management of a type of WSRM configuration is enabled or disabled. Opnum: 9
EnableDisable	Enables or disables the management of a specified type of WSRM configuration. Opnum: 10
GetExclusionList	Gets the list of processes not managed by the WSRM management service. Opnum: 11
SetExclusionList	Updates the contents of the exclusion list. Opnum: 12
WSRMActivate	Activates or deactivates WSRM. Opnum: 13
IsWSRMActivated	Returns whether WSRM is active. Opnum: 14
RestoreExclusionList	Restores the exclusion list to default values. Opnum: 15

A number of protocol message processing steps result in the server writing accounting records to a database. Minimally, using the SetConfig method:

- Set the accounting database to a known database<71> or
- Set the accounting interval and turn on accounting.

3.2.4.5.1 GetConfig (Opnum 7)

The GetConfig method gets WSRM configuration information concerning accounting and notifications.

```
[id(1), helpstring("method GetConfig")] HRESULT GetConfig(
  [out] BSTR* pbstrConfigInfo,
  [in] CONFIGTYPE enumConfigType
);
```

pbstrConfigInfo: A pointer to a string that returns WSRM configuration information. The type of information is determined by the value of the *enumConfigType* parameter.

enumConfigType: A CONFIGTYPE enumeration (section 2.2.3.1) value that specifies the type of WSRM configuration information to get.

Obtaining calendar information is not supported. If this parameter is set to **CONFIGTYPE_CALENDARING**, **ERROR_NOT_SUPPORTED** SHOULD be returned.

Value	Meaning
CONFIGTYPE_ACCOUNTING 1	The WSRM configuration information is in the form of an AccountingConfigInfo element (section 2.2.5.2).
CONFIGTYPE_NOTIFICATION	The WSRM configuration information is in the form of a NotificationConfigInfo element (section 2.2.5.19).

Value	Meaning
2	

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070032 ERROR_NOT_SUPPORTED	The requested action is not supported.

Additional IWRMConfig interface methods are specified in section 3.2.4.5.

3.2.4.5.2 SetConfig (Opnum 8)

The SetConfig method sets WSRM configuration information concerning accounting and notifications.

```
[id(2), helpstring("method SetConfig")] HRESULT SetConfig(
  [in] BSTR bstrConfigInfo,
  [in] CONFIGTYPE enumConfigType
);
```

bstrConfigInfo: A string that contains WSRM configuration information. The type of information is determined by the value of the enumConfigType parameter.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

enumConfigType: A CONFIGTYPE enumeration (section 2.2.3.1) value that specifies the type of WSRM configuration information to set.

Setting calendar information is not supported. If this parameter is set to **CONFIGTYPE_CALENDARING**, ERROR_NOT_SUPPORTED SHOULD be returned.

Value	Meaning
CONFIGTYPE_ACCOUNTING 1	The WSRM configuration information is in the form of an AccountingConfigInfo element (section 2.2.5.2).
CONFIGTYPE_NOTIFICATION 2	The WSRM configuration information is in the form of an NotificationConfigInfo element (section 2.2.5.19).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070005 ERROR_ACCESS_DENIED	Access is denied.
0x80070032	The requested action is not supported.

Return value/code	Description
ERROR_NOT_SUPPORTED	
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006E WRM_ERR_TOO_LONG_CONFIG_VALUE	One or more specified values have exceeded an implementation-defined limit.<72>
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<73>
0xC1FF0190 WRM_ERR_INVALID_NTFY_ENABLE	The notification-enabled value MUST be Boolean (section 2.2.1.2).
0xC1FF0194 WRM_ERR_INVALID_EVENTLIST	The notification event list format is invalid.
0xC1FF01F6 WRM_ERR_INVALID_ACC_ENABLE	The accounting-enabled value MUST be Boolean (section 2.2.1.2).
0xC1FF01F9 WRM_ERR_ACC_INVALID_DUMPING_INTERVAL	The logging interval for accounting is invalid. The interval MUST be between 2 and 60,000 minutes, inclusive.

Note When the **CONFIGTYPE_ACCOUNTING** option is used, an accounting client SHOULD call SetClientPermissions (section 3.2.4.3.14) prior to SetConfig in order to obtain authorization to modify an accounting database on a remote WSRM server. If a client does not have permission, ERROR_ACCESS_DENIED SHOULD be returned.<74>

Additional IWRMConfig interface methods are specified in section 3.2.4.5.

3.2.4.5.3 IsEnabled (Opnum 9)

The IsEnabled method returns whether a type of WSRM configuration object is enabled or disabled.

```
[id(3), helpstring("method IsEnabled")] HRESULT IsEnabled(
  [out] BOOL* pbEnable,
  [in] CONFIGTYPE enumConfigType
);
```

pbEnable: A pointer to a Boolean value that indicates whether the WSRM configuration object is enabled or disabled.

enumConfigType: A CONFIGTYPE enumeration (section 2.2.3.1) value that specifies the type of WSRM configuration object.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Additional IWRMConfig interface methods are specified in section 3.2.4.5.

3.2.4.5.4 EnableDisable (Opnum 10)

The EnableDisable method enables or disables the management of a specified type of WSRM configuration.

```
[id(4), helpstring("method EnableDisable")] HRESULT EnableDisable(
  [in] BOOL bEnableDisable,
  [in] CONFIGTYPE enumConfigType
);
```

bEnableDisable: A Boolean value that specifies whether to enable or disable the management of the WSRM configuration. The type of configuration is determined by the value of the *enumConfigType* parameter.

Value	Meaning
FALSE 0x00000000	Management of the WSRM configuration MUST be disabled.
TRUE 0x00000001	Management of the WSRM configuration MUST be enabled.

enumConfigType: A CONFIGTYPE enumeration (section 2.2.3.1) value that specifies the type of WSRM configuration to enable or disable.

If this parameter value is outside the range of the CONFIGTYPE enumeration, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070032 ERROR_NOT_SUPPORTED	The requested action is not supported.<75>
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0x81FF006D WRM_WARNING_CALENDARING_ALREADY_ENABLED	The management service is already in calendar mode.
0x81FF0196 WRM_WARNING_NOTIFICATION_ALREADY_DISABLED	Notification is already disabled.
0x81FF0197 WRM_WARNING_NOTIFICATION_ALREADY_ENABLED	Notification is already enabled.
0x81FF01F4 WRM_WARNING_ACCOUNTING_ALREADY_DISABLED	Accounting is already disabled.
0x81FF01F5	Accounting is already enabled.

Return value/code	Description
WRM_WARNING_ACCOUNTING_ALREADY_ENABLED	
0xC1FF007E WRM_ERR_ACCENABLED_UNDER_GROUPPOLICY	The accounting setting is currently configured under **Group Policy** and cannot be modified.<76>
0xC1FF0191 WRM_ERR_SMTPSERVERNAME_CANNOT_BE_NULL	Both the Simple Mail Transfer Protocol (SMTP) server and email name are required in order to enable notification.
0xC1FF0193 WRM_ERR_EVENTLIST_CANNOT_BE_NULL	Notification requires at least one event.
0xC1FF01F8 WRM_ERR_ACCOUNTING_NOT_CONFIGURED_PROPERLY	Accounting has not been configured properly. The accounting server might not be present or access might be denied.<77>
0xC1FF020B WRM_ERR_REMOTE_SERVICE_NOT_STARTED	Connection to the remote server could not be established because the WSRM management service has not been started on the remote server.<78>

Additional IWRMConfig interface methods are specified in section 3.2.4.5.

3.2.4.5.5 GetExclusionList (Opnum 11)

The GetExclusionList method gets the current exclusion list of processes not managed by the WSRM management service.

```
[id(5), helpstring("method GetExclusionList")] HRESULT GetExclusionList(
  [out] BSTR* pbstrExclusionList,
  [in] EXCLUSIONLIST_TYPE enumListType
);
```

pbstrExclusionList: A pointer to a string that returns the exclusion list, in the form of an ExclusionList element (section 2.2.5.16). For an example, see ExclusionList Example (section 4.2.13).

enumListType: An EXCLUSIONLIST_TYPE enumeration (section 2.2.3.2) value that specifies whether the list is system-defined or user-defined.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006B WRM_ERR_TOO_LONG_PROCESS_NAME	The exclusion list could not be returned because the number of characters in a process name has exceeded an implementation-defined limit.<79>
0xC1FF006C WRM_ERR_EXCLUSION_LIST_LIMIT_EXCEEDED	The exclusion list could not be returned because the number of processes that can be excluded has exceeded

Return value/code	Description
	an implementation-defined limit.<80>

Additional IWRMConfig interface methods are specified in section 3.2.4.5.

3.2.4.5.6 SetExclusionList (Opnum 12)

The SetExclusionList method updates the contents of the exclusion list which is of type USER_EXCLUSION_LIST.

```
[id(6), helpstring("method SetExclusionList")] HRESULT SetExclusionList(
  [in] BSTR bstrExclusionList
);
```

bstrExclusionList: A string that specifies a list of processes, in the form of an ExclusionList element (section 2.2.5.16). For an example, see (section 4.2.13).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0xC1FF006B WRM_ERR_TOO_LONG_PROCESS_NAME	The exclusion list could not be returned because the number of characters in a process name has exceeded an implementation-defined limit.<81>.
0xC1FF006C WRM_ERR_EXCLUSION_LIST_LIMIT_EXCEEDED	The exclusion list could not be returned because the number of processes that can be excluded has exceeded an implementation-defined limit.<82>

Additional IWRMConfig interface methods are specified in section 3.2.4.5.

3.2.4.5.7 WSRMActivate (Opnum 13)

The WSRMActivate method activates or deactivates WSRM management state.

```
[id(7), helpstring("method WSRMActivate")] HRESULT WSRMActivate(
  [in] BOOL bActivate
);
```

bActivate: A Boolean value that specifies whether to activate or deactivate WSRM.

Value	Meaning
FALSE 0x00000000	Management state of WSRM management service MUST be set to inactive or stopped.
TRUE 0x00000001	Management state of WSRM management service MUST be set to active or running.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x81FF0075 WRM_WRN_WSRM_ALREADY_ACTIVATED	WSRM has already been activated.
0x81FF0074 WRM_WRN_WSRM_ALREADY_DEACTIVATED	WSRM has already been deactivated.

This method MUST update the **registry** according to the management state of WSRM. If the WSRM management service later restarts, for example, after a reboot of the operating system, the WSRM management service MUST attempt to restore its prior management state.<83>

Additional IWRMConfig interface methods are specified in section 3.2.4.5.

3.2.4.5.8 IsWSRMActivated (Opnum 14)

The IsWSRMActivated method returns whether WSRM is active.

```
[id(8), helpstring("method IsWSRMActivated")] HRESULT IsWSRMActivated(
  [out] BOOL* pbActivated
);
```

pbActivated: A pointer to a Boolean value that returns whether the WSRM server is in the active management state. If TRUE, WSRM is active; otherwise, it is inactive.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Additional IWRMConfig interface methods are specified in section 3.2.4.5.

3.2.4.5.9 RestoreExclusionList (Opnum 15)

The RestoreExclusionList method restores the exclusion list, which is of type USER_EXCLUSION_LIST, to default values.

```
[id(9), helpstring("method RestoreExclusionList")] HRESULT RestoreExclusionList();
```

This method has no parameters.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IWRMConfig interface methods are specified in section 3.2.4.5.

3.2.4.6 IWRMMachineGroup Interface

The IWRMMachineGroup interface defines the following methods.<84>

Methods in RPC Opnum Order

Method	Description
CreateMachineGroup	Create and initialize a machine group. Opnum: 7
GetMachineGroupInfo	Return information about a machine group. Opnum: 8
ModifyMachineGroup	Modify an existing machine group. Opnum: 9
DeleteMachineGroup	Delete an existing machine group. Opnum: 10
RenameMachineGroup	Change the name of an existing machine group. Opnum: 11
AddMachine	Add a machine to a machine group. Opnum: 12
GetMachineInfo	Return information about a machine in a machine group. Opnum: 13
ModifyMachineInfo	Modify a machine in a machine group. Opnum: 14
DeleteMachine	Delete a machine from a machine group. Opnum: 15

3.2.4.6.1 CreateMachineGroup (Opnum 7)

The CreateMachineGroup method creates and initializes a machine group.

```
[id(1), helpstring("method CreateMachineGroup")] HRESULT CreateMachineGroup(
  [in] BSTR bstrParentMachineGroupId,
  [in] BSTR bstrMachineGroupInfo
);
```

bstrParentMachineGroupId: A string that specifies the identifier of the parent machine group in which to create a new machine group.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrMachineGroupInfo: A string that specifies information about the machine group to be created, including its identifier, in the format of a MachineGroup element (section 2.2.5.18).<85>

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<86>
0xC1FF0385 WRM_ERR_MACHINE_GROUP_LIMIT_EXCEEDED	The total number of machine groups has exceeded an implementation-defined limit.<87>
0xC1FF0387 WRM_ERR_MACHINES_LIMIT_IN_MACHINEGROUP_EXCEEDED	The total number of machines directly under a machine group has exceeded an implementation-defined limit.<88>
0xC1FF0388 WRM_ERR_MACHINEGROUP_ALREADY_EXISTS	A machine group with the specified name in *bstrMachineGroupInfo* XML already exists in the entire WSRM configuration.
0xC1FF0389 WRM_ERR_MACHINEGROUPID_INVALID	The specified parent machine group id does not exist.

Additional IWRMMachineGroup interface methods are specified in section 3.2.4.6.

3.2.4.6.2 GetMachineGroupInfo (Opnum 8)

The GetMachineGroupInfo method returns information about a machine group.

```
[id(2), helpstring("method GetMachineGroupInfo")] HRESULT GetMachineGroupInfo(
  [in] BSTR bstrMachineGroupId,
  [out] BSTR* pbstrMachineGroupInfo
);
```

bstrMachineGroupId: A string that specifies the identifier of the machine group for which to return information.

pbstrMachineGroupInfo: A pointer to a string that SHOULD return information about the machine group, in the format specified in MachineGroup (section 2.2.5.17).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Return value/code	Description
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0389 WRM_ERR_MACHINEGROUPID_INVALID	The specified machine group id is invalid.

Additional IWRMMachineGroup interface methods are specified in section 3.2.4.6.

3.2.4.6.3 ModifyMachineGroup (Opnum 9)

The ModifyMachineGroup method modifies an existing machine group. The method replaces or merges the machine group information according to the value specified in the **enumMGMergeOptions** member.

```
[id(3), helpstring("method ModifyMachineGroup")] HRESULT ModifyMachineGroup(
  [in] BSTR bstrMachineGroupId,
  [in] BSTR bstrMachineGroupInfo,
  [in] MACHINE_GROUP_MERGE_OPTIONS enumMGMergeOptions
);
```

bstrMachineGroupId: A string that specifies the identifier of the machine group to be modified.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrMachineGroupInfo: A string that specifies the new information for the machine group, in the format specified in MachineGroup element (section 2.2.5.18).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

enumMGMergeOptions: Options for machine group modification, from the MACHINE_GROUP_MERGE_OPTIONS enumeration (section 2.2.3.4).<89>

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.<90>
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<91>
0xC1FF0385 WRM_ERR_MACHINE_GROUP_LIMIT_EXCEEDED	The total number of machine groups as specified in **bstrMachineGroupInfo**, has exceeded an implementation-defined limit.<92>
0xC1FF0387 WRM_ERR_MACHINES_LIMIT_IN_MACHINEGROUP_EXCEEDED	The machine group information could not be modified because the total number of machines directly under a machine group has exceeded an implementation-defined limit.<93>

Return value/code	Description
0xC1FF0388 WRM_ERR_MACHINEGROUP_ALREADY_EXISTS	A machine group with the specified name in *bstrMachineGroupInfo* XML already exists in the entire WSRM configuration. For example, if **ModifyMachineGroup** is used to modify a machine group ID that is identical to the existing group ID, this error will be generated.<94>
0xC1FF0389 WRM_ERR_MACHINEGROUPID_INVALID	The specified machine group ID is invalid.

Additional IWRMMachineGroup interface methods are specified in section 3.2.4.6.

3.2.4.6.4 DeleteMachineGroup (Opnum 10)

The DeleteMachineGroup method deletes an existing machine group.

```
[id(4), helpstring("method DeleteMachineGroup")] HRESULT DeleteMachineGroup(
  [in] BSTR bstrMachineGroupId
);
```

bstrMachineGroupId: A string that specifies the identifier of the machine group to be deleted.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0389 WRM_ERR_MACHINEGROUPID_INVALID	The specified machine group id is invalid.

Additional IWRMMachineGroup interface methods are specified in section 3.2.4.6.

3.2.4.6.5 RenameMachineGroup (Opnum 11)

The RenameMachineGroup method changes the name of an existing machine group.

```
[id(5), helpstring("method RenameMachineGroup")] HRESULT RenameMachineGroup(
  [in] BSTR ~~bstrOldMachineGroupId~~bstrOldMachineGroupName,
  [in] BSTR ~~bstrNewMachineGroupId~~bstrNewMachineGroupName
);
```

~~bstrOldMachineGroupId~~bstrOldMachineGroupName: A string that specifies the identifier of the machine group to be renamed.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

~~bstrNewMachineGroupId~~bstrNewMachineGroupName: A string that specifies the new identifier of the machine group.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0388 WRM_ERR_MACHINEGROUP_ALREADY_EXISTS	A machine group with the specified name already exists in the entire WSRM configuration.
0xC1FF0389 WRM_ERR_MACHINEGROUPID_INVALID	The specified machine group id is invalid.

Additional IWRMMachineGroup interface methods are specified in section 3.2.4.6.

3.2.4.6.6 AddMachine (Opnum 12)

The AddMachine method adds a machine to a machine group.

```
[id(6), helpstring("method AddMachine")] HRESULT AddMachine(
  [in] BSTR bstrParentMachineGroupId,
  [in] BSTR bstrMachineInfo
);
```

bstrParentMachineGroupId: A string that specifies the identifier of the machine group in which to add a machine.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrMachineInfo: A string that specifies the machine, in the format specified in Machine element (section 2.2.5.17).<95>

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<96>
0xC1FF0389 WRM_ERR_MACHINEGROUPID_INVALID	The specified machine group id is invalid.

Return value/code	Description
0xC1FF038B WRM_ERR_MACHINE_LIMIT_EXCEEDED	The total number of machines has exceeded an implementation-defined limit.<97>
0xC1FF038C WRM_ERR_MACHINE_ALREADY_EXISTS	A machine with the specified name already exists as the direct child in the machine group.

Additional IWRMMachineGroup interface methods are specified in section 3.2.4.6.

3.2.4.6.7 GetMachineInfo (Opnum 13)

The GetMachineInfo method returns information about a machine. If more than one machine with the specified **bstrMachineId** value exists in the hierarchy of machine groups, the information can be returned from any of the machine groups.

```
[id(7), helpstring("method GetMachineInfo")] HRESULT GetMachineInfo(
  [in] BSTR bstrMachineId,
  [out] BSTR* pbstrMachineInfo
);
```

bstrMachineId: A string that specifies the identifier of the machine for which to return information.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

pbstrMachineInfo: A pointer to a string that returns information about the machine, in the format specified in Machine element (section 2.2.5.17).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF038A WRM_ERR_MACHINEID_INVALID	The specified machine ID is not found in any machine group of the configuration.<98>

Additional IWRMMachineGroup interface methods are specified in section 3.2.4.6.

3.2.4.6.8 ModifyMachineInfo (Opnum 14)

The ModifyMachineInfo method modifies a machine in a machine group. This method modifies the direct child machine with the specified machine ID under the specified parent machine group.

```
[id(8), helpstring("method ModifyMachineInfo")] HRESULT ModifyMachineInfo(
  [in] BSTR bstrParentMachineGroupId,
  [in] BSTR bstrMachineId,
  [in] BSTR bstrMachineInfo
);
```

bstrParentMachineGroupId: A string that specifies the identifier of the machine group that contains the machine to modify.<99>

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrMachineId: A string that specifies the identifier of the machine to modify.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrMachineInfo: A string that specifies the new information for the machine, in the format specified in Machine element (section 2.2.5.17).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<100>
0xC1FF0389 WRM_ERR_MACHINEGROUPID_INVALID	The specified machine group ID is invalid.
0xC1FF038A WRM_ERR_MACHINEID_INVALID	The specified machine ID is not found.<101>
0xC1FF038C WRM_ERR_MACHINE_ALREADY_EXISTS	A machine with the specified machine name in the *bstrMachineInfo* XML already exists as the direct child in the machine group.<102>

Additional IWRMMachineGroup interface methods are specified in section 3.2.4.6.

3.2.4.6.9 DeleteMachine (Opnum 15)

The DeleteMachine method deletes a machine from a machine group.

```
[id(9), helpstring("method DeleteMachine")] HRESULT DeleteMachine(
  [in] BSTR bstrParentMachineGroupId,
  [in] BSTR bstrMachineId,
  BOOL bRecursive
);
```

bstrParentMachineGroupId: A string that specifies the identifier of the machine group that contains the machine to delete.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrMachineId: A string that specifies the identifier of the machine to delete.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bRecursive: A Boolean value that specifies whether to recursively delete all instances of the specified machine.<103>

Value	Meaning
TRUE 0x00000001	All instances of the specified machine MUST be recursively searched in the specified machine group and deleted.
FALSE 0x00000000	Only a machine that is a direct child in the specified machine group SHOULD be deleted.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0389 WRM_ERR_MACHINEGROUPID_INVALID	The specified machine group ID is invalid.
0xC1FF038A WRM_ERR_MACHINEID_INVALID	The specified machine ID is not found.<104>

Additional IWRMMachineGroup interface methods are specified in section 3.2.4.6.

3.2.4.7 IWRMPolicy Interface

The IWRMPolicy interface defines the following methods.

Methods in RPC Opnum Order

Method	Description
GetPolicyInfo	Gets Policy or policies with specified policy ID. Opnum: 7
CreatePolicy	Creates a new policy. Opnum: 8
ModifyPolicy	Modifies an existing policy. Opnum: 9
DeletePolicy	Deletes an existing policy. Opnum: 10
RenameAllocationPolicy	Renames a name allocated to a policy. Opnum: 11
MoveBefore	Moves a specified resource group before a reference resource group. Opnum: 12
MoveAfter	Moves a specified resource group after a reference resource group. Opnum: 13
SetCalDefaultPolicyName	Stores the ID of the default policy in the registry.

Method	Description
	Opnum: 14
GetCalDefaultPolicyName	Get the name of the default policy from the registry. Opnum: 15
GetProcessList	Gives the list of processes that will match to the policy. Opnum: 16
GetCurrentPolicy	Gets the active policy in XML format. Opnum: 17
SetCurrentPolicy	Sets the policy with the given ID as the current resource policy. Opnum: 18
GetCurrentStateAndActivePolicyName	Gets the name of the current resource policy and the management state in which WSRM is currently available. Opnum: 19
GetConditionalPolicy	Returns conditions from a specified resource policy. Opnum: 20
SetConditionalPolicy	Loads specified conditions into the current resource policy. Opnum: 21

3.2.4.7.1 GetPolicyInfo (Opnum 7)

The GetPolicyInfo method gets one or more specified resource allocation policies (RAP).

```
[id(1), helpstring("method GetPolicyInfo")] HRESULT GetPolicyInfo(
  [in] BSTR bstrPolicyName,
  [out] BSTR* pbstrPolicyInfo
);
```

bstrPolicyName: The name of the policy to get. If the string value is "\", this method returns a list of all allocation policies.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

pbstrPolicyInfo: A pointer to a string that returns the policy specified by *bstrPolicyName*, in the form of a Policy element (section 2.2.5.21). For an example, see the Policy example (section 4.2.19).

If *pbstrPolicyName* is "\", all policies MUST be returned, in the form of a PolicyCollection element (section 2.2.5.22).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Return value/code	Description
0xC1FF00C9 WRM_ERR_POLICYID_INVALID	The specified RAP does not exist.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.2 CreatePolicy (Opnum 8)

The CreatePolicy method creates a new resource allocation policy (RAP).

```
[id(2), helpstring("method CreatePolicy")] HRESULT CreatePolicy(
  [in] BSTR bstrPolicyInfo
);
```

bstrPolicyInfo: A string that specifies the new policy to be created in the form of a Policy element (section 2.2.5.21). For an example, see Policy Example (section 4.2.19).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen "-", cannot contain spaces, and cannot contain any of the following characters: `\ / ? * | : < > " , ;`
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<105>
0xC1FF00CA WRM_ERR_POLICYID_ALREADY_EXISTS	The request has been aborted because a RAP with the specified name already exists.
0xC1FF00DF WRM_ERR_POLICY_LIMIT_EXCEEDED	The request has been aborted because the total number of RAPs has exceeded an implementation-defined limit.<106>
0xC1FF00E3 WRM_ERR_CANNOT_CREATE_RESERVED_POLICY	A user created policy cannot have the same name as that of a built-in policy.
0xC1FF012E WRM_ERR_RESOURCEGROUPID_INVALID	The request has been aborted because the process matching criteria (PMC) name could not be found.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.3 ModifyPolicy (Opnum 9)

The ModifyPolicy method modifies an existing resource allocation policy (RAP).

```
[id(3), helpstring("method ModifyPolicy")] HRESULT ModifyPolicy(
  [in] BSTR bstrPolicyInfo,
  [in] BOOL bOverwrite
);
```

bstrPolicyInfo: A string that contains the policy to modify, in the form of a Policy element (section 2.2.5.21). For an example, see Policy example (section 4.2.19).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bOverwrite: A Boolean value that specifies whether to ignore the timestamp of the specified policy when validating.

A timestamp MUST be defined inside a common node at the root level of an XML element, as shown in the Calendar example (section 4.2.6). The format of a timestamp is specified in section 2.2.1.4.

Value	Meaning
FALSE 0x00000000	The timestamp of the new policy MUST specify a time that is later than or equal to the timestamp of any modifications made to a policy object on the server. Otherwise, the modification MUST fail, and **WRM_ERR_OLD_INFORMATION** MUST be returned.
TRUE 0x00000001	The policy is validated and modified without checking the timestamp.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF0069 WRM_ERR_OLD_INFORMATION	The XML timestamp is out of date.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters: \ / ? * | : < > " , ;
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<107>
0xC1FF00C9 WRM_ERR_POLICYID_INVALID	The specified RAP does not exist.
0xC1FF00E0	The specified policy is a built-in policy. It cannot be altered.

Return value/code	Description
WRM_ERR_WSRM_RESERVED_POLICY	
0xC1FF012E WRM_ERR_RESOURCEGROUPID_INVALID	The request has been aborted because the process matching criteria (PMC) name could not be found.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.4 DeletePolicy (Opnum 10)

The DeletePolicy method deletes an existing resource policy.

```
[id(4), helpstring("method DeletePolicy")] HRESULT DeletePolicy(
  [in] BSTR bstrPolicyName
```

);

bstrPolicyName: The name of the resource policy to be deleted.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters: \ / ? * | : < > " , ;
0xC1FF00CF WRM_ERR_IS_CURRENT_POLICY	This resource allocation policy (RAP) is being used by WSRM and cannot be deleted.
0xC1FF00E0 WRM_ERR_WSRM_RESERVED_POLICY	The specified policy is a built-in policy. It cannot be altered.
0xC1FF00E7 WRM_ERR_DELETING_POLICY	The specified policy could not be deleted. A policy cannot be deleted if it is a member of one or more conditional policies.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed. Windows returns this value if the XML data is corrupt.
0xC1FF00C9 WRM_ERR_POLICYID_INVALID	The specified RAP does not exist.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.5 RenameAllocationPolicy (Opnum 11)

The RenameAllocationPolicy method renames an existing resource allocation policy (RAP).

```
[id(5), helpstring("method RenameAllocationPolicy")] HRESULT RenameAllocationPolicy(
  [in] BSTR bstrOldPolicyNamebstrNewPolicyName,
  [in] BSTR bstrNewPolicyName
);
bstrOldPolicyName: The old policy name to be replaced.


If this parameter is NULL, E_INVALIDARG MUST be returned.


);
```

bstrNewPolicyName: The new policy name that replaces the old one.

If this parameter is NULL, E_INVALIDARG MUST be returned.

The following RAP names are reserved. If this parameter specifies a reserved name, **WRM_ERR_POLICYID_RESERVED_WORD** SHOULD be returned: "\", "current", "none", and "Residual". Reserved words are case-insensitive.

bstrOldPolicyName: The old policy name to be replaced.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters: \ / ? * | : < > " , ;
0xC1FF00C8 WRM_ERR_TOO_LONG_POLICY_ID	The request has been aborted, because the RAP name has exceeded an implementation-defined limit.<108>
0xC1FF00C9 WRM_ERR_POLICYID_INVALID	The specified RAP does not exist.
0xC1FF00CA WRM_ERR_POLICYID_ALREADY_EXISTS	A RAP with the specified name already exists.
0xC1FF00CB WRM_ERR_POLICYID_RESERVED_WORD	The specified RAP name is a reserved word used by WSRM and cannot be used as a name. Reserved words for RAPs are "current", "none", "\" and "Residual". Reserved words are case-insensitive.

Return value/code	Description
0xC1FF00E0 WRM_ERR_WSRM_RESERVED_POLICY	The specified policy is a built-in policy. It cannot be altered.
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<109>
0xC1FF00E5 WRM_ERR_RENAME_ACTIVE_POLICY	The specified RAP is being used by WSRM and cannot be renamed.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.6 MoveBefore (Opnum 12)

The MoveBefore method moves a specified resource group to a location just before a reference resource group.

```
[id(6), helpstring("method MoveBefore")] HRESULT MoveBefore(
  [in] BSTR bstrPolicyName,
  [in] BSTR bstrResourceGroupName,
  [in] BSTR bstrRefResourceGroupName
);
```

bstrPolicyName: The name of the policy in which the resource group is to be moved.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrResourceGroupName: The name of the resource group to be moved.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrRefResourceGroupName: The name of the reference resource group.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0xC1FF012E WRM_ERR_RESOURCEGROUPID_INVALID	The specified process matching criteria (PMC) does not exist.
0xC1FF00C9 WRM_ERR_POLICYID_INVALID	The specified resource allocation policy does not exist.
0xC1FF00E0 WRM_ERR_WSRM_RESERVED_POLICY	The specified policy is a built-in policy. It cannot be altered.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.7 MoveAfter (Opnum 13)

The MoveAfter method moves a specified resource group to a location just after a reference resource group.

```
[id(7), helpstring("method MoveAfter")] HRESULT MoveAfter(
  [in] BSTR bstrPolicyName,
  [in] BSTR bstrResourceGroupName,
  [in] BSTR bstrRefResourceGroupName
);
```

bstrPolicyName: The name of the policy in which resource group is to be moved.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrResourceGroupName: The name of the resource group to be moved.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrRefResourceGroupName: The name of the reference resource group.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF00E0 WRM_ERR_WSRM_RESERVED_POLICY	The specified policy is a built-in policy. It cannot be altered.
0xC1FF00C9 WRM_ERR_POLICYID_INVALID	The specified resource allocation policy does not exist.
0xC1FF012E WRM_ERR_RESOURCEGROUPID_INVALID	The specified process matching criteria (PMC) does not exist.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.8 SetCalDefaultPolicyName (Opnum 14)

The SetCalDefaultPolicyName method stores the name of the default resource allocation policy (RAP) in the registry.

```
[id(8), helpstring("method SetCalDefaultPolicyName")] HRESULT SetCalDefaultPolicyName(
  [in] BSTR bstrDefaultPolicyName
);
```

bstrDefaultPolicyName: The name of the policy to be used as the default policy.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF00C9 WRM_ERR_POLICYID_INVALID	The request has been aborted because the specified resource allocation policy does not exist.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters: `\ / ? * | : < > " , ;`

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.9 GetCalDefaultPolicyName (Opnum 15)

The GetCalDefaultPolicyName method used to get the name of the default resource allocation policy (RAP).

```
[id(9), helpstring("method GetCalDefaultPolicyName")] HRESULT GetCalDefaultPolicyName(
  [out] BSTR* pbstrDefaultPolicyName
);
```

pbstrDefaultPolicyName: A pointer to a string that returns the default RAP name.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0xC1FF038F WRM_NO_DEF_RAP	The WSRM server does not have a default resource allocation policy.<110>
0x00000000 S_OK	Operation successful.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.10 GetProcessList (Opnum 16)

The GetProcessList method returns a list of processes for a specified policy.

```
[id(10), helpstring("method GetProcessList")] HRESULT GetProcessList(
  [in] BSTR bstrPolicyName,
  [out] BSTR* pbstrProcessList
);
```

bstrPolicyName: Name of the policy for which a matching process list is to be returned.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

pbstrProcessList: A pointer to a string that returns a list of processes in the form of a ProcessList element (section 2.2.5.23). The data about matching processes is queried from the operating system to create the ProcessList element.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are not valid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters: \ / ? * | : < > " , ;
0xC1FF00C9 WRM_ERR_POLICYID_INVALID	The specified resource allocation policy does not exist.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.11 GetCurrentPolicy (Opnum 17)

The GetCurrentPolicy method returns the current resource policy.

```
[id(11), helpstring("method GetCurrentPolicy")] HRESULT GetCurrentPolicy(
  [out] BSTR* pbstrCurrentPolicyInfo,
  [out] MANAGEMENT_TYPE* enumManage
);
```

pbstrCurrentPolicyInfo: A pointer to a string that returns the current resource policy, in the form of a Policy element (section 2.2.5.21). For an example, see Policy example (section 4.2.19). If *enumManage* is **PROFILING**, this parameter returns the info of the policy whose name was specified when setting PROFILING mode using the **SetCurrentPolicy** method.

enumManage: A pointer to a MANAGEMENT_TYPE enumeration (section 2.2.3.5) value that returns the current mode in which the management service is operating.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.12 SetCurrentPolicy (Opnum 18)

The **SetCurrentPolicy** method sets the current resource policy to a specified resource policy by name.

```
[id(12), helpstring("method SetCurrentPolicy")] HRESULT SetCurrentPolicy(
  [in] BSTR bstrPolicyName,
  [in] MANAGEMENT_TYPE enumManage
);
```

bstrPolicyName: The name of the policy to become the current resource policy. If *enumManage* is **PROFILING**, the specified policy is used to gather PMC details required for accounting. The WSRM service categorizes every process based on the PMCs of the policy and collects data to serve accounting-related queries.

enumManage: The MANAGEMENT_TYPE enumeration (section 2.2.3.5) value that specifies the current mode in which the management service is operating. This function does not support calendar mode, so the value **CALENDAR_POLICY** is not valid for this parameter.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are not valid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters: \ / ? * | : < > " , ;
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF00C9 WRM_ERR_POLICYID_INVALID	The specified RAP does not exist.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.13 GetCurrentStateAndActivePolicyName (Opnum 19)

The GetCurrentStateAndActivePolicyName method returns the name of the current resource policy and the management state in which the WSRM management service is currently available.

```
[id(13), helpstring("method GetCurrentStateAndActivePolicyName")]
HRESULT GetCurrentStateAndActivePolicyName(
  [out] BSTR* pbstrCurrentPolicyName,
  [out] MANAGEMENT_TYPE* enumManage
);
```

pbstrCurrentPolicyName: A pointer to a string that returns the name of the current resource policy.

enumManage: A pointer to a MANAGEMENT_TYPE enumeration (section 2.2.3.5) value that returns the mode in which the management service is currently operating.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.14 GetConditionalPolicy (Opnum 20)

The GetConditionalPolicy function returns conditions for a specified conditional policy.

```
[id(14), helpstring("method GetConditionalPolicy")] HRESULT GetConditionalPolicy(
  [in] BSTR bstrPolicyName,
  [out] BSTR* pbstrPolicyInfo
);
```

bstrPolicyName: The name of the conditional policy for which conditions are to be returned.<111>

pbstrPolicyInfo: A pointer to a string that returns the resource policy conditions, in the form of a ConditionalPolicy element (section 2.2.5.12). For an example, see ConditionalPolicy example (section 4.2.9).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid or the specified policy name is not found.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.7.15 SetConditionalPolicy (Opnum 21)

The **SetConditionalPolicy** method loads specified conditions into the conditional policy, which contains the conditions and the respective **SwitchToPolicy** element values. When a specified condition occurs and the management mode is MANUAL_ACTIVE_POLICY, WSRM MUST start managing resources using the policy specified by the corresponding **SwitchToPolicy** element value.

```
[id(15), helpstring("method SetConditionalPolicy")] HRESULT SetConditionalPolicy(
  [in] BSTR bstrPolicyInfo
);
```

bstrPolicyInfo: A string that specifies the conditions to be loaded in the form of a ConditionalPolicy element (section 2.2.5.12). For an example, see ConditionalPolicy example (section 4.2.9).

If this parameter is NULL, **E_INVALIDARG** SHOULD be returned.<112>

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Additional IWRMPolicy interface methods are specified in section 3.2.4.7.

3.2.4.8 IWRMProtocol Interface

The IWRMProtocol interface defines the following methods.

Methods in RPC Opnum Order

Method	Description
GetSupportedClient	Gets a list of the clients that are supported by this server. Opnum: 7

3.2.4.8.1 GetSupportedClient (Opnum 7)

The GetSupportedClient method retrieves the level of support for clients on the WSRM server.

```
[id(1), helpstring("method GetSupportedClient")] HRESULT GetSupportedClient(
  [out] BSTR* pbstrSupportedClients
);
```

pbstrSupportedClients: A pointer to a string that returns the level of support for clients, in the format specified in SupportedClients (section 2.2.5.29).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IWRMProtocol interface methods are specified in section 3.2.4.8.

3.2.4.9 IWRMRemoteSessionMgmt Interface

The IWRMRemoteSessionMgmt provides methods that manage user category and remote session information.<113>

Methods in RPC Opnum Order

Method	Description
GetRemoteUserCategories	Retrieves user category information from the WSRM server. Opnum: 7

Method	Description
SetRemoteUserCategories	Sets user categories information on the WSRM server. Opnum: 8
RefreshRemoteSessionWeights (Opnum 9)	Updates the weights of remote sessions to force reallocation of CPU quotas. Opnum: 9

3.2.4.9.1 GetRemoteUserCategories (Opnum 7)

The GetRemoteUserCategories method retrieves user categories information from the WSRM server.

```
[id(1), helpstring("method GetRemoteUserCategories")] HRESULT GetRemoteUserCategories(
  [out] BSTR* pbstrRemoteUserCategoriesInfo
);
```

pbstrRemoteUserCategoriesInfo: A pointer to a string that returns categories of remote session users, in the format of a Users element (section 2.2.5.30).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.

Additional IWRMRemoteSessionMgmt interface methods are specified in section 3.2.4.9.

3.2.4.9.2 SetRemoteUserCategories (Opnum 8)

The SetRemoteUserCategories method sets user categories information on the WSRM server.

```
[id(2), helpstring("method SetRemoteUserCategories")] HRESULT SetRemoteUserCategories(
  [in] BSTR bstrRemoteUserCategoriesInfo
);
```

bstrRemoteUserCategoriesInfo: A string that specifies categories of remote session users, in the format of a Users element (section 2.2.5.30).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.

Return value/code	Description
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is invalid or cannot be processed.<114>

Additional IWRMRemoteSessionMgmt interface methods are specified in section 3.2.4.9.

3.2.4.9.3 RefreshRemoteSessionWeights (Opnum 9)

The RefreshRemoteSessionWeights method forces reallocation of CPU quotas for the sessions run by users according to the category type specified in *bstrTargetUserSessions*.

```
[id(3), helpstring("method RefreshRemoteSessionWeights")]
HRESULT RefreshRemoteSessionWeights(
  [in] BSTR bstrTargetUserSessionsbstrTaregetUserSessions,
  [in] BOOL bUpdateAll
);
```

bstrTargetUserSessions: A string that specifies a list of user categories, in the format of a ConfigurationFiles element (section 2.2.5.13).

bUpdateAll: A Boolean value that specifies whether to recursively delete all instances of the specified machine group.

Value	Meaning
FALSE 0x00000000	Only the CPU quotas for users specified in *bstrTargetUserSessions* are reallocated according to their category type.
TRUE 0x00000001	The CPU quotas for all remote sessions are reallocated according to the category type specified in *bstrTargetUserSessions*.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data is invalid or cannot be processed.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen (–), cannot contain spaces, and cannot contain any of the following characters: \ / ? * | : < > " , ;
0x00000000 S_OK	Operation successful.

Additional IWRMRemoteSessionMgmt interface methods are specified in section 3.2.4.9.

3.2.4.10 IWRMResourceGroup Interface

The IWRMResourceGroup interface defines the following methods.

Methods in RPC Opnum Order

Method	Description
GetResourceGroupInfo	Gets the information about the resource group with the specified ID. Opnum: 7
ModifyResourceGroup	Modifies an existing resource group. Opnum: 8
CreateResourceGroup	Creates a new resource group. Opnum: 9
DeleteResourceGroup	Deletes the resource group with the specified ID. Opnum: 10
RenameResourceGroup	Renames an existing resource group. Opnum: 11

3.2.4.10.1 GetResourceGroupInfo (Opnum 7)

The **GetResourceGroupInfo** method gets information about the resource group with the specified ID. If the ID is "\", this method returns all selection criteria.

```
[id(1), helpstring("method GetResourceGroupInfo")] HRESULT GetResourceGroupInfo(
  [in] BSTR bstrResourceGroupName,
  [out] BSTR* pbstrResourceGroupInfo
);
```

bstrResourceGroupName: A string that specifies the name of the selection criteria.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

pbstrResourceGroupInfo: A pointer to a string that returns the selection criteria, in the form of a ProcessMatchingCriteria element (section 2.2.5.24). For an example, see the ProcessMatchingCriteria example (section 4.2.20).

If *bstrResourceGroupName* is "\", all resource groups MUST be returned, in the form of a ProcessMatchingCriteriaCollection element (section 2.2.5.25). For an example, see the ProcessMatchingCriteriaCollection example (section 4.2.21).

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters; "\" cannot be used with other characters.

Return value/code	Description
	/ ? * | : < > " , ;
0xC1FF012E WRM_ERR_RESOURCEGROUPID_INVALID	The request has been aborted because the specified resource group does not exist.

Additional IWRMResourceGroup interface methods are specified in section 3.2.4.10.

3.2.4.10.2 ModifyResourceGroup (Opnum 8)

The **ModifyResourceGroup** method modifies an existing resource group.

```
[id(2), helpstring("method ModifyResourceGroup")] HRESULT ModifyResourceGroup(
  [in] BSTR bstrResourceGroupInfo,
  [in] BOOL bOverwrite
);
```

bstrResourceGroupInfo: A string that contains the modified resource group, in the form of a ProcessMatchingCriteria element (section 2.2.5.24). Sample XML is provided in ProcessMatchingCriteria example (section 4.2.20).

bOverwrite: A Boolean value that specifies whether to ignore the timestamp of the resource group when validating.

A timestamp MUST be defined inside a common node at the root level of an XML element, as shown in the Calendar example (section 4.2.6). The format of a timestamp is specified in section 2.2.1.4.

Value	Meaning
FALSE 0x00000000	The timestamp of the new resource group MUST specify a time that is equal to or later than the timestamp of any modifications made to the PMC object on the server. Otherwise, the modification SHOULD fail, and **WRM_ERR_OLD_INFORMATION** SHOULD be returned.
TRUE 0x00000001	The resource group is validated and modified without checking the timestamp.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	The call was successful.
0xC1FF0069 WRM_ERR_OLD_INFORMATION	The XML timestamp is out-of-date.
0xC1FF006A WRM_ERR_NO_TIMESTAMP_PRESENT	The specified resource group could not be modified because the XML timestamp in the *bstrResourceGroupInfo* parameter was not found.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name of the ProcessMatchingCriteria element contains characters that are not valid. The name cannot start with a hyphen ("-"), cannot contain spaces,

Return value/code	Description
	and cannot contain any of the following characters: `\ / ? * | : < > " , ;`
0xC1FF0070 WRM_ERR_TAGS_NOT_IN_ORDER	The XML data that is maintained by the management service is not valid or cannot be processed.<115>
0xC1FF012D WRM_ERR_TOO_LONG_RESOURCE_GROUP_ID	The resource group name has exceeded an implementation-defined<116> limit. The modify request has been aborted.
0xC1FF012E WRM_ERR_RESOURCEGROUPID_INVALID	The specified resource group does not exist, or "\" was specified. "\" is a reserved name for resource group identifiers.
0xC1FF0132 WRM_ERR_PATH_LIMIT_EXCEEDED	The command-line length has exceeded an implementation-defined limit.<117>
0xC1FF0134 WRM_ERR_USER_LIMIT_EXCEEDED	The user name or group value has exceeded an implementation-defined limit.<118>
0xC1FF0135 WRM_ERR_RULE_LIMIT_EXCEEDED	The specified resource group could not be modified because a resource group cannot have more than 64 rules.
0xC1FF0138 WRM_ERR_RESERVED_RESOURCEGROUP	The specified resource group is built-in. It cannot be modified.<119>

Additional IWRMResourceGroup interface methods are specified in section 3.2.4.10.

3.2.4.10.3 CreateResourceGroup (Opnum 9)

The CreateResourceGroup function creates a new resource group.

```
[id(3), helpstring("method CreateResourceGroup")] HRESULT CreateResourceGroup(
  [in] BSTR bstrResourceGroupInfo
);
```

bstrResourceGroupInfo: A string that specifies a new resource group, in the form of a ProcessMatchingCriteria (section 2.2.5.24) element (section 2.2.5.24). For an example, see ProcessMatchingCriteria example (section 4.2.20).

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	The operation was successfully done.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F	The specified name contains characters that

Return value/code	Description
WRM_ERR_ID_VALUE	are not valid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters: \ / ? * | : < > " , ;
0xC1FF012F WRM_ERR_RESOURCEGROUPID_ALREADY_EXISTS	The request has been aborted because the resource group with the specified name already exists.
0xC1FF0136 WRM_ERR_PMC_LIMIT_EXCEEDED	The total number of resource groups has exceeded an implementation-defined limit.<120>.
0xC1FF0139 WRM_ERR_CANNOT_CREATE_RESERVED_RESOURCEGROUP	A user-created resource group cannot have the same name as that of a built-in resource group.

Additional IWRMResourceGroup interface methods are specified in section 3.2.4.10.

3.2.4.10.4 DeleteResourceGroup (Opnum 10)

The DeleteResourceGroup deletes a resource group with the specified identifier.

```
[id(4), helpstring("method DeleteResourceGroup")] HRESULT DeleteResourceGroup(
  [in] BSTR bstrResourceGroupName
);
```

bstrResourceGroupName: A string that specifies the name of the resource group that is to be deleted.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000 S_OK	Operation successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are invalid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters: \ / ? * | : < > " , ;
0xC1FF012E WRM_ERR_RESOURCEGROUPID_INVALID	The specified resource group does not exist.
0xC1FF0137	resource group that are members of one or more RAPs cannot

Return value/code	Description
WRM_ERR_DELETING_RESOURCE_GROUP	be deleted.
0xC1FF0138 WRM_ERR_RESERVED_RESOURCEGROUP	The specified resource group is built-in. It cannot be modified.

Additional IWRMResourceGroup interface methods are specified in section 3.2.4.10.

3.2.4.10.5 RenameResourceGroup (Opnum 11)

The RenameResourceGroup renames an existing resource group.

```
[id(5), helpstring("method RenameResourceGroup")] HRESULT RenameResourceGroup(
  [in] BSTR bstrNewResourceGroupName,
  [in] BSTR bstrOldResourceGroupName,
  [in] BSTR );
```

bstrNewResourceGroupName: A string that specifies the new name of the resource group.

);

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrOldResourceGroupName: A string that specifies the name of the resource group to be renamed.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

bstrNewResourceGroupName: A string that specifies the new name of the resource group.

If this parameter is NULL, **E_INVALIDARG** MUST be returned.

Return Values: This method returns 0x00000000 for success or a negative HRESULT value (in the following table or in [MS-ERREF] section 2.1.1) if an error occurs.

Return value/code	Description
0x00000000	The call was successful.
0x80070057 E_INVALIDARG	One or more arguments are invalid.
0xC1FF006F WRM_ERR_ID_VALUE	The specified name contains characters that are valid. The name cannot start with a hyphen ("-"), cannot contain spaces, and cannot contain any of the following characters: \ / ? * | : < > " , ;
0xC1FF012D WRM_ERR_TOO_LONG_RESOURCE_GROUP_ID	The resource group name has exceeded an implementation-defined limit. <121>.
0xC1FF012E WRM_ERR_RESOURCEGROUPID_INVALID	The specified resource group does not exist.

Return value/code	Description
0Xc1FF012F WRM_ERR_RESOURCEGROUPID_ALREADY_EXISTS	A resource group with the specified name already exists.
0xC1FF0130 WRM_ERR_RESOURCEGROUPID_RESERVED_WORD	"\" is a reserved name for resource group identifiers.
0xC1FF0138 WRM_ERR_RESERVED_RESOURCEGROUP	The specified resource group is built-in. It cannot be modified.

Additional IWRMResourceGroup interface methods are specified in section 3.2.4.10.

3.2.5 Timer Events

Timer events are not specified for this interface.

3.2.6 Other Local Events

The server MUST track the following local events:

- Changes in Cluster Events to make conditional policies based on Cluster Events work.
- Addition of CPU and memory for those conditional policies based on such events.

4 Protocol Examples

4.1 Message Processing Examples

This section presents the following sample scenarios that illustrate the allocation of resources using methods of the Windows System Resource Manager (WSRM) Protocol.

Interface	Method	Section	Description
IWRMConfig	EnableDisable	4.1.1	Enable accounting on a local machine.
IResourceManager2	ExportXml	4.1.2	Export selected objects from the configuration.
IResourceManager2	GetImportConflicts	4.1.3	Check whether import will lead to conflicting objects.
IResourceManager2	ImportXml	4.1.4	Export selected policies to a machine group.
IWRMMachineGroup	AddMachine	4.1.5	Add a machine to a machine group.
IWRMMachineGroup	CreateMachineGroup	4.1.6	Create and initialize a machine group.
IWRMPolicy	CreatePolicy	4.1.7	Create a custom resource allocation policy (RAP).
IWRMPolicy	SetCurrentPolicy	4.1.8	Set the current resource policy.
IWRMResourceGroup	CreateResourceGroup	4.1.9	Create a custom process matching criteria (PMC).

In each scenario, it is assumed that a WSRM management service is running on the server machine.

4.1.1 Enable Accounting on a Local Machine

The following sequence of WSRM client actions can be used to enable accounting on a local machine:

1. Connect to the WSRM server by creating a **ResourceManager** COM object.
2. Get a pointer to the IWRMConfig interface.
3. Call the EnableDisable method with the following parameters:
   - *bEnableDisable* set to 1 to indicate **TRUE**.
   - *enumConfigType* set to **CONFIGTYPE_ACCOUNTING**.
4. Release the pointer to the interface.

4.1.2 Export Selected Objects from the Configuration

The following sequence of WSRM client actions can be used to export selected objects from the configuration:

1. Connect to the WSRM server by creating a **ResourceManager** COM object.
2. Get pointers to the IWRMMachineGroup, IResourceManager2 and other interfaces needed to gather the data of target objects.
3. Gather the names and types of all target objects.
4. Call the IResourceManager2 ExportObjects method with following parameters:

- *bstrObjectIds* set to the ids of the objects to export.
- *enumObjectType* set to the object type of the objects specified by *bstrObjectIds*.
- *pbstrObjectXml* set to a previously-allocated buffer that will receive the XML for the target objects.

5. Release all pointers to interfaces.

4.1.3 Check Whether Import Will Lead to Conflicting Objects

The following sequence of WSRM client actions can be used to determine whether the import of a given object will lead to conflicts.

1. Connect to the WSRM server by creating a **ResourceManager** COM object.
2. Get pointers to the IWRMMachineGroup, IResourceManager2 and other interfaces needed to gather the data of desired objects.
3. Gather the names and types of all target objects.
4. Call the IResourceManager2 ExportObjects method with following parameters:
   - *bstrObjectIds* set to the ids of the objects to check.
   - *enumObjectType* set to the object type of the objects specified by *bstrObjectIds*.
   - *pbstrObjectXml* set to a previously-allocated buffer that will receive the XML for the target objects.
5. Call the IResourceManager2 GetImportConflicts method with the following parameters:
   - *bstrPMCXml*, NULL or an XML string specifying desired PMC objects.
   - *bstrPolicyXml*, NULL or an XML string specifying desired resource policy objects.
   - *bstrCalendarXml*, NULL or an XML string specifying desired calendar objects.
   - *bstrConditionalXml*, set to NULL.
   - *bstrMachineGroupXml*, set to NULL.
   - *bstrConfigurationXml*, set to NULL.
   - *pbstrConflictingObjects*, set to a previously-allocated buffer that will receive the XML for the conflicting object identifiers.
6. Release all pointers to interfaces.

4.1.4 Export Selected Policies to a Machine Group

The following sequence of WSRM client actions can be used to export selected resource policies to a machine group.

1. Connect to the WSRM server by creating a **ResourceManager** COM object.
2. Get pointers to the IWRMMachineGroup and IResourceManager2 interfaces.
3. Gather the names of target resource policies.
4. Call the IResourceManager2 ExportObjects method with following parameters:

- *bstrObjectIds* set to the ids of the objects to check.
- *enumObjectType* set to the value **OBJECT_POLICY** from the OBJECT_TYPE enumeration.
- *pbstrObjectXml* set to a previously-allocated buffer that will receive the XML for the target objects.

5. Call the IWRMMachineGroup GetMachineGroupInfo method with the following parameters:
   - *bstrMachineGroupId*, set it to the name of the machine group that contains the resource policy.
   - *pbstrMachineGroupInfo*, to receive the data for the machine group.
6. Extract the names of all machines from the machine group, and connect to the WSRM servers running on those machines by creating remote **ResourceManager** COM objects.
7. Get a pointer to the IResourceManager2 interface.
8. Call the IResourceManager2 ImportXml method with following parameters:
   - *bstrPMCXml*, set to NULL.
   - *bstrPolicyXml*, set to the value of *pbstrObjectXml* returned from the prior call to IResourceManager2 ExportObjects.
   - *bstrCalendarXml*, set to NULL.
   - *bstrConditionalXml*, set to NULL.
   - *bstrMachineGroupXml*, set to NULL.
   - *enumImportType*, set to **SMART_MERGE_RENAME_EXISTING_IMPORT** from the IMPORT_TYPE enumeration.
9. Release all pointers to interfaces.

4.1.5 Add a Machine to a Machine Group

1. Connect to the WSRM server by creating a **ResourceManager** COM object.
2. Get a pointer to the IWRMMachineGroup interface.
3. Call the GetMachineGroupInfo method to identify the name of the machine group under which the machine will be added.
4. Call the AddMachine method with the following parameters:
   - *bstrParentMachineGroupId* set to the name of the machine group to which the machine will be added.
   - *bstrMachineInfo* set to the machine to be added, in the format of a Machine element (section 2.2.5.17).
5. Release the pointer to the interface.

4.1.6 Create And Initialize a Machine Group

The following sequence of WSRM client actions can be used to create and initialize a new machine group on local machine:

1. Connect to the WSRM server by creating a **ResourceManager** COM object.
2. Get a pointer to the IWRMMachineGroup interface.
3. Call the CreateMachineGroup method with the following parameters:
    - *bstrParentMachineGroupId* can be either "Machine Groups", "Default" or the name of any existing machine group. The new machine group will be created with this machine group as the parent.
    - *bstrMachineGroupInfo* set to the machine group to be created, in the format of a MachineGroup element (section 2.2.5.17).
4. Release the pointer to the interface.

4.1.7 Create a Custom Resource Allocation Policy

The following sequence of WSRM client actions can be used to create a custom RAP:

1. Connect to the WSRM server by creating a **ResourceManager** COM object.
2. Get a pointer to the IWRMPolicy interface.
3. Call the CreatePolicy method with the parameter *bstrPolicyInfo* set to the sample RAP string described in section 4.2.18.
4. Release the pointer to the interface.

4.1.8 Set the Current Resource Policy

The following sequence of WSRM client actions can be used to set the current resource policy to an existing RAP:

1. Connect to the WSRM server by creating a **ResourceManager** COM object.
2. Get a pointer to the IWRMPolicy interface.
3. Call the SetCurrentPolicy method with the following parameters:
    - *bstrPolicyName* set to "Equal_Per_Process".
    - *enumManage* set to **MANUAL_ACTIVE_POLICY**.
4. Release the pointer to the interface.

4.1.9 Create a Custom Process Matching Criteria

The following sequence of WSRM client actions can be used to create a custom process matching criteria (PMC):

1. Connect to the WSRM server by creating a **ResourceManager** COM object.
2. Get a pointer to the IWRMResourceGroup interface.
3. Call the CreateResourceGroup method with the parameter *bstrResourceGroupInfo* set to the sample PMC string described in section 4.2.21.
4. Release the pointer to the interface.

4.2 Sample XML Data

The WSRM Protocol specifies XML data formats for strings that are used in messages and method parameters. These formats are specified in XML Data Formats (section 2.2.5).

The following table lists the types of sample XML data that are presented in this section.

Name	Description
AccountingClientList (section 4.2.1)	Specifies a list of accounting clients.
AccountingConfigInfo (section 4.2.2)	Specifies configuration information concerning the accounting database.
AccountingMetaData (section 4.2.3)	Specifies accounting metadata, including database properties.
AccountingProcessList (section 4.2.4)	Specifies a list of accounting processes, including process properties.
AccountingQueryCondition (section 4.2.5)	Specifies a query on the accounting database.
Calendar (section 4.2.6)	Controls the scheduling of resource allocation.
Calendars (section 4.2.7)	Specifies one or more calendar objects.
CalendarsCollection (section 4.2.8)	Specifies a collection of calendar and schedule objects.
ConditionalPolicy (section 4.2.9)	Controls the selection of resource allocation policies (RAP).
ConfigurationFiles (section 4.2.10)	Specifies configurations used for the import and export of user information.
DependencyList (section 4.2.11)	Specifies dependent policies and calendar events.
Events (section 4.2.12)	Specifies a list of scheduled calendar events.
ExclusionList (section 4.2.13)	Specifies processes to exclude from management.
Machine (section 4.2.14)	Specifies parameters for managing machines within machine group.
MachineGroup (section 4.2.15)	Specifies parameters for managing machine groups.
NotificationConfigInfo (section 4.2.16)	Specifies the notification configuration.
Policy (section 4.2.19)	Specifies parameters for managing resource allocation.
ProcessMatchingCriteria (section 4.2.20)	Specifies parameters for resource management.
ProcessMatchingCriteriaCollection (section 4.2.21)	Specifies a collection of process matching criteria (PMC) elements.
Schedule (section 4.2.22)	Schedules a calendar event for resource management.
ServiceList (section 4.2.23)	Specifies services that are registered with the server.
SupportedClients (section 4.2.24)	Specifies the level of support for clients on the WSRM server.
Users (section 4.2.25)	Specifies categories of remote session users.

4.2.1 AccountingClientList Example

The following is an example of the AccountingClientList XML element (section 2.2.5.1).

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
<AccountingClientList>
    <AccountingClient Enabled="true">.</AccountingClient>
    <AccountingClient Enabled="true">wsrm001</AccountingClient>
</AccountingClientList>
```

Additional XML data examples are specified in section 4.2.

4.2.2 AccountingConfigInfo Example

The following is an example of the AccountingConfigInfo (section 2.2.5.2) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <AccountingConfigInfo>
    <AccountingEnabled>
      true
    </AccountingEnabled>
    <RecordWriteInterval>
      10
    </RecordWriteInterval>
    <DatabaseLocation>
      D:\Windows\system32\Windows System Resource Manager\DB
    </DatabaseLocation>
    <DatabaseServer>
      .
    </DatabaseServer>
    <DatabaseInstance>
      WSRM#MSSEE
    </DatabaseInstance>
  </AccountingConfigInfo>
```

Additional XML data examples are specified in section 4.2.

4.2.3 AccountingMetaData Example

The following is an example of the AccountingMetaData XML element (section 2.2.5.1).

```
<?xml version="1.0" ?>
<?WSRM version="1.0"?>
<AccountingMetaData>
    <Column>
        <BitmaskPosition>1</BitmaskPosition>
        <ColumnName>TimeStamp</ColumnName>
        <IsVisible>0</IsVisible>
        <IsArchivable>1</IsArchivable>
        <IsPivotable>0</IsPivotable>
        <ColumnType>1</ColumnType>
        <Units />
        <AggregationCollection>
            <Aggregation>
            <Pivot>ProcessName</Pivot>
            <AggregateFunction>MAX(</AggregateFunction>
            </Aggregation>
```

```
        </AggregationCollection>
    </Column>
</AccountingMetaData>
```

Additional XML data examples are specified in section 4.2.

4.2.4 AccountingProcessList Example

The following is an example of the AccountingProcessList XML element (section 2.2.5.4).

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
<AccountingProcessList>
    <Process>
        <EventType>L</EventType>
        <UserModeTime>156250</UserTime>
        <KernelModeTime>312500</KernelTime>
        <ReadOperationCount>11</ReadOperationCount>
        <WriteOperationCount>4</WriteOperationCount>
        <OtherOperationCount>341</OtherOperationCount>
        <ReadTransferCount>49690</ReadTransferCount>
        <WriteTransferCount>4</WriteTransferCount>
        <OtherTransferCount>7400</OtherTransferCount>
        <WorkingSetSize>1064960</WorkingSetSize>
        <VirtualSize>5070848</VirtualSize>
        <PrivatePageCount>421888</PrivatePageCount>
        <ImageName>smss.exe</ImageName>
        </ResourceGroupName>
        <UserName>SYSTEM</UserName>
        <DomainName>NT AUTHORITY</DomainName>
        <ImagePath>\SystemRoot\System32\smss.exe</ImagePath>
        <ProcessCommandLine>\SystemRoot\System32\smss.exe</ProcessCommandLine>
        <PolicyName>272</PolicyName>
        <CreationTime>128693080334531250</CreationTime>
        </CreationSystemTime>
        </PolicySetTime>
        </ProcessId>
        </ParentProcessId>
        <SessionId>0</SessionId>
        <ThreadCount>2</ThreadCount>
        <PageFaultCount>559</PageFaultCount>
        <PagefileUsage>421888</PagefileUsage>
        <PeakPagefileUsage>1118208</PeakPagefileUsage>
        <QuotaNonPagedPoolUsage>1504</QuotaNonPagedPoolUsage>
        <QuotaPagedPoolUsage>10520</QuotaPagedPoolUsage>
        <QuotaPeakNonPagedPoolUsage>6160</QuotaPeakNonPagedPoolUsage>
        <QuotaPeakPagedPoolUsage>36616</QuotaPeakPagedPoolUsage>
    </Process>
</AccountingProcessList>
```

Additional XML data examples are specified in section 4.2.

4.2.5 AccountingQueryCondition Example

The following is an example of the AccountingQueryCondition (section 2.2.5.5) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
     <AccountingQueryCondition>
     <SelectFieldCollection>
```

```
      <Column>ProcessName</Column>
      <Column>ComputerName</Column>
      <Column>OtherOperationCount</Column>
      <Column>UserName</Column>
      <Column>Domain</Column>
      <Column>PolicyName</Column>
    </SelectFieldCollection>
     <OrderColumnCollection>
         <OrderInfo>
          <Column>ProcessName</Column>
          <Column>ComputerName</Column>
          <Column>OtherOperationCount</Column>
          <Column>UserName</Column>
          <Column>Domain</Column>
          <Column>PolicyName</Column>
          <IsAscending>0</IsAscending>
        </OrderInfo>
     </OrderColumnCollection>
    </AccountingQueryCondition>
```

Additional XML data examples are specified in section 4.2.

4.2.6 Calendar Example

The following is an example of the Calendar (section 2.2.5.7) XML element.

```
<?xml version="1.0"?>
<!--TIMESTAMP WHEN FILE READ :29983232:4128238849: [CHANGING THIS VALUE CAN CAUSE UNDESIRED
EFFECTS]-->

<Calendar>
    <CalendarName>Monthly</CalendarName>
    <CalendarRule>
        <DtStart>2008-11-20</DtStart>
        <DtEnd/>
        <Freq>Monthly</Freq>
        <Interval>1</Interval>
        <ByDay>
            <Day>Su</Day>
        </ByDay>
        <ByMonth/>
        <ByMonthDay/>
        <ByYearDay/>
        <ByWeekNo/>
        <BySetPos>
            <SetPos>1</SetPos>
        </BySetPos>
    </CalendarRule>
    <ScheduleName>Untitled</ScheduleName>
    <ScheduleDtStart/>
    <ScheduleDtEnd/>
    <Description></Description>
</Calendar>
```

Additional XML data examples are specified in section 4.2.

4.2.7 Calendars Example

The following is an example of the Calendars (section 2.2.5.10) XML element.

```
<?xml version="1.0"?>
```

```
<!--TIMESTAMP WHEN FILE READ :29983232:4128238849: [CHANGING THIS VALUE CAN CAUSE UNDESIRED
EFFECTS]-->
<Calendars>
    <Calendar>
        <CalendarName>Untitled</CalendarName>
        <CalendarDate>2009-01-29</CalendarDate>
        <CalendarEvent>
            <PolicyName>Weighted_Sessions</PolicyName>
            <TmStart>02:12:00</TmStart>
            <DurationDays>0</DurationDays>
            <DurationHours>0</DurationHours>
            <DurationMinutes>30</DurationMinutes>
        </CalendarEvent>
        <ScheduleName></ScheduleName>
        <ScheduleDtStart/>
        <ScheduleDtEnd/>
        <Description></Description>
    </Calendar>
    <Calendar>
        <CalendarName>Untitled_rec</CalendarName>
        <CalendarRule>
            <DtStart>2009-01-29</DtStart>
            <DtEnd/>
            <Freq>Daily</Freq>
            <Interval>1</Interval>
            <ByDay/>
            <ByMonth/>
            <ByMonthDay/>
            <ByYearDay/>
            <ByWeekNo/>
            <BySetPos/>
        </CalendarRule>
        <ScheduleName>Untitled</ScheduleName>
        <ScheduleDtStart/>
        <ScheduleDtEnd/>
        <Description></Description>
    </Calendar>
</Calendars>
```

Additional XML data examples are specified in section 4.2.

4.2.8 CalendarsCollection Example

The following is an example of the CalendarsCollection (section 2.2.5.11) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <Calendars>
    <Calendars>
      <Calendar>
        <CalendarName>
          clitest_calevent1
        </CalendarName>
        <CalendarDate>
          2004-09-29
        </CalendarDate>
        <CalendarEvent>
          <PolicyName>
            CliTest_Pol1
          </PolicyName>
          <TmStart>
            11:07:00
          </TmStart>
          <DurationDays>
            0
```

```
            </DurationDays>
            <DurationHours>
              0
            </DurationHours>
            <DurationMinutes>
              30
            </DurationMinutes>
          </CalendarEvent>
          <ScheduleName/>
          <ScheduleDtStart/>
          <ScheduleDtEnd/>
        </Calendar>
        <ScheduleName/>
        <ScheduleDtStart/>
        <ScheduleDtEnd/>
      </Calendars>
      <Schedules>
        <Schedule>
          <ScheduleName>
            clitest_schedule1
          </ScheduleName>
          <CalendarEvent>
            <PolicyName>
              CliTest_Pol1
            </PolicyName>
            <TmStart>
              01:00:00
            </TmStart>
            <DurationDays>
              0
            </DurationDays>
            <DurationHours>
              1
            </DurationHours>
            <DurationMinutes>
              0
            </DurationMinutes>
          </CalendarEvent>
        </Schedule>
      </Schedules>
    </Calendars>
```

Additional XML data examples are specified in section 4.2.

4.2.9 ConditionalPolicy Example

The following is an example of the ConditionalPolicy (section 2.2.5.12) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <ConditionalPolicy Name=" GlobalCP ">
    <ConditionsList>
      <Condition Active="true" ID="0" Category="Processor" Name="ProcessorNew">
        <ConditionString ConditionState="ProcessorNew" Operator="New"/>
        <ConditionRelationship Operator="AND" NextNode="ConditionString"/>
        <ConditionString ConditionState="ProcessorGT" Operator="GT" Num="4"/>
        <ConditionRelationship Operator="None" NextNode="None"/>
        <ConditionEvaluation Result="false"/>
        <Action>
          <SwitchToPolicy>
            No Action</SwitchToPolicy>
        </Action>
      </Condition>
      <Condition Active="true" ID="1" Category="Processor" Name="ProcessorNew">
        <ConditionString ConditionState="ProcessorNew" Operator="New"/>
        <ConditionRelationship Operator="AND" NextNode="ConditionString"/>
```

```
          <ConditionString ConditionState="ANY"/>
          <ConditionRelationship Operator="None" NextNode="None"/>
          <ConditionEvaluation Result="false"/>
          <Action>
            <SwitchToPolicy>
              Equal_Per_Process</SwitchToPolicy>
          </Action>
        </Condition>
      </ConditionsList>
    </ConditionalPolicy>
```

Additional XML data examples are specified in section 4.2.

4.2.10 ConfigurationFiles Example

The following is an example of the ConfigurationFiles (section 2.2.5.13) XML element.

```
<?xml version="1.0"?>
<?WSRM version="3.0"?>
<ConfigurationFiles>
    <ConfigurationFile FileName="UserCategories.xml">
        <Users>
            <Category Type="Premium">
                <User DisplayName="Sanjeev Kumar" ID="sanjkuma"
                    Domain="SampleDomain" Type="Individual"
                    Description="" />
                <User DisplayName="Senior Management" ID="srmnmgmt"
                    Domain="SampleDomain" Type="Group"
                    Description="" />
            </Category>
            <Category Type="Standard">
                <User DisplayName="Ankur Kulshrestha" ID="ankurkul"
                    Domain="SampleDomain" Type="Individual"
                    Description="" />
                <User DisplayName="Developer Divisions" ID="DeveDivs"
                    Domain="SampleDomain" Type="Group"
                    Description="" />
            </Category>
            <Category Type="Basic">
                <User DisplayName="Ashutosh Akhouri" ID="aakhouri"
                    Domain="SampleDomain" Type="Individual"
                    Description="" />
                <User DisplayName="Trainee Department" ID="trainees"
                    Domain="SampleDomain" Type="Group"
                    Description="" />
            </Category>
        </Users>
    </ConfigurationFile>
</ConfigurationFiles>
```

Additional XML data examples are specified in section 4.2.

4.2.11 DependencyList Example

The following is an example of the DependencyList XML element (section 2.2.5.14).

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <DependencyList>

    <ScheduleName>sched1</ScheduleName>
    <CalendarName>cal2</CalendarName>
```

```
      <Condition Num="NodeName">MSCSNodeUp</Condition>
    </DependencyList>
```

Additional XML data examples are specified in section 4.2.

4.2.12 Events Example

The following is an example of the Events (section 2.2.5.15) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <Events>
    <EventData>
      <DtTmStart>
        2004-08-14T00:00:00
      </DtTmStart>
      <DtTmEnd>
        2004-08-15T11:07:00
      </DtTmEnd>
      <PolicyName/>
      <CalendarName/>
    </EventData>
    <EventData>
      <DtTmStart>
        2004-08-15T11:07:00
      </DtTmStart>
      <DtTmEnd>
        2004-08-15T11:37:00
      </DtTmEnd>
      <PolicyName>
        CliTest_Pol5
      </PolicyName>
      <CalendarName>
        clitest_calevent5
      </CalendarName>
    </EventData>
    <EventData>
      <DtTmStart>
        2004-08-15T11:37:00
      </DtTmStart>
      <DtTmEnd>
        2004-08-17T00:00:00
      </DtTmEnd>
      <PolicyName/>
      <CalendarName/>
    </EventData>
  </Events>
```

Additional XML data examples are specified in section 4.2.

4.2.13 ExclusionList Example

The following is an example of the ExclusionList (section 2.2.5.16) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <ExclusionList>
    <Process>
      System
    </Process>
    <Process>
      %windir%\system32\csrss.exe
    </Process>
```

```
    <Process>
      %windir%\system32\dfssvc.exe
    </Process>
    <Process>
      %windir%\system32\dumprep.exe
    </Process>
    <Process>
      %windir%\system32\dwwin.exe
    </Process>
    <Process>
      %windir%\system32\llssrv.exe
    </Process>
    <Process>
      %windir%\system32\lsass.exe
    </Process>
    <Process>
      %windir%\system32\services.exe
    </Process>
    <Process>
      %windir%\system32\smss.exe
    </Process>
    <Process>
      %windir%\system32\spoolsv.exe
    </Process>
    <Process>
      %windir%\system32\winlogon.exe
    </Process>
    <Process>
      %windir%\system32\wsrm.exe
    </Process>
    <Process>
      %windir%\system32\msdtc.exe
    </Process>
    <Process>
      %windir%\system32\Windows System Resource Manager\bin\wsrmc.exe
    </Process>
    <Process>
      %windir%\system32\taskmgr.exe
    </Process>
  </ExclusionList>
```

Additional XML data examples are specified in section 4.2.

4.2.14 Machine Example

The following is an example of the Machine (section 2.2.5.17) XML element.

```
<Machine Name="WSRMServer01" Description="" FQDN="CN=WSRMServer01,
OU=Workstations,OU=Machines,DC=fareast,DC=corp,DC=microsoft,DC=com"
OS="Windows Server 7 Enterprise" OSVersion="6.1 (6759)"
LastUpdatedTime="2008:8:1:11:23:12:23:664"/>
```

Additional XML data examples are specified in section 4.2.

4.2.15 MachineGroup Example

The following is an example of the MachineGroup (section 2.2.5.18) XML element.

```
<?xml version="1.0"?>
<?WSRM version="3.0"?>
<MachineGroup Name="Machine Groups">
    <MachineGroup Name="Default"
```

```
            Description="Default Machine Group"/>
        <MachineGroup Name="Company Servers"
              Description="All Servers in my company">
            <MachineGroup Name="File Server Group"
                  Description="These are the File Servers ">
                <Machine Name="FS1" Description="" />
                <Machine Name="FS2" Description="" />
            </MachineGroup>
            <MachineGroup Name="Communication Servers"
                  Description="These are the Communication Servers">
                <Machine Name="CS1" Description="" />
                <Machine Name="CS2" Description="" />
            </MachineGroup>
            <MachineGroup Name="Backup Servers"
                  Description="These are the backup servers">
                <Machine Name="BS1" Description="" />
            </MachineGroup>
        </MachineGroup>
    </MachineGroup>
```

Additional XML data examples are specified in section 4.2.

4.2.16 NotificationConfigInfo Example

The following is an example of the NotificationConfigInfo XML element (section 2.2.5.19).

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
<NotificationConfigInfo>
    <NotificationEnabled>true</NotificationEnabled>
    <SMTPServer>this</SMTPServer>
    <EMailIds>ankurkul@microsoft.com</EMailIds>
    <EventList>107,109,112</EventList>
</NotificationConfigInfo>
```

Additional XML data examples are specified in section 4.2.

4.2.17 ObjectIds Example

The following is an example of the ObjectIds (section 2.2.5.20) XML element with multiple types.

```
<?xml version="1.0"?>
<?WSRM version="3.0"?>
<ObjectIds>
    <ObjectId Type="PMC">PMC1</ObjectId>
    <ObjectId Type="Policy">Pol1</ObjectId>
    <ObjectId Type="Calendar">Cal1</ObjectId>
    <ObjectId Type="Schedule">Sched1</ObjectId>
</ObjectIds>
```

The following is an example of the ObjectIds (section 2.2.5.20) XML element with a single type.

```
<?xml version="1.0"?>
<?WSRM version="3.0"?>
<ObjectIds>
    <ObjectId Type="PMC">PMC1</ObjectId>
```

```
    <ObjectId Type="PMC">PMC2</ObjectId>
    <ObjectId Type="PMC">PMC3</ObjectId>
    <ObjectId Type="PMC">PMC4</ObjectId>
</ObjectIds>
```

Additional XML data examples are specified in section 4.2.

4.2.18 Policy Example

The following is an example of the Policy (section 2.2.5.21) XML element.

```
<Policy Name="CliTest_Pol1">
    <AllocationCriteria Name="CliTest_MC1">
        <ProcessMatchingCriteria RefName="CliTest_MC1"/>
        <CPUAllocation>
            10
        </CPUAllocation>
        <MaximumWorkingSet>
            110
        </MaximumWorkingSet>
        <MaximumCommittedMemory>
            100
        </MaximumCommittedMemory>
        <CommittedMemoryExceededOption>
            TerminateApp
        </CommittedMemoryExceededOption>
    </AllocationCriteria>
    <AllocationCriteria Name="CliTest_MC2">
        <ProcessMatchingCriteria RefName="CliTest_MC2"/>
        <CPUAllocation>
            15
        </CPUAllocation>
        <MaximumWorkingSet>
            110
        </MaximumWorkingSet>
        <MaximumCommittedMemory>
            100
        </MaximumCommittedMemory>
        <CommittedMemoryExceededOption>
            TerminateApp
        </CommittedMemoryExceededOption>
    </AllocationCriteria>
    <Description />
</Policy>
```

Additional XML data examples are described in section 4.2.

4.2.19 Policy Collection Example

The following is an example of the Policy (section 2.2.5.21) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <Policy>
    <Policy Name="CliTest_Pol1">
      <AllocationCriteria Name="CliTest_MC1">
        <ProcessMatchingCriteria RefName="CliTest_MC1"/>
        <CPUAllocation>
          10
```

```
        </CPUAllocation>
        <MaximumWorkingSet>
          110
        </MaximumWorkingSet>
        <MaximumCommittedMemory>
          100
        </MaximumCommittedMemory>
        <CommittedMemoryExceededOption>
          TerminateApp
        </CommittedMemoryExceededOption>
      </AllocationCriteria>
      <AllocationCriteria Name="CliTest_MC2">
        <ProcessMatchingCriteria RefName="CliTest_MC2"/>
        <CPUAllocation>
          15
        </CPUAllocation>
        <MaximumWorkingSet>
          110
        </MaximumWorkingSet>
        <MaximumCommittedMemory>
          100
        </MaximumCommittedMemory>
        <CommittedMemoryExceededOption>
          TerminateApp
        </CommittedMemoryExceededOption>
      </AllocationCriteria>
    </Policy>
  </Policy>
```

Additional XML data examples are specified in section 4.2.

4.2.20 ProcessMatchingCriteria Example

The following is an example of process matching criteria (PMC) specified with the ProcessMatchingCriteria (section 2.2.5.24) XML element.

```
<?xml version="1.0"?>
<ProcessMatchingCriteria Name="AWSTART_PMC">
    <Rule>
        <Path>w*;a*</Path>
        <User/>
    </Rule>
    <Description></Description>
</ProcessMatchingCriteria>
```

Additional XML data examples are specified in section 4.2.

4.2.21 ProcessMatchingCriteriaCollection Example

The following is an example of a ProcessMatchingCriteriaCollection (section 2.2.5.25) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <ProcessMatchingCriteriaCollection>
    <ProcessMatchingCriteria Name="PmcUsedAsDefault">
      <Rule>
        <Path>*</Path>
        <User/>
      </Rule>
    </ProcessMatchingCriteria>
```

```
    <ProcessMatchingCriteria Name="CliTest_MC1">
      <Rule>
        <Path>clitest_abcd1.exe</Path>
        <User/>
      </Rule>
    </ProcessMatchingCriteria>
    <ProcessMatchingCriteria Name="CliTest_MC2">
      <Rule>
        <Path>clitest_abcd2.exe</Path>
        <User/>
      </Rule>
    </ProcessMatchingCriteria>
  </ProcessMatchingCriteriaCollection>
```

Additional XML data examples are specified in section 4.2.

4.2.22 Schedule Example

The following is an example of the Schedule (section 2.2.5.26) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <Schedule>
    <ScheduleName>
      clitest_schedule3
    </ScheduleName>
    <CalendarEvent>
      <PolicyName>
        CliTest_Pol3
      </PolicyName>
      <TmStart>
        01:00:00
      </TmStart>
      <DurationDays>
        0
      </DurationDays>
      <DurationHours>
        1
      </DurationHours>
      <DurationMinutes>
        0
      </DurationMinutes>
    </CalendarEvent>
    <Description>
    </Description>
  </Schedule>
```

Additional XML data examples are specified in section 4.2.

4.2.23 ServiceList Example

The following is an example of the ServiceList (section 2.2.5.28) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
  <ServiceList>
    <Service>
      <Name>
        ALG
      </Name>
      <StartName>
        NT AUTHORITY\LocalService
      </StartName>
```

```
        <Path>
          D:\Windows\System32\alg.exe
        </Path>
        <Description>
          Provides support for 3rd party protocol plug-ins for Internet
          Connection Sharing
        </Description>
      </Service>
      <Service>
        <Name>
          wudfsvc
        </Name>
        <StartName>
          LocalSystem
        </StartName>
        <Path>
          D:\Windows\system32\svchost.exe -k LocalSystemNetworkRestricted
        </Path>
        <Description>
          Manages user-mode driver host processes
        </Description>
      </Service>
    </ServiceList>
```

Additional XML data examples are specified in section 4.2.

4.2.24 SupportedClients Example

The following is an example of the SupportedClients (section 2.2.5.29) XML element.

```
<?xml version="1.0"?>
<?WSRM version="1.0"?>
<SupportedClients>
    <Client>V1</Client>
    <Client>V2</Client>
    <Client>V3</Client>
</SupportedClients>
```

Additional XML data examples are specified in section 4.2.

4.2.25 Users Example

The following is an example of the Users (section 2.2.5.30) XML element.

```
<?xml version="1.0"?>
<?WSRM version="3.0"?>
<Users>
    <Category Type="Premium">
        <User DisplayName="Sanjeev Kumar" ID="sanjkuma"
             Domain="SampleDomain" Type="Individual" Description=""/>
        <User DisplayName="Senior Management" ID="srmnmgmt"
             Domain="SampleDomain" Type="Group" Description=""/>
    </Category>
    <Category Type="Standard">
        <User DisplayName="Ankur Kulshrestha" ID="ankurkul"
             Domain="SampleDomain" Type="Individual" Description=""/>
        <User DisplayName="Developer Divisions" ID="DeveDivs"
             Domain="SampleDomain" Type="Group" Description=""/>
    </Category>
    <Category Type="Basic">
        <User DisplayName="Ashutosh Akhouri" ID="aakhouri"
             Domain="SampleDomain" Type="Individual" Description=""/>
```

```
            <User DisplayName="Trainee Department" ID="trainees"
                 Domain="SampleDomain" Type="Group" Description=""/>
        </Category>
    </Users>
```

Additional XML data examples are specified in section 4.2.

5 Security

5.1 Security Considerations for Implementers

WSRM allows any user who has administrator privileges on a machine to connect to a server. Therefore, a user can exploit security vulnerabilities in the WSRM server implementation. A secure server implementation enforces security on each method.

5.2 Index of Security Parameters

Security parameter	Section
Authentication protocol	1.7

6 Appendix A: Full IDL

For ease of implementation, the full **IDL** is provided as follows, where "ms-dtyp.idl" is the IDL found in [MS-DTYP] appendix A, and "ms-oaut.idl" is the IDL found in [MS-OAUT] appendix A:

```
import "ms-dtyp.idl";
import "ms-oaut.idl";

typedef [v1_enum] enum
{
    CONFIGTYPE_ACCOUNTING       = 1,
    CONFIGTYPE_NOTIFICATION     = 2,
    CONFIGTYPE_CALENDARING      = 3
}
CONFIGTYPE;

typedef [v1_enum] enum
{
    RESTORE_LAST_GOOD_STATE     = 1,
    RESTORE_EMPTY_FILES         = 2
}
RESTORE_MODE;

typedef [v1_enum] enum
{
    OBJECT_SELECTION_CRITERIA   = 1,
    OBJECT_POLICY               = 2,
    OBJECT_SCHEDULE             = 3
}
OBJECT_TYPE;

typedef [v1_enum] enum
{
    MANUAL_ACTIVE_POLICY        = 1,
    CALENDAR_POLICY             = 2,
    PROFILING                   = 3
}
MANAGEMENT_TYPE;

typedef [v1_enum] enum
{
    SYSTEM_EXCLUSION_LIST       = 1,
    USER_EXCLUSION_LIST         = 2,
    DEFAULT_USER_EXCLUSION_LIST = 4
}
EXCLUSIONLIST_TYPE;

typedef [v1_enum] enum
{
    OVERWRITE_IMPORT                    = 1,
    IGNORE_EXISTING_IMPORT              = 2,
    OVERRIDE_EXISTING_IMPORT            = 3,
    SMART_MERGE_RENAME_EXISTING_IMPORT  = 4,
    SMART_MERGE_RENAME_IMPORTED_IMPORT  = 5,
}
IMPORT_TYPE;

typedef [v1_enum] enum
{
    OVERWRITE_MG_MERGE_OPTION           = 1,
    OVERRIDE_MG_MERGE_OPTION            = 2,
    APPEND_MG_MERGE_OPTION              = 3,
    SMART_MG_MERGE_OPTION               = 4,
}
MACHINE_GROUP_MERGE_OPTIONS;

[
```

```
    uuid(E8BCFFAC-B864-4574-B2E8-F1FB21DFDC18),
    helpstring("ResourceManager Class")
]
coclass ResourceManager
{
    [default] interface IResourceManager;
    interface IWRMCalendar;
    interface IWRMPolicy;
    interface IWRMResourceGroup;
    interface IWRMAccounting;
    interface IWRMConfig;
    interface IWRMProtocol;
    interface IWRMMachineGroup;
    interface IResourceManager2;
    interface IWRMRemoteSessionMgmt;

};

[
    object,
    uuid(C5CEBEE2-9DF5-4CDD-A08C-C2471BC144B4),
    dual,
    helpstring("IResourceManager Interface"),
    pointer_default(unique)
]
interface IResourceManager : IDispatch
{
    [id(1), helpstring("method RetrieveEventList")]
        HRESULT RetrieveEventList(
            [out] BSTR *pbstrEventList);
    [id(2), helpstring("method GetSystemAffinity")]
        HRESULT GetSystemAffinity(
            [out] DWORD64 *pdwSysAffinity);
    [id(3), helpstring("method ImportXMLFiles")]
        HRESULT ImportXMLFiles(
            [in] BSTR bstrPMCXml,
            [in] BSTR bstrPolicyXml,
            [in] BSTR bstrCalendarXml,
            [in] BSTR bstrConditionalXml);
    [id(4), helpstring("method ExportXMLFiles")]
        HRESULT ExportXMLFiles(
            [out] BSTR *pbstrPMCXml,
            [out] BSTR *pbstrPolicyXml,
            [out] BSTR *pbstrCalendarXml,
            [out] BSTR *pbstrConditionalXml);
    [id(5), helpstring("method RestoreXMLFiles")]
        HRESULT RestoreXMLFiles(
            [in] RESTORE_MODE enumRestore);
    [id(6), helpstring("method GetDependencies")]
        HRESULT GetDependencies(
            [in] BSTR bstrObjectName,
            [in] OBJECT_TYPE enumObject,
            [out] BSTR *pbstrDependencyList);
    [id(7), helpstring("method GetServiceList")]
        HRESULT GetServiceList(
            [out] BSTR *pbstrServiceList);
    [id(8), helpstring("method GetIISAppPoolNames")]
        HRESULT GetIISAppPoolNames(
            [out] BSTR *pbstrIISAppPoolList,
            [out] BSTR *pbstrSystemDirectory);
    [id(9), helpstring("method GetServerName")]
        HRESULT GetServerName(
            [out] BSTR *pbstrServerName);
    [id(10),helpstring("method GetCurrentMemory")]
        HRESULT GetCurrentMemory(
            [out] DWORD64 *pdwCurrMemory);
};

[
    object,
```

```
    uuid(481E06CF-AB04-4498-8FFE-124A0A34296D),
    dual,
    helpstring("IWRMCalendar Interface"),
    pointer_default(unique)
]
interface IWRMCalendar : IDispatch
{
    [id(1), helpstring("method GetCalendarInfo")]
        HRESULT GetCalendarInfo(
            [in] BSTR  bstrCalendarName,
            [out] BSTR* pbstrCalendarXML);
    [id(2), helpstring("method CreateCalendar")]
        HRESULT CreateCalendar(
            [in] BSTR bstrCalendarXML,
            [in] BOOL bChangeActivePolicy);
    [id(3), helpstring("method ModifyCalendar")]
        HRESULT ModifyCalendar(
            [in] BSTR bstrCalendarXML,
            [in] BOOL bOverwrite,
            [in] BOOL bChangeActivePolicy);
    [id(4), helpstring("method DeleteCalendar")]
        HRESULT DeleteCalendar(
            [in] BSTR bstrCalendarName,
            [in] BOOL bChangeActivePolicy);
    [id(5), helpstring("method RenameCalendar")]
        HRESULT RenameCalendar(
            [in] BSTR bstrOldCalendarName,
            [in] BSTR bstrNewCalendarName);
    [id(6), helpstring("method ComputeEvents")]
        HRESULT ComputeEvents(
            [in] BSTR szStartTime,
            [in] BSTR szEndTime,
            [in] BOOL fMergeEvents,
            [out] BSTR *pbstrEvents,
            [out] BSTR *pbstrConflicts);
    [id(7), helpstring("method GetScheduleInfo")]
        HRESULT GetScheduleInfo(
            [in] BSTR  bstrScheduleName,
            [out] BSTR* pbstrScheduleXML);
    [id(8), helpstring("method CreateSchedule")]
        HRESULT CreateSchedule(
            [in] BSTR bstrScheduleXML);
    [id(9), helpstring("method ModifySchedule")]
        HRESULT ModifySchedule(
            [in] BSTR bstrScheduleXML,
            [in] BOOL bOverwrite,
            [in] BOOL bChangeActivePolicy);
    [id(10), helpstring("method DeleteSchedule")]
        HRESULT DeleteSchedule(
            [in] BSTR bstrScheduleName);
    [id(11), helpstring("method RenameSchedule")]
        HRESULT RenameSchedule(
            [in] BSTR bstrOldScheduleName,
            [in] BSTR bstrNewScheduleName);
    [id(12), helpstring("method MoveBeforeCalendar")]
        HRESULT MoveBeforeCalendar(
            [in] BSTR bstrCalendarName,
            [in] BSTR bstrRefCalendarName,
            [in] BOOL bChangeActivePolicy);
    [id(13), helpstring("method MoveAfterCalendar")]
        HRESULT MoveAfterCalendar(
            [in] BSTR bstrCalendarName,
            [in] BSTR bstrRefCalendarName,
            [in] BOOL bChangeActivePolicy);
    [id(14), helpstring("method GetServerTimeZone")]
        HRESULT GetServerTimeZone(
            [out] int* pnServerTimeZone);
};

[
```

```
    object,
    uuid(59602EB6-57B0-4FD8-AA4B-EBF06971FE15),
    dual,
    helpstring("IWRMPolicy Interface"),
    pointer_default(unique)
]
interface IWRMPolicy: IDispatch
{
    [id(1), helpstring("method GetPolicyInfo")]
        HRESULT GetPolicyInfo(
            [in] BSTR bstrPolicyName,
            [out] BSTR *pbstrPolicyInfo);
    [id(2), helpstring("method CreatePolicy")]
        HRESULT CreatePolicy(
            [in] BSTR bstrPolicyInfo);
    [id(3), helpstring("method ModifyPolicy")]
        HRESULT ModifyPolicy(
            [in] BSTR bstrPolicyInfo,
            [in] BOOL bOverwrite);
    [id(4), helpstring("method DeletePolicy")]
        HRESULT DeletePolicy(
            [in] BSTR bstrPolicyName);
    [id(5), helpstring("method RenameAllocationPolicy")]
        HRESULT RenameAllocationPolicy(
            [in] BSTR bstrNewPolicyName,
            [in] BSTR bstrOldPolicyName);
    [id(6), helpstring("method MoveBefore")]
        HRESULT MoveBefore(
            [in] BSTR bstrPolicyName,
            [in] BSTR bstrResourceGroupName,
            [in] BSTR bstrRefResourceGroupName);
    [id(7), helpstring("method MoveAfter")]
        HRESULT MoveAfter(
            [in] BSTR bstrPolicyName,
            [in] BSTR bstrResourceGroupName,
            [in] BSTR bstrRefResourceGroupName);
    [id(8), helpstring("method SetCalDefaultPolicyName")]
        HRESULT SetCalDefaultPolicyName(
            [in] BSTR bstrDefaultPolicyName);
    [id(9), helpstring("method GetCalDefaultPolicyName")]
        HRESULT GetCalDefaultPolicyName(
            [out] BSTR *pbstrDefaultPolicyName);
    [id(10), helpstring("method GetProcessList")]
        HRESULT GetProcessList(
            [in] BSTR bstrPolicyName,
            [out] BSTR *pbstrProcessList);
    [id(11), helpstring("method GetCurrentPolicy")]
        HRESULT GetCurrentPolicy(
            [out] BSTR *pbstrCurrentPolicyInfo,
            [out] MANAGEMENT_TYPE *enumManage);
    [id(12), helpstring("method SetCurrentPolicy")]
        HRESULT SetCurrentPolicy(
            [in] BSTR bstrPolicyName,
            [in] MANAGEMENT_TYPE enumManage);
    [id(13), helpstring("method GetCurrentStateAndActivePolicyName")]
        HRESULT GetCurrentStateAndActivePolicyName(
            [out] BSTR *pbstrCurrentPolicyName,
            [out] MANAGEMENT_TYPE *enumManage);
    [id(14), helpstring("method GetConditionalPolicy")]
        HRESULT GetConditionalPolicy(
            [in] BSTR bstrPolicyName,
            [out] BSTR *pbstrPolicyInfo);
    [id(15), helpstring("method ~~SetCondtionalPolicy~~SetConditionalPolicy")]
        HRESULT SetConditionalPolicy(
            [in] BSTR bstrPolicyInfo);
};

[
    object,
    uuid(BC681469-9DD9-4BF4-9B3D-709F69EFE431),
```

```
    dual,
    helpstring("IWRMResourceGroup Interface"),
    pointer_default(unique)
]
interface IWRMResourceGroup: IDispatch
{
    [id(1), helpstring("method GetResourceGroupInfo")]
        HRESULT GetResourceGroupInfo(
            [in] BSTR bstrResourceGroupName,
            [out] BSTR *pbstrResourceGroupInfo);
    [id(2), helpstring("method ModifyResourceGroup")]
        HRESULT ModifyResourceGroup(
            [in] BSTR bstrResourceGroupInfo,
            [in] BOOL bOverwrite);
    [id(3), helpstring("method CreateResourceGroup")]
        HRESULT CreateResourceGroup(
            [in] BSTR bstrResourceGroupInfo);
    [id(4), helpstring("method DeleteResourceGroup")]
        HRESULT DeleteResourceGroup(
            [in] BSTR bstrResourceGroupName);
    [id(5), helpstring("method RenameResourceGroup")]
        HRESULT RenameResourceGroup(
            [in] BSTR bstrNewResourceGroupName,
            [in] BSTR bstrOldResourceGroupName);
};

[
    object,
    uuid(4F7CA01C-A9E5-45B6-B142-2332A1339C1D),
    dual,
    helpstring("IWRMAccounting Interface"),
    pointer_default(unique)
]
interface IWRMAccounting: IDispatch
{
    [id(1), helpstring("method CreateAccountingDb")]
        HRESULT CreateAccountingDb(
            [in] BSTR bstrServerName,
            [in] BOOL bWindowsAuth,
            [in] BSTR bstrUserName,
            [in] BSTR bstrPasswd);
    [id(2), helpstring("method GetAccountingMetadata")]
        HRESULT GetAccountingMetadata(
            [out] BSTR *pbstrMetaData);
    [id(3), helpstring("method ExecuteAccountingQuery")]
        HRESULT ExecuteAccountingQuery(
            [in] BSTR bstrAccountingQuery,
            [in] BSTR bstrStartingDate,
            [in] BSTR bstrEndingDate,
            [out] BSTR *pbstrResult,
            [out] BOOL *pbIsThereMoreData);
    [id(4), helpstring("method GetRawAccountingData")]
        HRESULT GetRawAccountingData(
            [in] BSTR bstrStartingDate,
            [in] BSTR bstrEndingDate,
            [in] BSTR bstrMachineName,
            [out] BSTR *pbstrResult,
            [out] BOOL *pbIsThereMoreData);
    [id(5), helpstring("method GetNextAccountingDataBatch")]
        HRESULT GetNextAccountingDataBatch(
            [out] BSTR *pbstrResult,
            [out] BOOL *pbIsThereMoreData);
    [id(6), helpstring("method DeleteAccountingData")]
        HRESULT DeleteAccountingData(
            [in] BSTR bstrStartingDate,
            [in] BSTR bstrEndingDate,
            [in] BSTR bstrMachineName);
    [id(7), helpstring("method DefragmentDB")]
        HRESULT DefragmentDB();
    [id(8), helpstring("method CancelAccountingQuery")]
```

```
        HRESULT CancelAccountingQuery(
            [in] BOOL flag);
    [id(9), helpstring("method RegisterAccountingClient")]
        HRESULT RegisterAccountingClient(
            [in] BSTR bstrClientId);
    [id(10), helpstring("method DumpAccountingData")]
        HRESULT DumpAccountingData(
            [in] BSTR bstrAccountingData);
    [id(11), helpstring("method GetAccountingClients")]
        HRESULT GetAccountingClients(
            [out] BSTR *pbstrClientIds);
    [id(12), helpstring("method SetAccountingClientStatus")]
        HRESULT SetAccountingClientStatus(
            [in] BSTR bstrClientIds);
    [id(13), helpstring("method CheckAccountingConnection")]
        HRESULT CheckAccountingConnection();
    [id(14), helpstring("method SetClientPermissions")]
        HRESULT SetClientPermissions(
            [in] BSTR bstrClientId,
            [in] BOOL fAddPermissions);
};

[
    object,
    uuid(21546AE8-4DA5-445E-987F-627FEA39C5E8),
    dual,
    helpstring("IWRMConfig Interface"),
    pointer_default(unique)
]
interface IWRMConfig: IDispatch
{
    [id(1), helpstring("method GetConfig")]
        HRESULT GetConfig(
            [out] BSTR *pbstrConfigInfo,
            [in] CONFIGTYPE enumConfigType);
    [id(2), helpstring("method SetConfig")]
        HRESULT SetConfig(
            [in] BSTR bstrConfigInfo,
            [in] CONFIGTYPE enumConfigType);
    [id(3), helpstring("method IsEnabled")]
        HRESULT IsEnabled(
            [out] BOOL *pbEnable,
            [in] CONFIGTYPE enumConfigType);
    [id(4), helpstring("method EnableDisable")]
        HRESULT EnableDisable(
            [in] BOOL bEnableDisable,
            [in] CONFIGTYPE enumConfigType);
    [id(5), helpstring("method GetExclusionList")]
        HRESULT GetExclusionList(
            [out] BSTR *pbstrExclusionList,
            [in] EXCLUSIONLIST_TYPE enumListType);
    [id(6), helpstring("method SetExclusionList")]
        HRESULT SetExclusionList(
            [in] BSTR bstrExclusionList);
    [id(7), helpstring("method WSRMActivate")]
        HRESULT WSRMActivate(
            [in] BOOL bActivate);
    [id(8), helpstring("method IsWSRMActivated")]
        HRESULT IsWSRMActivated(
            [out] BOOL *pbActivated);
    [id(9), helpstring("method RestoreExclusionList")]
        HRESULT RestoreExclusionList();
};

[
    object,
    uuid(F31931A9-832D-481C-9503-887A0E6A79F0),
    dual,
    helpstring("IWRMProtocol Interface"),
    pointer_default(unique)
```

```
]
interface IWRMProtocol: IDispatch
{
    [id(1), helpstring("method GetSupportedClient")]
        HRESULT GetSupportedClient(
            [out] BSTR *pbstrSupportedClients);
};

[
    object,
    uuid(943991a5-b3fe-41fa-9696-7f7b656ee34b),
    dual,
    helpstring("IWRMMachineGroup Interface"),
    pointer_default(unique)
]
interface IWRMMachineGroup: IDispatch
{
    [id(1), helpstring("method CreateMachineGroup")]
        HRESULT CreateMachineGroup(
            [in] BSTR bstrParentMachineGroupId,
            [in] BSTR bstrMachineGroupInfo);
    [id(2), helpstring("method GetMachineGroupInfo")]
        HRESULT GetMachineGroupInfo(
            [in] BSTR bstrMachineGroupId,
            [out] BSTR *pbstrMachineGroupInfo);
    [id(3), helpstring("method ModifyMachineGroup")]
        HRESULT ModifyMachineGroup(
            [in] BSTR bstrMachineGroupId,
            [in] BSTR bstrMachineGroupInfo,
            [in] MACHINE_GROUP_MERGE_OPTIONS enumMGMergeOptions);
    [id(4), helpstring("method DeleteMachineGroup")]
        HRESULT DeleteMachineGroup(
            [in] BSTR bstrMachineGroupId);
    [id(5), helpstring("method RenameMachineGroup")]
        HRESULT RenameMachineGroup(
            [in] BSTR bstrOldMachineGroupName,
            [in] BSTR bstrNewMachineGroupName);
    [id(6), helpstring("method AddMachine")]
        HRESULT AddMachine(
            [in] BSTR bstrParentMachineGroupId,
            [in] BSTR bstrMachineInfo);
    [id(7), helpstring("method GetMachineInfo")]
        HRESULT GetMachineInfo(
            [in] BSTR bstrMachineId,
            [out] BSTR *pbstrMachineInfo);
    [id(8), helpstring("method ModifyMachineInfo")]
        HRESULT ModifyMachineInfo(
            [in] BSTR bstrParentMachineGroupId,
            [in] BSTR bstrMachineId,
            [in] BSTR bstrMachineInfo);
    [id(9), helpstring("method DeleteMachine")]
        HRESULT DeleteMachine(
            [in] BSTR bstrParentMachineGroupId,
            [in] BSTR bstrMachineId,BOOL bRecursive);
};

[
    object,
    uuid(2A3EB639-D134-422d-90D8-AAA1B5216202),
    dual,
    helpstring("IResourceManager2 Interface"),
    pointer_default(unique)
]
interface IResourceManager2 : IDispatch
{
    [id(1), helpstring("method ExportObjects")]
        HRESULT ExportObjects(
            [in] BSTR bstrObjectIds,
            [in] OBJECT_TYPE enumObjectType,
            [out] BSTR *pbstrObjectXml);
```

```
        [id(2), helpstring("method GetImportConflicts")]
            HRESULT GetImportConflicts(
                [in] BSTR bstrPMCXml,
                [in] BSTR bstrPolicyXml,
                [in] BSTR bstrCalendarXml,
                [in] BSTR bstrConditionalXml,
                [in] BSTR bstrMachineGroupXml,
                [in] BSTR bstrConfigurationXmls,
                [out] BSTR *pbstrConflictingObjects);
        [id(3), helpstring("method ImportXml")]
            HRESULT ImportXml(
                [in] BSTR bstrPMCXml,
                [in] BSTR bstrPolicyXml,
                [in] BSTR bstrCalendarXml,
                [in] BSTR bstrConditionalXml,
                [in] BSTR bstrMachineGroupXml,
                [in] BSTR bstrConfigurationXmls,
                [in] IMPORT_TYPE enumImportType);
        [id(4), helpstring("method ExportXml")]
            HRESULT ExportXml(
                [out] BSTR *pbstrPMCXml,
                [out] BSTR *pbstrPolicyXml,
                [out] BSTR *pbstrCalendarXml,
                [out] BSTR *pbstrConditionalXml,
                [out] BSTR *pbstrMachineGroupXml,
                [out] BSTR *pbstrConfigurationXmls);
    };

    [
        object,
        uuid(FC910418-55CA-45ef-B264-83D4CE7D30E0),
        dual,
        helpstring("IWRMRemoteSessionMgmt Interface"),
        pointer_default(unique)
    ]
    interface IWRMRemoteSessionMgmt : IDispatch
    {
        [id(1), helpstring("method GetRemoteUserCategories")]
            HRESULT GetRemoteUserCategories(
                [out] BSTR *pbstrRemoteUserCategoriesInfo);
        [id(2), helpstring("method SetRemoteUserCategories")]
            HRESULT SetRemoteUserCategories(
                [in] BSTR bstrRemoteUserCategoriesInfo);
        [id(3), helpstring("method RefreshRemoteSessionWeights")]
            HRESULT RefreshRemoteSessionWeights(
                [in] BSTR bstrTaregetUserSessions,
                [in] BOOL bUpdateAll);
    }
```

7 Appendix B: 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.

- Windows Vista operating system with Service Pack 1 (SP1)
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 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 1.8: Aside from the HRESULT values explicitly defined in this specification, Windows uses return values as specified in [MS-ERREF].

<2> Section 1.8: Windows WSRM provides both a graphical user interface and a command line interface.

<3> Section 2.2.1.3: Windows exchanges an extra digit between date-month: date-month-day, separated by ":", which actually denotes the days of the week [0-6], 0 for Sunday, 1 for Monday, and ending with 6 for Saturday.

<4> Section 2.2.2.8: This value was added with support for Terminal Services in Windows Vista operating system and Windows Server 2008.

<5> Section 2.2.2.8: Windows Server 2008, Windows 7, Windows Server 2008 R2 operating system, Windows 8, and Windows Server 2012: This option was added for computers on which Terminal Services is running.

<6> Section 2.2.2.9: This value corresponds to WSRM servers implemented on Windows Server 2008 and Windows Vista.

<7> Section 2.2.2.9: This value corresponds to WSRM servers implemented on Windows 7 and Windows Server 2008 R2 without **DFSS**.

<8> Section 2.2.2.9: This value corresponds to WSRM servers implemented on Windows 7 and Windows Server 2008 R2 with DFSS.

<9> Section 2.2.3.2: In the Windows implementation, several **svchost.exe** processes are included in the default user-defined exclusion list. Important operating system services run inside the **svchost.exe** processes. To ensure that these processes have access to their required CPU bandwidth, they are excluded from management.

<10> Section 2.2.5.9: In Windows Vista and Windows Server 2008, this function is not supported.

<11> Section 2.2.5.9: In Windows Vista and Windows Server 2008, this function is not supported.

<12> Section 2.2.5.11: Windows uses root element as Calendars instead of CalendarsCollection in CalendarsCollection schema.

<13> Section 2.2.5.21: For Windows, the name is the same as the value of the **RefName** attribute of the <ProcessMatchingCriteria> element.

<14> Section 2.2.5.22: Windows uses root element as Policy instead of PolicyCollection in Policy Collection schema.

<15> Section 2.2.5.25: Windows uses root element as ProcessMatchingCriteria instead of ProcessMatchingCriteriaCollection in ProcessMatchingCriteriaCollection schema.

<16> Section 3.2.1.1: This structure was added with support for Terminal Services in Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, and Windows Server 2012.

<17> Section 3.2.1.1: This structure was added with support for Terminal Services in Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, and Windows Server 2012.

<18> Section 3.2.1.2: The default location of the accounting database for Windows is:

```
C:\windows\system32\Windows System Resource Manager\JetDb\wsrm.edb
```

<19> Section 3.2.3: WSRM in Windows ships with following hard-coded built-in policies:

Name	Description
Equal per process	When the Equal_Per_Process resource allocation policy is managing the system, each running process is given an equal percentage of CPU.
Equal per user	When the Equal_Per_User resource allocation policy is managing the system, processes are grouped according to the user account that is running them and each of these process groups is given an equal percentage of CPU.
Equal per session	When the Equal_Per_Session resource allocation policy is managing the system, resources are allocated on an equal basis for each session connected to the system. This policy is for use with terminal servers.
Equal per IIS application pool	When the Equal_Per_IISAppPool resource allocation policy is managing the system, each running IIS application pool is given an equal percentage of CPU, and applications that are not in an IIS application pool can use only resources that are not being consumed by IIS application pools.
Weighted Remote Sessions	When the Weighted_Remote_Sessions resource allocation policy is managing the system, the processes are grouped according to the priority assigned with the user account.

<20> Section 3.2.4: The following table shows support for interfaces of the WSRM protocol in client and server implementations by Windows version. See [MSDN-WSRM1] for more information.

Interface	Windows Version			
	Vista	Server 2008	7	Server 2008 R2
IResourceManager	X	X	X	X
IResourceManager2			X	X
IWRMAccounting	X	X	X	X

Interface	Windows Version			
IWRMCalendar	X	X	X	X
IWRMConfig	X	X	X	X
IWRMMachineGroup			X	X
IWRMPolicy	X	X	X	X
IWRMProtocol	X	X	X	X
IWRMRemoteSessionMgmt			X	X
IWRMResourceGroup	X	X	X	X

<21> Section 3.2.4: This interface is not supported in Windows Vista and Windows Server 2008.

<22> Section 3.2.4: This interface is not supported in Windows Vista and Windows Server 2008.

<23> Section 3.2.4: This interface is not supported in Windows Vista and Windows Server 2008.

<24> Section 3.2.4.1: This method was added with support for Terminal Services in Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, and Windows Server 2012.

<25> Section 3.2.4.1: This method was added with support for Terminal Services in Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, and Windows Server 2012.

<26> Section 3.2.4.1: This method was added with support for Terminal Services in Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, and Windows Server 2012.

<27> Section 3.2.4.1.3: Windows returns this value if the XML data is corrupt.

<28> Section 3.2.4.1.4: Windows uses this method for performing automatic backups of the WSRM system.

<29> Section 3.2.4.1.4: Windows returns this value if the XML data is corrupt.

<30> Section 3.2.4.1.5: If *enumRestore* is **RESTORE_LAST_GOOD_STATE**, the XML files from the backup folder "System32/Windows System Resource Manager/backup" are read.

<31> Section 3.2.4.1.5: Windows returns this value if the XML data is corrupt.

<32> Section 3.2.4.1.8: Windows returns the path for the system folder, such as "c:\windows\system32".

<33> Section 3.2.4.1.9: On Windows, if an error value is returned, it is the return value of the Windows SDK system information function **GetComputerName**, converted to an HRESULT.

<34> Section 3.2.4.2: This interface is not supported in Windows Vista and Windows Server 2008.

<35> Section 3.2.4.2.1: Windows returns 0x80004005 (Unspecified error) when *enumObjectType* set to OBJECT_SCHEDULE and *pbstrObjectXml* contains elements other than <Calendar> or <Schedule>.

<36> Section 3.2.4.2.2: Windows returns this value if the XML data is corrupt.

<37> Section 3.2.4.2.3: In Windows if this parameter is used with **OVERRIDE_EXISTING_IMPORT** then the error 0x80004005 can be returned. In some cases, this is a false error and data is actually imported successfully.

<38> Section 3.2.4.2.3: Windows can return this error if used with **OVERRIDE_EXISTING_IMPORT**.

<39> Section 3.2.4.2.3: Windows returns this value if the XML data is corrupt.

<40> Section 3.2.4.3.1: Windows uses a pre-built database and always returns the WRM_ERR_ACCOUNTING_FAILED error.

<41> Section 3.2.4.3.3: Windows returns this error when an accounting query is executed after the connection to the **Windows internal database** server has been lost.

<42> Section 3.2.4.3.3: Windows returns this error when the management service is busy executing the accounting query from another client.

<43> Section 3.2.4.3.4: Windows supports the Windows Internal Database, a built-in relational database component for use by other Windows components.

<44> Section 3.2.4.3.4: Windows returns this error when the management service tries to acquire the accounting semaphore before executing the accounting query.

<45> Section 3.2.4.3.5: Windows gets data in batches of 1,024 records.

<46> Section 3.2.4.3.5: Windows returns this error when the management service tries to acquire the accounting semaphore before executing the accounting query.

<47> Section 3.2.4.3.6: Windows Server 2008, Windows Server 2008 R2, and Windows Server 2012 support the Windows Internal Database, a built-in relational database component for use by other Windows components.

<48> Section 3.2.4.3.6: Windows returns S_OK when the *bstrStartingDate* or *bstrEndingDate* parameters are invalid and no data is deleted.

<49> Section 3.2.4.3.8: Windows returns this error when this method is called before accounting has been initialized or after accounting has been uninitialized. Because accounting executes on a separate thread, an accounting client can call CancelAccountingQuery while a command for shutdown has already been initiated.

<50> Section 3.2.4.3.9: Windows Server 2008, Windows Server 2008 R2, and Windows Server 2012 support the Windows Internal Database, a built-in relational database component for use by other Windows components.

<51> Section 3.2.4.3.9: To reproduce on a Windows implementation, perform the following steps:

1. Use the WSRM management service running on a server (server A) to set up another server (server B) to log accounting data remotely by calling RegisterAccountingClient of WSRM management service running on server B.
2. That server A is now an accounting client.
3. Have another server (server C) call RegisterAccountingClient of WSRM management service on server A.
4. This error is returned.

<52> Section 3.2.4.3.9: To reproduce on a Windows implementation, perform the following steps:

1. Stop the WSRM management service on the accounting server machine:

```
net stop wsrm
```

2. Go to "%windir%\System32\Windows System Resource Manager" and corrupt the database folder by deleting, moving, or renaming some files in the database directory.

3. Start the WSRM management service on the accounting server machine, which now will not be able to initialize its accounting because of the corrupt database:

```
net start wsrm
```

4. Have an accounting client call RegisterAccountingClient with a client machine name.

5. WSRM is unable to connect to the Windows Internal Database, and this error is returned.

<53> Section 3.2.4.3.12: Windows returns an error if the SetAccountingClientStatus method is called to disable accounting functionality of an accounting client.

<54> Section 3.2.4.3.13: On Windows, this error is reproduced by the following procedure:

1. Register an accounting client with a server.

2. Set the accounting client status to disable.

3. Call CheckAccountingConnection.

4. The error is returned.

<55> Section 3.2.4.3.13: Windows calls the CheckAccountingConnection method prior to calling DumpAccountingData.

<56> Section 3.2.4.4.2: A value of FALSE causes the Windows WSRM management service to crash.

<57> Section 3.2.4.4.2: Windows returns this value if the XML data is corrupt.

<58> Section 3.2.4.4.2: Windows returns this error if the number of calendar events that already exist in the system is equal to 512.

<59> Section 3.2.4.4.3: A value of FALSE causes the Windows WSRM management service to crash.

<60> Section 3.2.4.4.3: Windows returns this value if the XML data is corrupt.

<61> Section 3.2.4.4.3: Windows returns a description of this error as an invalid day of the year.

<62> Section 3.2.4.4.4: A value of FALSE causes the Windows WSRM management service to crash.

<63> Section 3.2.4.4.5: Windows returns this error if the calendar name is longer than 200 characters.

<64> Section 3.2.4.4.8: Windows returns this value if the XML data is corrupt.

<65> Section 3.2.4.4.8: Windows returns this error if the number of schedules that already exist in the system is equal to 512.

<66> Section 3.2.4.4.9: A value of FALSE causes the Windows WSRM management service to crash.

<67> Section 3.2.4.4.9: Windows returns this value if the XML data is corrupt.

<68> Section 3.2.4.4.11: Windows returns this error if the schedule name is longer than 200 characters.

<69> Section 3.2.4.4.12: A value of FALSE causes the Windows WSRM management service to crash.

<70> Section 3.2.4.4.13: A value of FALSE causes the Windows WSRM management service to crash.

<71> Section 3.2.4.5: Windows Server 2008 supports the Windows Internal Database, a built-in relational database component for use by other Windows components.

<72> Section 3.2.4.5.2: Windows returns this error if the configuration value is longer than 260 characters.

<73> Section 3.2.4.5.2: Windows returns this value if the XML data is corrupt.

<74> Section 3.2.4.5.2: In Windows Server 2008, Windows Server 2008 R2, and Windows Server 2012, the ERROR_ACCESS_DENIED error can occur even if SetClientPermissions (section 3.2.4.3.14) is called prior to SetConfig. The following additional action is required: Before calling SetConfig, add the machine account of the WSRM server acting as the remote accounting client to the "Administrators" group of the WSRM server acting as the remote accounting server.

<75> Section 3.2.4.5.4: Windows returns this error if EnableDisable method is called to disable calendar.

<76> Section 3.2.4.5.4: Windows implementations return this error in the following scenario:

1. Create a new registry entry "AccountingEnabled" under the path "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\WSRM".

2. Set the value of "AccountingEnabled" to 0x00000001 (DWORD).

3. Call the EnableDisable method to disable the WSRM configuration for accounting.

<77> Section 3.2.4.5.4: Windows implementations return this error when the EnableDisable method is called to manage the WSRM configuration for Accounting before the management service has started or completely initialized the Accounting thread.

<78> Section 3.2.4.5.4: Windows returns this error in the following scenario, in which a remote database server is used:

1. Set a remote server to act as the database server of the current server.

2. On the current server, use the EnableDisable method to enable the accounting configuration.

3. Connect to the remote database server and stop the WSRM management service.

4. On the current server, use the EnableDisable method to disable the accounting configuration.

<79> Section 3.2.4.5.5: Windows returns this error if a process name in the registry is longer than 260 characters.

<80> Section 3.2.4.5.5: Windows returns this error if the number of excluded processes in the registry is greater than 128.

<81> Section 3.2.4.5.6: Windows returns this error if a specified process name is longer than 260 characters.

<82> Section 3.2.4.5.6: Windows returns this error if the number of specified processes is greater than 128.

<83> Section 3.2.4.5.7: In Windows, the registry is located at "HKLM\SYSTEM\CurrentControlSet\Services\WSRM\Parameters\WSRMActivate". This registry value is set to "true" when WSRM is in active management state; otherwise, the registry key value is set to "false". The WSRM management service looks for this key while starting the service. The value of this key determines the state of the WSRM Protocol at startup.

<84> Section 3.2.4.6: This interface is not supported in Windows Vista and Windows Server 2008.

<85> Section 3.2.4.6.1: Windows does not validate the machine name or its presence in a domain while validating machine details.

<86> Section 3.2.4.6.1: Windows returns this value if the XML data is corrupt.

<87> Section 3.2.4.6.1: Windows returns this error if the number of machine groups is greater than 128.

<88> Section 3.2.4.6.1: Windows returns this error if the number of machines in a machine group is greater than 128.

<89> Section 3.2.4.6.3: Windows does not register the enumMGMergeOptions value and always responds as if it is OVERWRITE_MG_MERGE_OPTION.

<90> Section 3.2.4.6.3: Windows does not return E_INVALIDARG for an invalid enumMGMergeOptions value and always behaves as if it is OVERWRITE_MG_MERGE_OPTION.

<91> Section 3.2.4.6.3: Windows returns this value if the XML data is corrupt.

<92> Section 3.2.4.6.3: Windows returns this error if the number of machine groups is greater than 128.

<93> Section 3.2.4.6.3: Windows returns this error if the number of machines in a machine group is greater than 128.

<94> Section 3.2.4.6.3: Windows only checks the current node and its children's group name for duplication while modifying the machine group. This error can be reproduced by making the following sequence of calls in a fresh configuration setup.

1. Import the machine group XML given in the example in section 4.2.15, using the **ImportXml** method.

2. Call **ModifyMachineGroup** with *bstrMachineGroupId* as "Communication Servers", *enumMGMergeOptions* as "OVERWRITE_MG_MERGE_OPTION", and *bstrMachineGroupInfo* as "<MachineGroup Name="Backup Servers" Description=""><Machine Name="BS1" Description="" /> </MachineGroup>".

3. **ModifyMachineGroup** fails with WRM_ERR_MACHINEGROUP_ALREADY_EXISTS because the name of the machine group "Communication Servers" is being modified to "Backup Servers", which already exists in the configuration.

<95> Section 3.2.4.6.6: Windows does not validate machine name or its presence in domain while validating machine details.

<96> Section 3.2.4.6.6: Windows returns this value if the XML data is corrupt.

<97> Section 3.2.4.6.6: Windows returns this error if the number of machines in a machine group is greater than 128.

<98> Section 3.2.4.6.7: Windows returns WRM_ERR_MACHINEGROUPID_INVALID if the machine ID is invalid.

<99> Section 3.2.4.6.8: On Windows, a ModifyMachineInfo call stops the WSRM service if called with a non-existing parent machine group ID and a **bstrMachineId** which exists in some machine group. If this function is called with a non-existing parent machine group ID and a **bstrMachineId** which does not exist in any of the machine groups, then the server returns WRM_ERR_MACHINEID_INVALID.

<100> Section 3.2.4.6.8: Windows returns this value if the XML data is corrupt.

<101> Section 3.2.4.6.8: Windows returns WRM_ERR_MACHINEID_INVALID instead of WRM_ERR_MACHINEGROUPID_INVALID if the machine group ID is invalid.

<102> Section 3.2.4.6.8: Windows returns a WRM_ERR_MACHINEID_INVALID error if some machine with the new machine name already exists.

<103> Section 3.2.4.6.9: Windows does not recursively delete all instances of machines even if the value of **bRecursive** is set to TRUE in DeleteMachine call.

<104> Section 3.2.4.6.9: Windows returns WRM_ERR_MACHINEGROUPID_INVALID instead of WRM_ERR_MACHINEID_INVALID for an invalid machine ID.

<105> Section 3.2.4.7.2: Windows returns this value if the XML data is corrupt.

<106> Section 3.2.4.7.2: Windows returns this error if the number of RAPs is greater than 128.

<107> Section 3.2.4.7.3: Windows returns this value if the XML data is corrupt.

<108> Section 3.2.4.7.5: Windows returns this error if the number of characters in a RAPs name is greater than 200.

<109> Section 3.2.4.7.5: Windows returns this value if the XML data is corrupt.

<110> Section 3.2.4.7.9: Windows Server 2008, Windows Vista, Windows 7, and Windows Server 2008 R2 never return this error because a default resource allocation policy (RAP) is always set. EqualPerProcess is the default RAP.

<111> Section 3.2.4.7.14: Windows only accepts "GlobalCP" as the conditional policy name.

<112> Section 3.2.4.7.15: On Windows Server 2008, a value of NULL causes the Windows WSRM management service to crash.

<113> Section 3.2.4.9: This interface is not supported in Windows Vista and Windows Server 2008.

<114> Section 3.2.4.9.2: Windows returns this value if the XML data is corrupt.

<115> Section 3.2.4.10.2: Windows returns this value if the XML data is corrupt.

<116> Section 3.2.4.10.2: Windows returns this error if the resource group name is longer than 200 characters.

<117> Section 3.2.4.10.2: Windows returns this error if the length of the command line is greater than 4160 characters.

<118> Section 3.2.4.10.2: Windows returns this error if the user name or group value is longer than 1040 characters.

<119> Section 3.2.4.10.2: **WSRM** in Windows Server 2008 and Windows Server 2008 R2 has the following hard-coded built-in resource groups.

Name	Description
Residual	The residual process matching criterion matches all processes that are not included in the exclusion list and do not match another process matching criterion.
IISAppPool	Matches all IIS application pool worker processes.

<120> Section 3.2.4.10.3: Windows returns this error if the total number of resource groups exceeds 128.

<121> Section 3.2.4.10.5: Windows returns this error if the resource group name is longer than 200 characters.

8 Change Tracking

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

9 Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Y