April 21, 2006

Candidate Standard 5106.2-2006

The Printer Working Group

Web-based Imaging Management Service V1.0 Abstract Protocol

Status: Approved

Abstract: This specification defines the abstract Web-based Imaging Management Service (WIMS) protocol. This specification defines five operations initiated by a WIMS Agent (embedded in services or devices), largely in support of Schedule-oriented remote management: RegisterForManagement (Agent allows management by an identified WIMS Manager); and UnregisterForManagement (cancel Agent association with a given WIMS Manager); GetSchedule (request a Schedule of planned actions); SendReports (send normal operational message); and SendAlerts (send warning or error exception message). This specification also defines four operations initiated by a WIMS Manager to support more conventional local management: BeginManagement (Manager requests ability to manage an identified Agent); EndManagement (Manager cancels association with Agent); SetSchedule (send a Schedule of planned actions with their timetables); ExecuteAction (execute the single identified action). This specification also defines sets of monitoring, management and administration actions that can be included within a Schedule. Transport bindings for the WIMS protocol are identified in the appendix.

This document is a PWG Candidate Standard. For a definition of a "PWG Candidate Standard", see: ftp://ftp.pwg.org/pub/pwg/general/pwg-process20.pdf

This document is available electronically at:
ftp://ftp.pwg.org/pub/pwg/candidates/cs-wims10-20060421.pdf

Title: Web-based Imaging Management Service Version 1.0

About the IEEE-ISTO:

The IEEE-ISTO is a not-for-profit corporation offering industry groups an innovative and flexible operational forum and support services. The IEEE Industry Standards and Technology Organization member organizations include printer manufacturers, print server developers, operating system providers, network operating systems providers, network connectivity vendors, and print management application developers. The IEEE-ISTO provides a forum not only to develop standards, but also to facilitate activities that support the implementation and acceptance of standards in the marketplace. The organization is affiliated with the IEEE (http://www.ieee.org/) and the IEEE Standards Association (http://standards.ieee.org/).

For additional information regarding the IEEE-ISTO and its industry programs visit:

http://www.ieee-isto.org.

About the Printer Working Group:

The Printer Working Group (or PWG) is a Program of the IEEE-ISTO. All references to the PWG in this document implicitly mean “The Printer Working Group, a Program of the IEEE ISTO.” The PWG is chartered to make printers and the applications and operating systems supporting them work together better. In order to meet this objective, the PWG will document the results of their work as open standards that define print related protocols, interfaces, data models, procedures and conventions. Printer manufacturers and vendors of printer related software would benefit from the interoperability provided by voluntary conformance to these standards.

In general, a PWG standard is a specification that is stable, well understood, and is technically competent, has multiple, independent and interoperable implementations with substantial operational experience, and enjoys significant public support.

Contact information:

The Printer Working Group
c/o The IEEE Industry Standards and Technology Organization
445 Hoes Lane
Piscataway, NJ 08854
USA

WIMS Web Page:

http://www.pwg.org/wims

WIMS Mailing List:

wims@pwg.org
Instructions for subscribing to the WIMS mailing list can be found at the following link:
http://www.pwg.org/mailhelp.html

Those interested in this specification are encouraged to join the WIMS Mailing List and to participate in any discussions clarifications or review of this specification. Not that, to reduce spam, the mailing list rejects mail from non-subscriber; you must subscribe to the mailing list to be able to send a question or comment to the mailing list.

Contributors

Lee Farrell	Canon
Richard Landau	Dell
Harry Lewis	IBM
Ira McDonald	High North
Jerry Thrasher	Lexmark
William A Wagner	Technical Interface Consulting
Peter Zehler	Xerox

Table of Contents

Table of Figures

1 Introduction

Providing ready access to printing facilities has been one of the prime driving forces in the development of local area networks. As networked printing became more prevalent, printers themselves became networked devices rather than peripherals to networked computers. The need to manage these networked imaging devices prompted network management capabilities such as SNMP, previously intended primarily for management of the network infrastructure and terminals, to be extended to imaging devices, and fostered the generation of the Printer MIB.

As the popularity of digital imaging has grown and other imaging devices such as copiers and facsimile machines have been networked, imaging equipment has been consolidated into networked multifunction imaging systems. These are variously known as Multifunction Peripherals or Multifunction Printers (MFPs), Multifunction Devices (MFDs) and, at the low end, All-in-Ones. Because Multifunction Imaging Systems typically deal with tangible hard copy and require consumables, servicing and maintenance support is a critical feature. Further, the complexity of these systems and the critical services they provide to an organization have prompted the creation of specialized internal and external maintenance organizations supporting “fleets” of imaging systems. Many organizations lease imaging equipment from MFD vendors or third party companies, delegating the responsibility for ensuring un-interrupted service to external companies. These factors have increased the requirement for a flexible capability allowing remote monitoring and management and providing readily accessible information on the status, configuration and utilization of imaging systems.

The Web-based Imaging Management System (WIMS) protocol is designed to support both fleet management (across the Internet by outside service providers) and enterprise management (within an administrative domain by in-house staff) of image processing devices (printers, scanners, copiers, etc.) and image processing services (print spoolers, etc.) WIMS defines three primary aspects:

- The Agent Interface, including the Operations necessary to allow access by a Manager, to solicit a Schedule of management actions, to report on requested elements and to provide alert information for identified events.
- The Management Interface including the Operations by which the required management information is requested
- The monitoring, management and administrative Actions requested of the managed entity in the Schedule or the Management Interface operations.

Note that WIMS specifically does NOT address equipment or service Discovery, although existing discovery protocols could be used in conjunction with WIMS. This reflects the basic “fleet management” function of WIMS, particularly by external agencies. Providing a third party (or in some cases, even an internal Imaging System maintenance organization) with the ability to search a network to discover imaging systems would represent an unacceptable security vulnerability. Rather, the premise of WIMS is that determination of what systems are to be managed by what manager, and indeed, what parameters the manager has access to are determined at the imaging system site independently of WIMS. That is, WIMS allows the manager to manage the imaging system fleet, but neither to determine what constitutes the fleet nor to determine the maximum degree of management access.

Further, WIMS is set up so that the primary communication path is always initiated by the managed entity or, more often, by a WIMS proxy for the managed entity. This “reverse” communication path, compared to the pre-emptive manager-oriented approach of traditional management protocols, predicated the concept of an “action Schedule”. Although a secondary, optional capability is provided by which a manager may contact a managed imaging system and request a specific action or response, in the primary management sequence a managed entity contacts the manager and requests a Schedule of actions to be implemented either on a time or condition basis. The managed entity then initiates a connection to the manager at the times or under the conditions identified in the Schedule. WIMS is design so that at no time would an external manager need to initiate a connection into network on which the Imaging Systems reside.

This document outlines representative scenarios that WIMS addresses, defines the requirements of WIMS and provides the detailed technical specification for the WIMS Interfaces.

WIMS is XML based to facilitate Web based operation, and to be compatible with Web Services, WIMS is intended, but not restricted to use a SOAP binding over HTTP or SMTP. Other transports may be used. Although WIMS defines the protocol by which monitoring, management and administrative actions are requested and the results of such actions are reported to the manager, it does require that there exist XML representations of the parameters to be configured or accessed. The Printer Working Group has other on-going efforts to provide suitable schema including management parameters in the Printer MIB and in IPP as well as new Multifunction Device oriented parameters. The XML encoding of management parameters for WIMS is defined in a set of W3C XML Schema (*.xsd) documents which extend the PWG Semantic Model. These XML Schema are contained in:

http://www.pwg.org/schemas/wims/1.0

Note that this directory is intended for development reference and is not to be actively referenced by actual products. These schema are not limited to WIMS but may be used in any application requiring XML coding of Imaging System parameters.

2 Terminology

This section defines terminology used throughout this document.

2.1 Conformance Terminology

Capitalized terms, such as MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, MAY, and OPTIONAL, have special meaning relating to conformance as defined in RFC 2119 [rfc2119].

2.2 Other Terminology

This document uses the same terminology as [rfc2911], such as “attribute”, “attribute value”, “keyword”, “operation”, “request”, “response”, and “support”, with the same meaning. In addition, the following terms are defined for use in this document:

Element – An abstract construct that defines an object, and the components of an object. In this document *element* is also used to describe an attribute of an object.

Interface - An interface is a collection of methods that are exposed by the service. An example of an interface is the *WIMS Agent Interface*.

Legacy Managed Entity – An imaging service or device which does not support WIMS but has an associated management agent which does allow access to management parameters by some other protocol, such as SNMP. A *Legacy Managed Entity* may be managed by a *WIMS Manager* by the use of a *Management Proxy*.

Managed Entity – A Legacy or WIMS managed entity.

Method – A *method* is an operation in an interface.

Object – An object is a self-contained entity that contains data and methods to manipulate the data. Example objects are Printer, Job, and Document.

Protocol – A protocol is an agreed-upon method for transmitting information between two devices. The protocol determines the communication method. An example of a protocol is TCP/IP.

Service - A service provides some desired functions and contains one or more interfaces used for communication. A Print Service is an example of a service.

WIMS Agent – An application in an imaging device or service, or in a WIMS proxy that implements the WIMS Agent interface.

WIMS Imaging System – The collection of WIMS Managed Devices, Managed Services and other objects supporting the processing of an imaging job.

WIMS Managed Entity - An imaging service or device which has an associated WIMS Agent.

WIMS Manager – An application implementing the WIMS Manager Interface

WIMS Proxy – An intermediary service that supports a WIMS Agent Interface and (optionally) a WIMS Manager Interface or some other management interface (such as an SNMP application). The WIMS Proxy provides WIMS functionality to one or more Legacy Managed Entities by acting as a local extension of the WIMS Manager, and communicating with the Legacy Managed Entities using a method supported by these entities. WIMS Proxies may also be used to more effectively control internet traffic between the WIMS Agents and the WIMS Manager.

3 Requirements

3.1 Rationale for WIMS Protocol

The ISO, IETF, and PWG standards for the printing industry define:

a) A rationale for an abstract model of printing (to support alternate encodings and protocols) in section 3 of the IETF IPP Rationale [RFC2568] which led to the later development of the PWG Semantic Model/1.0 [PWG5105.1].

b) A set of design goals for status monitoring in a printing protocol in section 3.1.3 'Viewing the status and capabilities of a printer' (for End User), section 3.2.1 'Alerting' (for Operator), and section 3.3 'Administrator' (the bullet requirement to 'administrate billing or other charge-back mechanisms') of the IETF IPP Design Goals [RFC2567].

c) An abstract model of a Print Service in section 2.1 of IETF IPP/1.1 [RFC2911].

d) A set of multifunction Service types for Imaging Systems in the 'JmJobServiceTypesTC' textual convention in section 4 of the IETF Job Monitoring MIB [RFC2707].

e) An abstract model of a multifunction Job in section 2 of the IETF Job Monitoring MIB [RFC2707].

f) An abstract model of a Print Job in section 2.2 of IETF IPP/1.1 [RFC2911].

g) A set of abstract Print Job counter attributes in section 4.3.18 of IETF IPP/1.1 [RFC2911], section 3.8 of PWG IPP Production Printing Attributes [PWG5100.3], section 5.1 of PWG IPP Job Extensions. [PWG5100.7], and section 4 of the IETF Job Monitoring MIB [RFC2707].

h) An abstract model of a Print Device in section 2.2 of the IETF Printer MIB v2 [RFC3805].

i) A set of abstract Print Device counter attributes in section 6 of the IETF Printer MIB v2 [RFC3805].

j) An abstract model of a printing Resource in section 6.3.7 and section 9.8 of ISO Document Printing Application (DPA) [ISO10175].

Over the past decade, network printers have evolved into multifunction Imaging Systems. To support monitoring, maintenance, and administration of these Imaging Systems, this document defines:

1) New abstract Agent, Device, Manager, Resource, Service, Subunit, and System objects with Status and Description element groups, as a framework extension to the PWG Semantic Model/1.0 [PWG5105.1].

2) New abstract Report and Schedule objects to support the delayed execution of monitoring, management, and administration actions, as a framework extension to the PWG Semantic Model/1.0 [PWG5105.1].

3) New abstract Alert and Subscription objects to support notifications for events from monitored objects, as a framework extension to the PWG Semantic Model/1.0 [PWG5105.1].

4) Two sets of abstract operations i.e., Agent Interface and Manager Interface to support monitoring, management, and administration, as a framework extension to the PWG Semantic Model/1.0 [PWG5105.1].

5) A set of conformance requirements for implementation of these new abstract objects, operations, and actions.

3.2 Use Models for WIMS Protocol

3.2.1 Service Providers - Monitoring and Billing

Outside service providers may lease and maintain imaging software and imaging equipment in remote customer enterprise networks in different administrative domains.

> Note: Typically monitoring proxies within customer enterprise networks are required for scalability of this use model. However, the deployment of monitoring proxies and of security credentials is outside the scope of this document.

1. To support basic usage billing, outside service providers may periodically read System counters from imaging systems (e.g., once a month).
2. To support detailed usage billing, outside service providers may periodically read Service and Subunit counters from imaging systems (e.g., once a month).
3. To support reordering of supplies, outside service providers periodically may read System and Subunit counters from imaging systems (e.g., every week).
4. To support preventive maintenance, outside service providers may periodically read System counters from imaging systems (e.g., every week) and may subscribe to System, Service, and Subunit events.
5. To support downtime guarantees, outside service providers may read System, Service, and Subunit counters from imaging systems, especially for configuration changes, critical alerts, and allocation errors (e.g., every 15 minutes).

3.2.2 System Administrators - Network Management

Network System administrators configure and manage Services and Subunits on imaging systems in local enterprise networks.

1. To support basic configuration, network system administrators may periodically read System elements from imaging systems for configuration checkpoints (e.g., every month).
2. To support detailed configuration, network system administrators may periodically read Service, Device, Subunit, and Resource elements from imaging systems for configuration checkpoints (e.g., every month).
3. To support configuration updates, network system administrators may write System, Service, Device, Subunit, and Resource elements on imaging systems (e.g., as needed).
4. To support usage and access policies, network system administrators may change enable and disable System, Service, Device, and Subunit elements on imaging systems (e.g., as needed) and may subscribe to System, Service, Device, and Subunit events.
5. To support preventive maintenance, network system administrators may read System counters from imaging systems (e.g., every week).
6. To support emergency maintenance, network system administrators may read System, Service, and Subunit counters from imaging systems, especially for configuration changes, critical alerts, and allocation errors (e.g., every 15 minutes) and may subscribe to System, Service, and Subunit events.

3.2.3 Network Applications - Accounting

Network accounting applications monitor Services and Jobs on imaging systems in local enterprise networks.

1. To support basic accounting, a network accounting application may periodically read System counters from imaging systems (e.g., every month).
2. To support detailed accounting, a network accounting application may periodically read Service counters from imaging systems (e.g., every week).
3. To support user accounting, a network accounting application may periodically read Service, Job, and Document counters from imaging systems (e.g., every minute) and may subscribe to Service, Job, and Document events.

3.3 Design Requirements for WIMS Protocol

1. The WIMS Protocol design MUST follow the naming conventions and element structuring requirements defined in the PWG Semantic Model/1.0 [PWG-5105.1], including group and element containment, counter datatype, and counter precision requirements.
2. The WIMS Protocol design MUST support mappings to multiple transport protocols (e.g., TCP or UDP.) See sections 3.2.1 and 3.2.
3. The WIMS Protocol design MUST support mappings to multiple session protocols (e.g., HTTP, SMTP, or BEEP.) See sections 3.2.1 and 3.2.
4. The WIMS Protocol design MUST support mappings to multiple security protocols (e.g., TLS or S/MIME.) See sections 3.2.1 and 3.2.
5. The WIMS Protocol design MUST support mappings to multiple management protocols e.g., OASIS WSDM or IETF SNMP) and multiple data modeling languages (e.g., XML Schema or SNMP SMIv2. See section 3.2.
6. The WIMS Protocol design MUST support Schedule objects corresponding to the schedTable element defined in IETF Schedule MIB [RFC3231]. See all use models in section 3.
7. The WIMS Protocol design MUST support Report objects for reporting results and status for delayed actions specified in Schedule objects. See all use models in section 3.
8. The WIMS Protocol design MUST support Subscription objects corresponding to the Subscription object defined in IETF IPP Event Notifications [RFC3995]. See all use models in section 3.
9. The WIMS Protocol design MUST support Alert objects corresponding to the Notification object defined in IETF IPP Event Notifications [RFC3995] and the printerV2Alert SNMP trap defined in IETF Printer MIB v2 [RFC3805]. See all use models in section 3.
10. The WIMS Protocol design MUST support Agent and Manager objects corresponding to management agent and management station endpoints in the WIMS Protocol and other network management protocols. See all use models in section 3.
11. The WIMS Protocol design MUST support System objects corresponding to the System group defined in IETF Host Resources MIB v2 [RFC2790]. See all use models in section 3.
12. The WIMS Protocol design MUST support Service objects corresponding to the Printer object defined in IETF IPP/1.1 [RFC2911]. See all use models in section 3.
13. The WIMS Protocol design MUST support Device objects corresponding to the Printer device defined in IETF Printer MIB v2 [RFC3805]. See all use models in section 3.
14. The WIMS Protocol design MUST support Subunit objects corresponding to the Printer device subunits defined in IETF Printer MIB v2 [RFC3805]. See all use models in section 3.
15. The WIMS Protocol design SHOULD support Resource objects corresponding to the Resource object defined in ISO Document Printing Application [ISO10175]. See section 3.2.

16. The WIMS Protocol design MUST support explicit counter persistence corresponding to 'prtMarkerLifeCount' and 'prtMarkerPowerOnCount' in IETF Printer MIB v2 [RFC3805]. See section 3.2.
17. The WIMS Protocol design MUST support both standard and vendor extensions that define new interfaces, operations, actions, objects, or elements. See section 3.2.

4 WIMS Object Model

The Printer Working Group (PWG) has defined a simplified printing model as part of the Semantic Model that represents printing in Web Services, traditional client/server or peer-to-peer print paradigms. The model identifies:

- A Printer object, which may contain zero or more Jobs.
- A Job object, which is contained in only one Printer Object. A Job can contain zero or more Documents and a Document is contained in only one job.

4.1 WIMS Model Objects Added to the Semantic Model

WIMS adds the following objects to the PWG Semantic Model. The definitions may include references to operations and actions defined in Section 6.

- Agent
- Alert
- Device
- Manager
- Report
- Resource
- Schedule
- Service
- Subscription
- Subunit
- System

The following definitions are the formal definitions for the model, as distinguished from the conceptual definitions given under terminology in Section 2.

4.1.1 Agent

An Agent is an abstract object representing a software component of a network host system that supports management of one or more Services or Devices. An Agent object always supports the WIMS Agent Interface as a request generator and may support the WIMS Management Interface as a response generator.

This document defines the Agent object as an extension and generalization of the SNMP Agent defined in the IETF Architecture for Describing SNMP Management Frameworks [RFC3411].

4.1.2 Alert

An Alert is abstract object representing the set of information associated with a normal event (e.g., ServiceConfigChanged) or exception event (e.g., ServiceStopped) on a managed entity. An Alert may be transferred to a Manager via a SendAlerts operation.

This document defines the Alert object as an extension and generalization of the Alert element defined in the IETF Printer MIB [RFC1759] [RFC3805].

4.1.3 Device

A Device is an abstract object representing a hardware component of a network host system that supports at least one imaging function (e.g., copy). A Device may be associated with one or more upstream Service objects. As used in this document, a Device exposes for monitoring and management every associated Subunit (e.g., Marker) on the associated network host system.

This document considers the device as an extension and generalization of the Printer element as defined in the IETF Printer MIB [RFC1759] [RFC3805].

4.1.4 Manager

A Manager is an abstract object representing a software component of a network host system that supports management of one or more managed entities via their associated Agents. A Manager object always supports the WIMS Agent Interface as a response generator and may support the WIMS Management Interface as a request generator.

This document defines the Manager object as an extension and generalization of the SNMP Manager defined in the IETF Architecture for Describing SNMP Management Frameworks [RFC3411].

4.1.5 Report

A Report is an abstract object that represents the set of information associated with the performance of a scheduled or immediate action (e.g., GetElements). A Report may be transferred to a Manager via a SendReports operation.

This document defines the Report object as an extension and generalization of the Get-Printer-Attributes response defined in the IETF Internet Printing Protocol (IPP) Model and Semantics [RFC2566] [RFC2911].

4.1.6 Resource

A Resource is an abstract object representing a software component of a network host system that is necessary for the operation of one or more Services or Devices (e.g., fonts or firmware).

This document defines the Resource object as an extension and generalization of the Resource object defined in the ISO Document Processing Application (DPA) [ISO10175].

4.1.7 Schedule

A Schedule is an abstract object that represents a set of planned actions and their timetables on a network host system. A Schedule object may be transferred to an Agent via a SetSchedule operation request (WIMS Management Interface) or a GetSchedule or RegisterForManagement operation response (WIMS Agent Interface).

This document defines the Schedule object as an extension and generalization of the Schedule element defined in the IETF Schedule MIB [RFC3231].

4.1.8 Service

A Service is an abstract object representing a software component of a network host system that supports one or more imaging functions (e.g., copy, print, and scan) and that may be associated with

one or more downstream Device objects. A Service exposes for monitoring and management every associated Subunit (e.g., Channel) on that network host system.

This document defines the Service object as an extension and generalization of the Printer object defined in the IETF Internet Printing Protocol (IPP) Model and Semantics [RFC2566] [RFC2911].

4.1.9 Subscription

A Subscription is an abstract object that represents a set of events to be monitored on a network host system and a recipient for notifications associated with those events (delivered via WIMS Alerts, SNMP traps, etc.). A Subscription may be transferred to an Agent via a SubscribeForAlerts action in a Schedule.

This document defines the Subscription object as an extension and generalization of the SNMP Subscription defined in the IETF SNMP Applications [RFC3413].

4.1.10 Subunit

A Subunit is an abstract object representing a hardware or software component of a network host system that is accessible for monitoring and management but that cannot be rebooted independently of the owner System, Service, or Device object.

This document imports the definition and semantics of a Subunit from the IETF Printer MIB [RFC1759] [RFC3805], including all of the following standard Subunit types: Channel, Console, Cover, InputTray, Interface, Interpreter, Marker, MediaPath, and OutputBin.

4.1.11 System

A System is an abstract object that represents a network host system and that may support one or more configured Services or Devices on that network host system. A System object exposes for monitoring and management every configured Subunit (e.g., Console) on that network host system.

This document defines the System object as an extension and generalization of the System group in IETF MIB-II (RFC 1213) and the System group in IETF Host Resources MIB [RFC1514] [RFC2790].

4.2 Imaging System Model Including Monitoring and Management

The PWG model contains methods that act upon these objects. The Basic model is shown in grey in Figure 1. WIMS adds methods by which the service and device entities performing the printing can be monitored and managed. These entities and methods are shown in solid black in Figure 1.

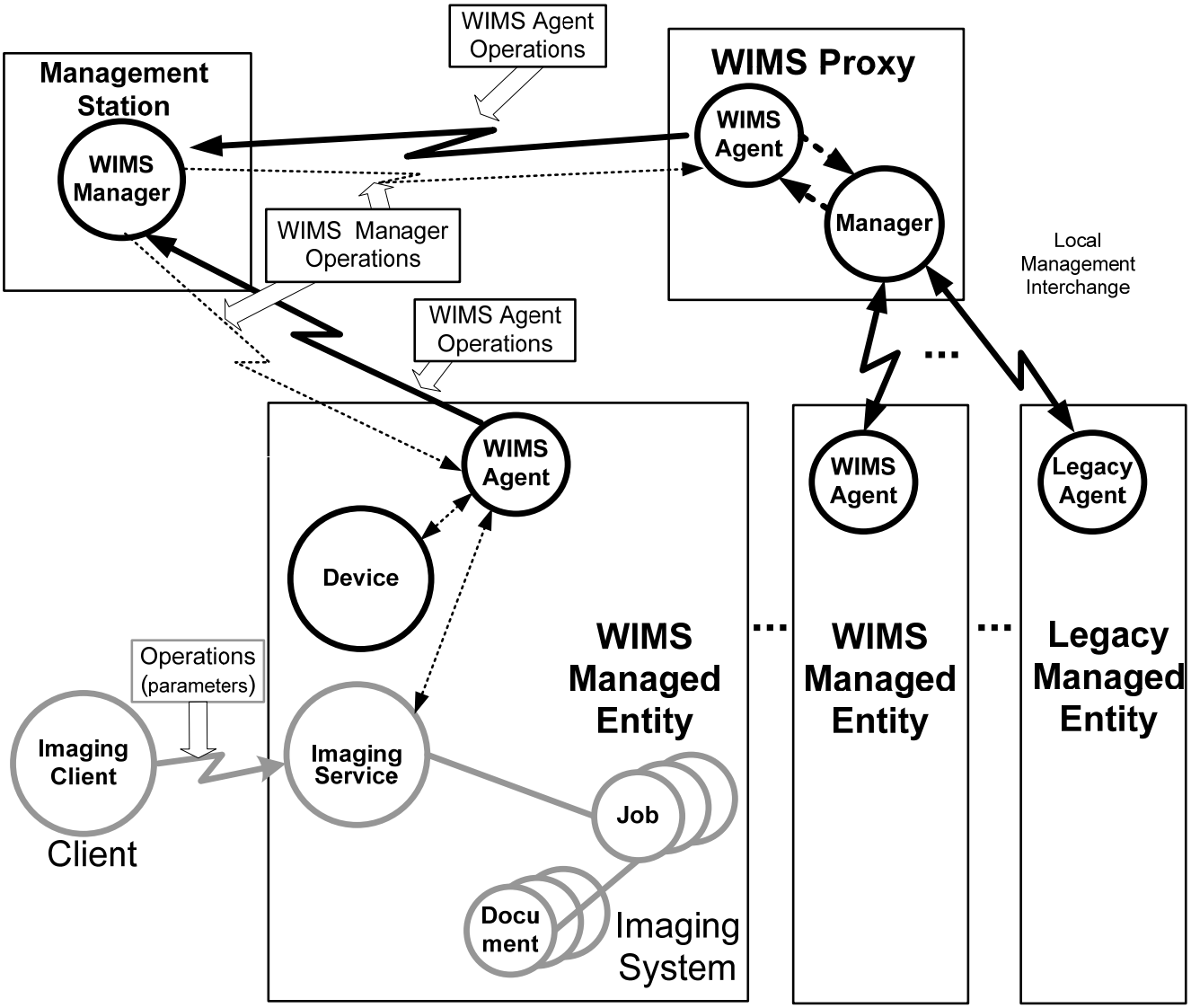

Figure 1 -WIMS Extensions to the PWG Semantic Model

Note that Figure 1 includes a WIMS Proxy, a composite entity consisting of a WIMS Agent and one or more WIMS Managers and/or Legacy Managers, and some logic and storage between the two. Although not a basic object in the model, the WIMS Proxy is important practically because it allows the management of legacy, WIMS-unaware entities. The use of WIMS Proxies is discussed in the following examples. The internal communications between the WIMS Agent and the Manager within a WIMS Proxy, and the communications between a Legacy Manager in a WIMS Proxy and the legacy agent in a Managed Entity are not subjects of this standard and are not addressed in this document.

The special case of a WIMS Proxy consisting of a WIMS Agent and a WIMS Manager is useful in establishing management networks with special characteristics, such a single point Internet access.

4.3 Operations and Actions

The details of the WIMS “Interfaces” are presented in Section 6. This summary is intended to assist in understanding of the following “Example” sections. As shown in Figure 1, there are a set of WIMS

Agent Operations, constituting the “Agent Interface”; and an optional set of Manager Operations, constituting the “Manager Interface”. These “Operations”, whether initiated by the Agent or by the Manager, are between the Manager and the Agent but do not directly affect the Managed Entity. Rather, these operations allow “Actions” to be communicated to the Agent, and it is these actions that dictate the communications with the Managed Entity.

Agent Operations are:

RegisterForManagement: establish a relationship
UnregisterForManagement: terminate a relationship
SendReports: agent sends information requested by the manager in a previously received Schedule
SendAlerts: agent sends information based on Schedule-defined alert conditions
GetSchedule; agent “pulling” a Schedule of actions from Manager

Manager Operations are:

BeginManagement: establish a relationship
EndManagement: terminate a relationship
Set Schedule: manager “pushing” a Schedule of actions to an agent
ExecuteAction: manager requesting that agent perform a single immediate action

Some of the Actions, which may be specified in a Schedule or (singly) in an ExecuteAction and are to be performed by the Agent are:

GetElements: get value of specified elements from the Managed Entity (target) at specified time and issue a SendReports
SubscribeForAlerts: monitor target for specified alert conditions and issue SendAlerts when they occur
UnsubscribeForAlerts: stop specified alert condition monitoring
UpdateSchedule: issue a GetSchedule at specified time.

Aside from the *UpdateSchedule* action, by which a WIMS Manager schedules the time at which the WIMS Agent executes a GetSchedule operation, most other actions require the Agent to interact with the Managed Entity (the Target).

4.4 Example of Remote Fleet Management

An example of the sequence of operations between the Agent, Manager, and Managed Entity in a remote management context is represented in Figure 2 and Figure 3. In this example the external WIMS Manager is not allowed to access nodes on the enterprise network. The entities to be managed are standard imaging devices supporting SNMP. A WIMS Proxy, consisting of a WIMS Agent to communicate with the Manager and an SNMP Legacy Manager to communicate with the Managed Entities, provides the interface between Manager and Managed Entities.

The sequence diagrams are keyed with note references for each step in the sequence. The notes are in the descriptive paragraphs.

4.4.1 Establishing the Relationship

In this example, the Manager is part of an external “third party” service that provides status monitoring and usage accounting for selected imaging services in the enterprise. The WIMS Manager is not allowed to initiate access to nodes in the enterprise network or search for services to manage on the network. Personnel at the enterprise, operating through the WIMS proxy, identify which entities (objects) are to be managed, and what operations and actions may be used in the interaction. The

WIMS Proxy then contacts the Manager with a *RegisterForManagement* operation, as shown in Figure 2.

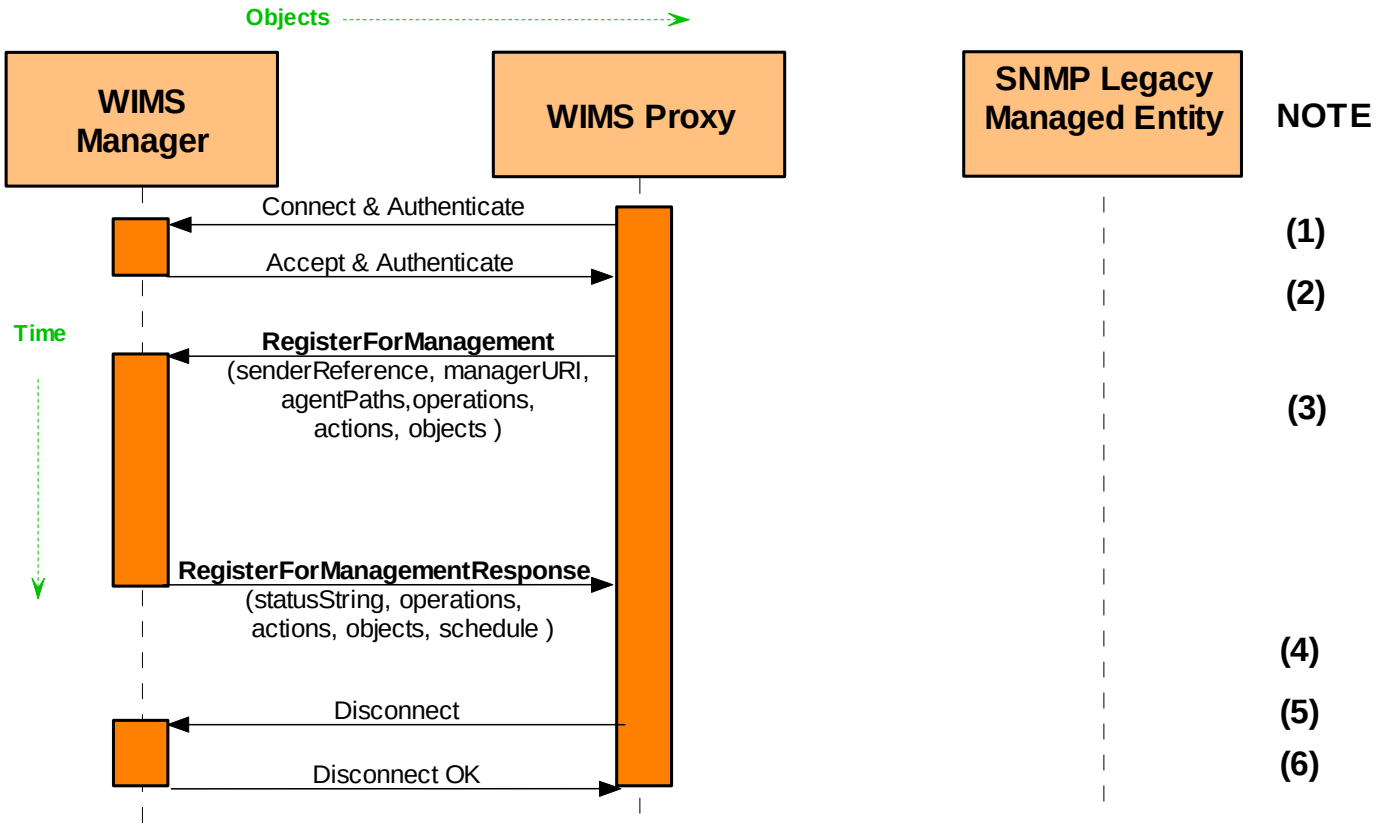

Figure 2 Sample WIMS Sequence Diagram – Agent Establishing Relationship with Manager

(1) WIMS Proxy initiates connection to identified WIMS Manager, including mutual authentication.

(2) WIMS Manager accepts incoming connection from WIMS Proxy and completes mutual authentication.

(3) WIMS Proxy, acting as a WIMS Agent, issues a RegisterForManagment operation, including identification of the agentPaths, objects, operations, and actions supported by the Agent. Operation arguments also include the identification of Agent (sender Reference) and Manager (managerURI).

(4) WIMS Manager accepts management association with WIMS Proxy by returning a RegisterForManagementResponse, specifying its own supported objects, operations, and actions, a status of "SuccessfulOk", and an initial Schedule. In this example, the only Action in the Schedule is an UpdateSchedule, requiring that the Agent execute a GetSchedule operation at some specified later time.

(5) WIMS Proxy sends disconnect indication to WIMS Manager.

(6) WIMS Manager sends disconnect OK to WIMS Proxy.

4.4.2 WIMS Proxy Executing Scheduled Actions

The response to the *RequestForManagement* interchange includes a Schedule. The Schedule is a list of actions that the WIMS Proxy is to perform, along with when they are to be performed. One of these scheduled actions usually is that the WIMS Proxy contact the WIMS Manager with a *GetSchedule*

operation. This provides for a continuing interaction with the WIMS Proxy contacting the WIMS Manager to get an updated Schedule of actions to be performed.

In this example, the WIMS Manager needs to work with the information provided in *RegisterForManagment* to develop a plan for managing the ManagedEntity. The initial Schedule therefore includes only a scheduled *UpdateSchedule* action.

Figure 3 shows the Schedule-*UpdateSchedule* Action-*GetSchedule* Operation sequence which is continually repeated. Acting on the *UpdateSchedule* action in the initial Schedule, the WIMS Proxy contacts the Manager to get an updated Schedule. In this example the Schedule indicates that, at a specific time, the Proxy is to execute a *SendReports* operation sending to the Manager the Printer Marker Life Count data obtained from a specific Managed Entity. The Proxy knows that it must use SNMP to obtain this information. So prior to the specified time, the Proxy obtains the information from the Managed Entity, formulates the appropriate message, and executes the *SendReports* operation.

The Schedule contains other actions, including an *UpdateSchedule,* which is executed at some later time, repeating the sequence with perhaps other actions.

The specific steps in Figure 3 are:

(1) WIMS Proxy initiates connection to identified WIMS Manager, including mutual authentication.

(2) WIMS Manager accepts incoming connection from WIMS Proxy and completes mutual authentication.

(3) WIMS Proxy, acting as a WIMS Agent, sends *GetSchedule* request, including identification of Agent (sender Reference) and Manager (managerURI).

(4) WIMS Manager responds with a PWG standard status of "SuccessfulOk", and a Schedule. In this example, the Actions in the Schedule include an *UpdateSchedule* action. as well as a *GetElements* Action requiring a *SendReports* operation with the sheet count from the Managed Entity..

(5) WIMS Proxy sends disconnect indication to WIMS Manager.

(6) WIMS Manager sends disconnect OK to WIMS Proxy.

(7) At scheduled time, WIMS Proxy issues SNMP GET request to Managed Entity for MIB object prtMarkerLifeCount

(8) Managed Entity responds with prtMarkerLifeCount value

(9) WIMS Proxy incorporates prtMarkerLifeCount into report and initiates connection to identified WIMS Manager, including mutual authentication.

(10) WIMS Manager accepts incoming connection from WIMS Proxy and completes mutual authentication.

(11) WIMS Proxy executes a *SendReports* operation, including the requested object value (sheet count), the identification of Agent (sender Reference) and Manager (managerURI).

(12) WIMS Manager responds with a *SendReports* Response including the status of "SuccessfulOk". There would be an empty UnsupportedElements in this response.

(13) WIMS Proxy sends disconnect indication to WIMS Manager.

(14) WIMS Manager sends disconnect OK to WIMS Proxy.

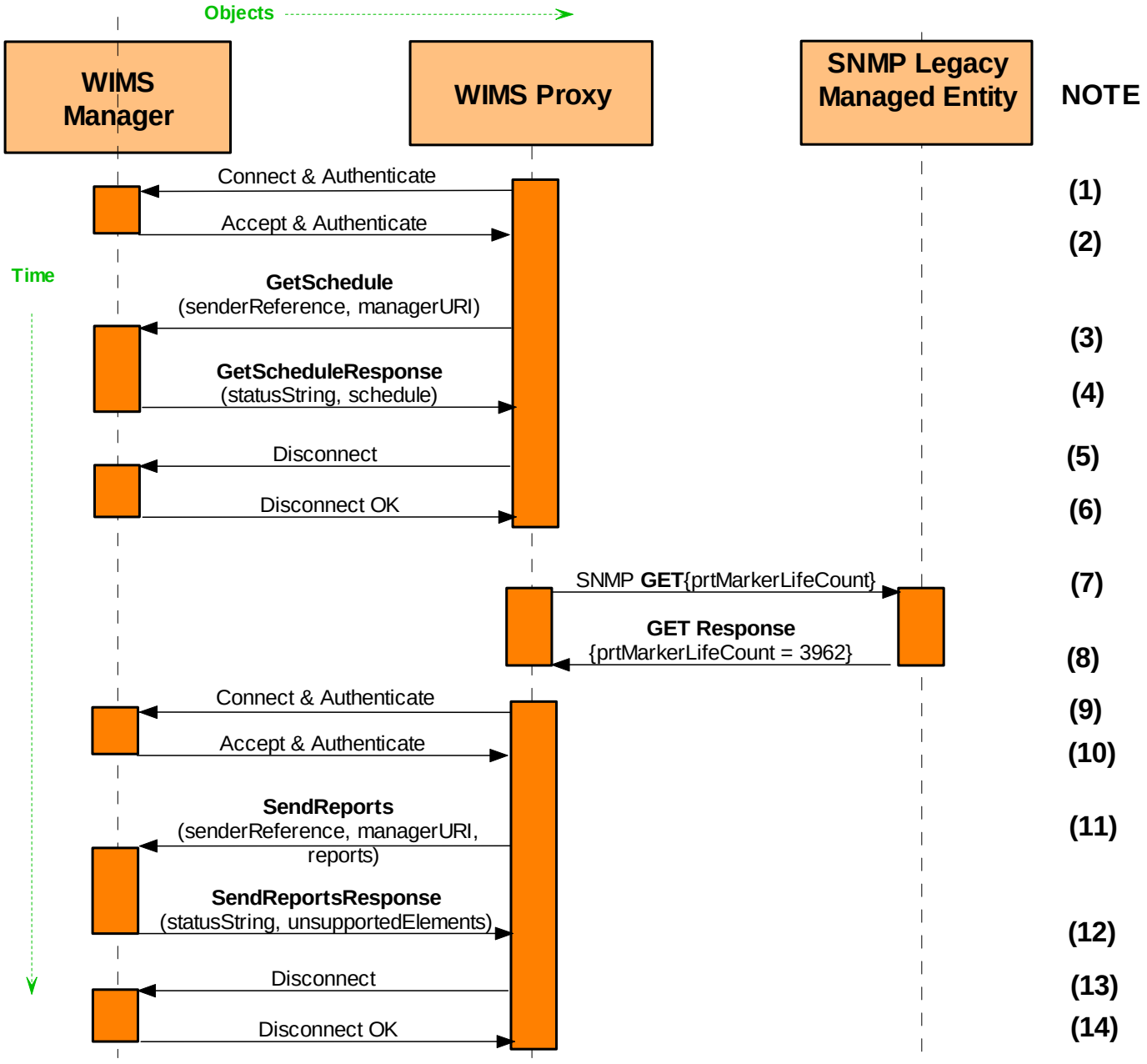

Figure 3 -WIMS Proxy Executing Scheduled Actions

4.5 Example of Intra-Enterprise Management

An example of the relation of a WIMS Manager, WIMS Agent (in a Proxy) and a Legacy Managed Entity, and sequence of operations by which management can be accomplished within an enterprise environment is represented in Figures 4, 5, 6 and 7. Because the WIMS Manager in this example is within the enterprise network, it can initiate access to network nodes and the management sequence is more similar to that of a traditional management protocol. Unlike with remote management where the Schedule mechanism is an important aspect to allow Agent/Manager interaction to continue, the Schedule is not necessary in this environment, although it may still be useful.

4.5.1 Example Sequence - Establishing a Manager-Agent Relationship

In an environment where the WIMS Manager can initiate direct communication with the WIMS Agent (either in WIMS Managed Entity or on a WIMS Proxy), the WIMS Manager can establish the Manager/Agent relationship with a BeginManagement operation. The following steps are keyed to the "notes" key Figure 4.

(1) WIMS Manager (on left in diagram) starts downstream connection to WIMS Proxy including authentication of WIMS Proxy.

(2) WIMS Proxy (in middle of diagram) accepts incoming connection from WIMS Manager and completes authentication.

(3) WIMS Manager proposes management association with WIMS Proxy by sending *BeginManagement* request containing its supported objects, operations, and actions and Schedule object (with a *SubscribeForAlerts* action).

(4) WIMS Proxy (Agent) accepts management association with WIMS Manager by replying with *BeginManagement* response, specifying its own supported objects, operations, and actions and a PWG standard status of "SuccessfulOk" and begins processing the new Schedule (i.e., scanning for Actions that have triggered).

(5) WIMS Manager sends disconnect indication to WIMS Proxy.

(6) WIMS Proxy sends disconnect OK to WIMS Manager.

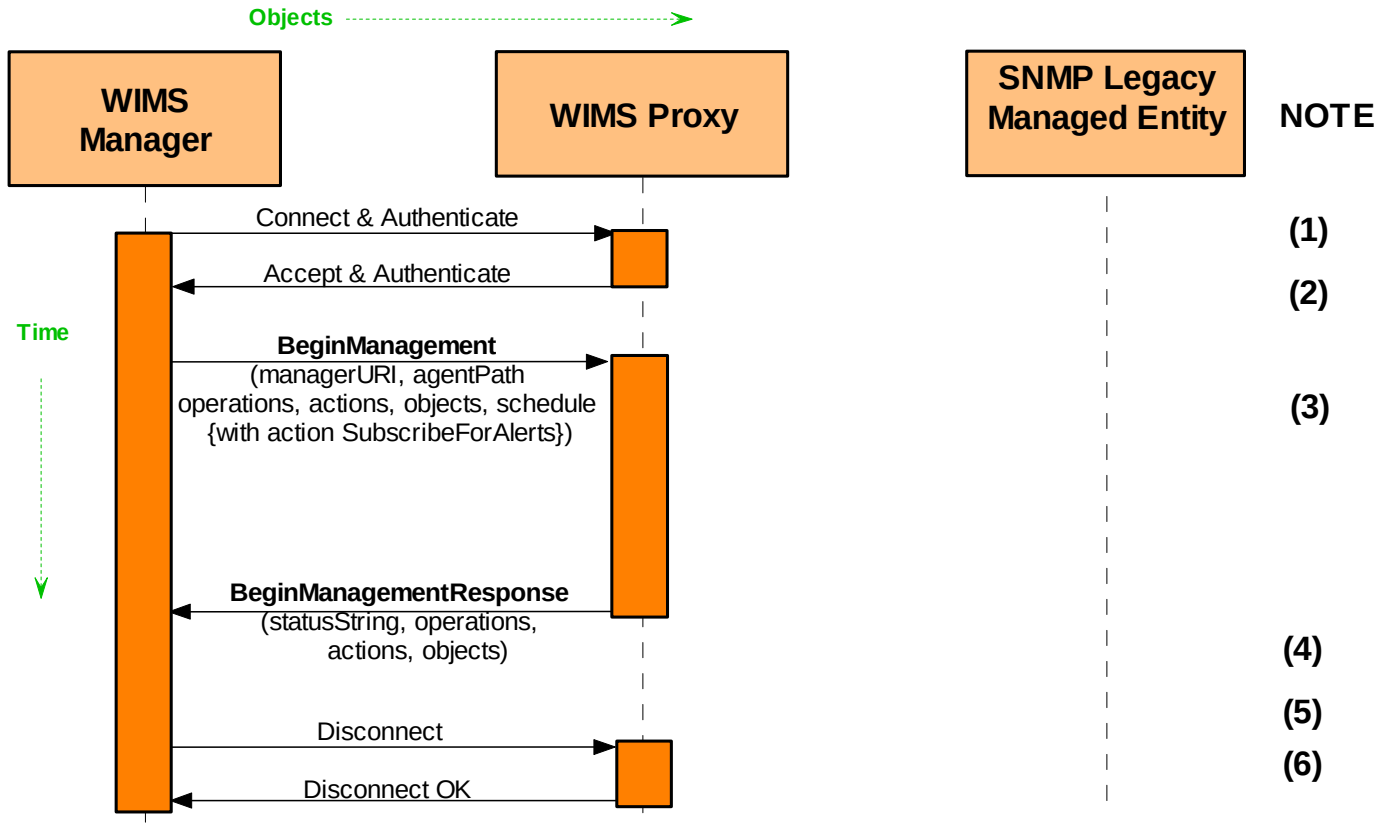

Figure 4 -Enterprise Management - Association

4.5.2 Example Sequence – Scheduled Agent "Send Reports" Action

The *BeginManagement* operation in the previous example included a Schedule of actions. One of the actions in the Schedule was a *SubscribeForAlerts* action to be performed at a specified time by the Proxy. The sequence in Figure 5 shows the WIMS Proxy executing the *SendReports* operation associated with the requested action. The following steps are keyed to the "notes" keys in Figure 5.

(1) WIMS Proxy processes *SubscribeForAlerts* action (in Schedule) by creating Subscription object (for event notifications) and starting upstream connection to WIMS Manager including authentication of WIMS Manager.

(2) WIMS Manager accepts incoming connection from WIMS Proxy and completes authentication.

(3) WIMS Proxy sends *SendReports* request to WIMS Manager specifying *NotifyEvents* element (list of events) from Subscription object.

(4) WIMS Manager replies with *SendReports* response specifying PWG standard status of "SuccessfulOk".

(5) WIMS Proxy sends disconnect indication to WIMS Manager.

(6) WIMS Manager sends disconnect OK to WIMS Proxy.

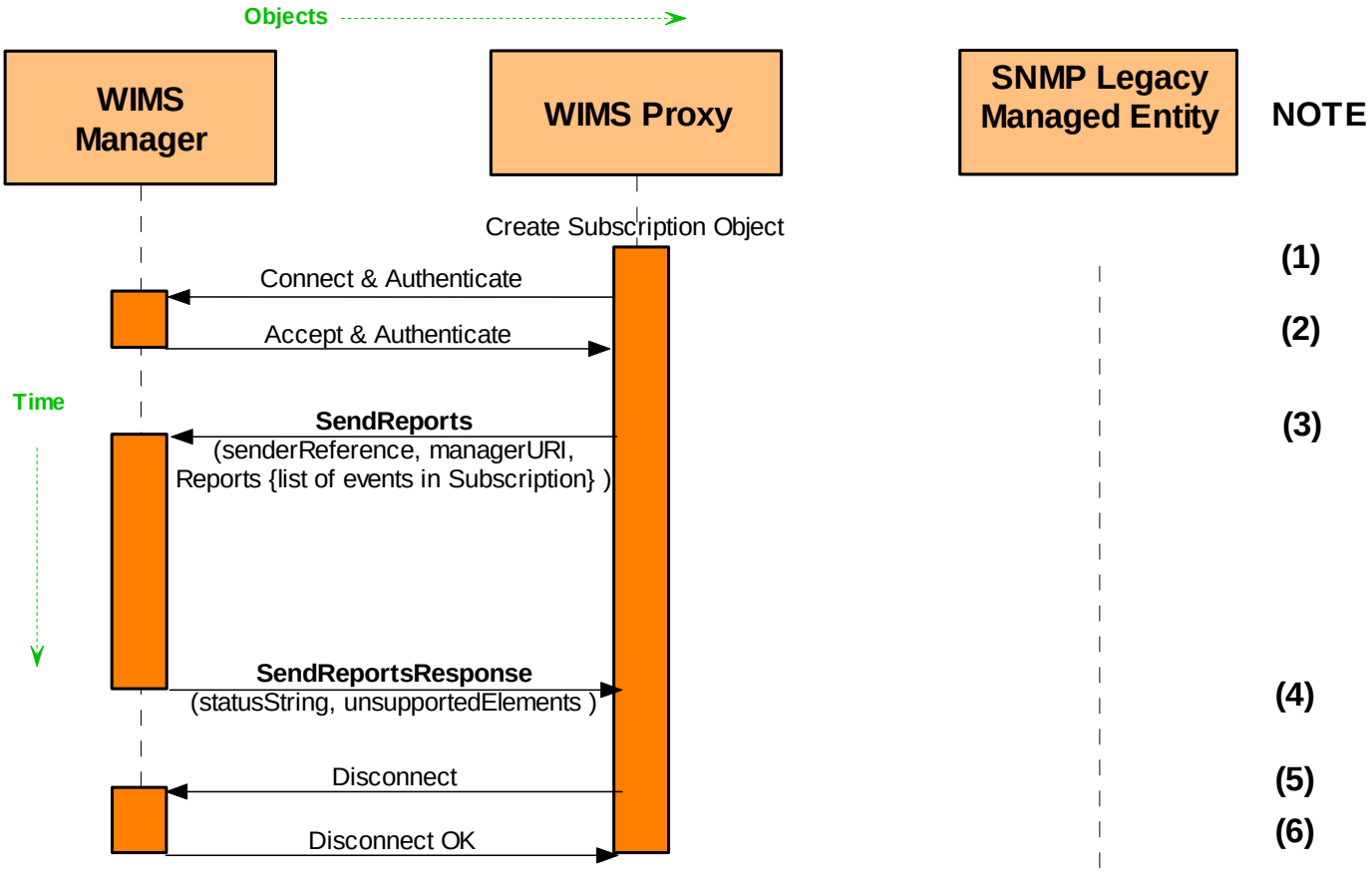

Figure 5 - Enterprise Management - Scheduled Action

4.5.3 Example Sequence - Manager "ExecuteAction" Operation

Once a WIMS Manager/Agent relationship has been established, the Manager can perform *SetSchedule* and *ExecuteAction* operations on the WIMS Agent in the Proxy. The *ExecuteAction* operation identifies an action to be immediately executed by the Proxy and requires an immediate response, more like traditional management operations than scheduled actions. The Action called for in Figure 6 can be executed within the Proxy. The following steps are keyed to the "notes" keys in Figure 6.

(1) WIMS Manager starts downstream connection to WIMS Proxy including authentication of WIMS Proxy.

(2) WIMS Proxy accepts incoming connection from WIMS Manager and completes authentication.

(3) WIMS Manager accesses list of supported WIMS or legacy agents with an *ExecuteAction* Operation containing a *GetElements* action requesting ManagerAgentSupported element in the Manager object of the Proxy.

(4) WIMS Proxy (Agent) processes *GetElements* action by sending *ExecuteAction* response specifying PWG standard status of "SuccessfulOk" and ManagerAgentSupported element from Manager object in a Report.

(5) WIMS Manager sends disconnect indication to WIMS Proxy.

(6) WIMS Proxy sends disconnect OK to WIMS Manager.

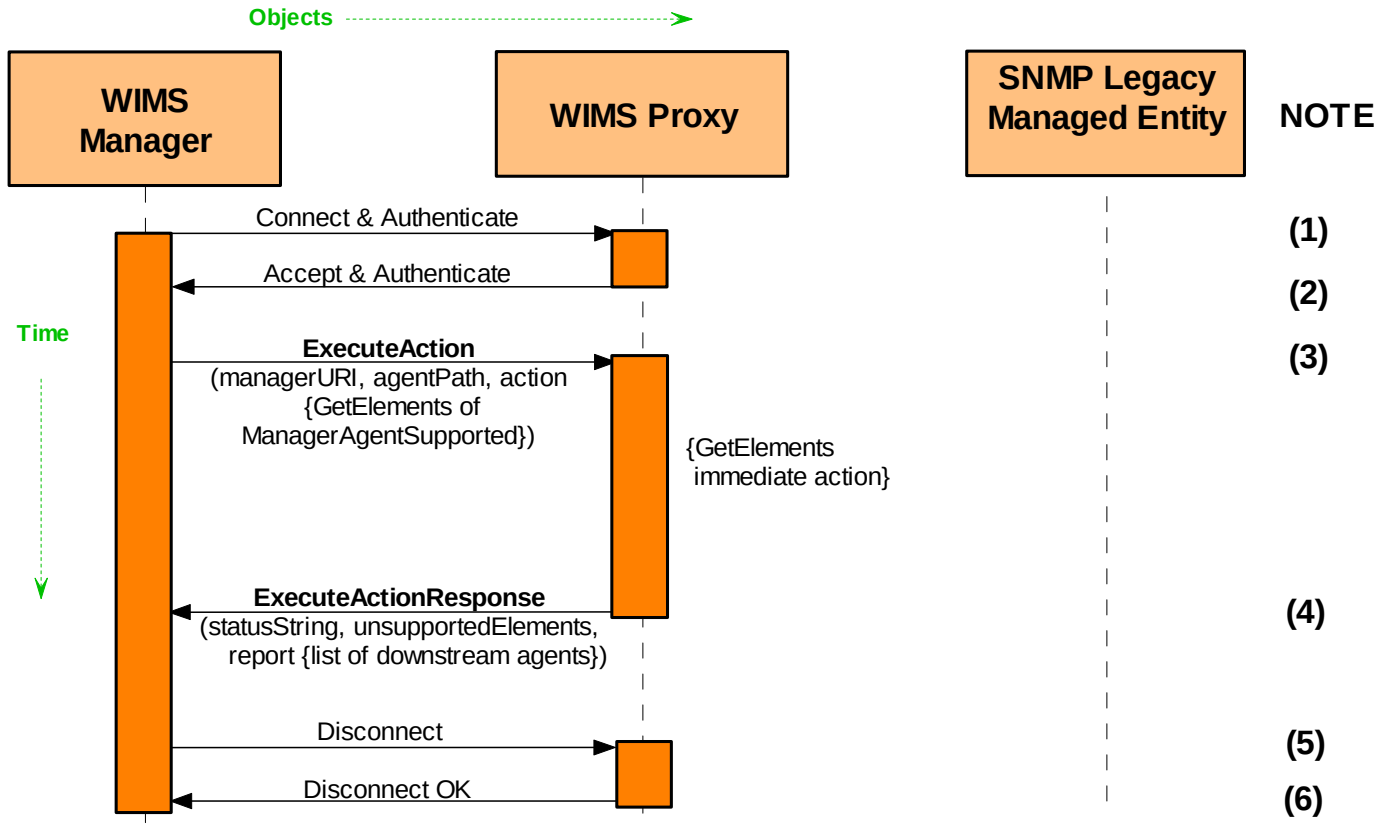

Figure 6 - Enterprise Management - Local Action

4.5.4 Example Sequence –Manager ExecuteAction Operation with Forwarded Action

The following steps are keyed to the “notes” keys in Figure 7. Note that this operation prompts an action requiring the acquisition of data from the SNMP Legacy Managed Entity, and that the return of that data in the ExecuteAction response.

(1) WIMS Manager starts downstream connection to WIMS Proxy including authentication of WIMS Proxy.

(2) WIMS Proxy accepts incoming connection from WIMS Manager and completes authentication.

(3) WIMS Manager acquires a description of the legacy agent by sending to the Proxy an *ExecuteAction* operation containing a *GetElements* action requesting an agent description element.

(4) WIMS Proxy processes *GetElements* action by sending SNMP Get request for sysDescr element in SNMP Agent MIB (RFC 3418, updates RFC 1213) to SNMP Agent in legacy system.

(5) SNMP Agent in legacy system replies with SNMP Get response specifying sysDescr to WIMS Proxy.

(6) WIMS Proxy completes processing of *GetElements* action by sending *ExecuteAction* response specifying PWG standard status of "SuccessfulOk" and Agent Info element (mapped from sysDescr).

(7) WIMS Manager sends disconnect indication to WIMS Proxy.

(8) WIMS Proxy sends disconnect OK to WIMS Manager.

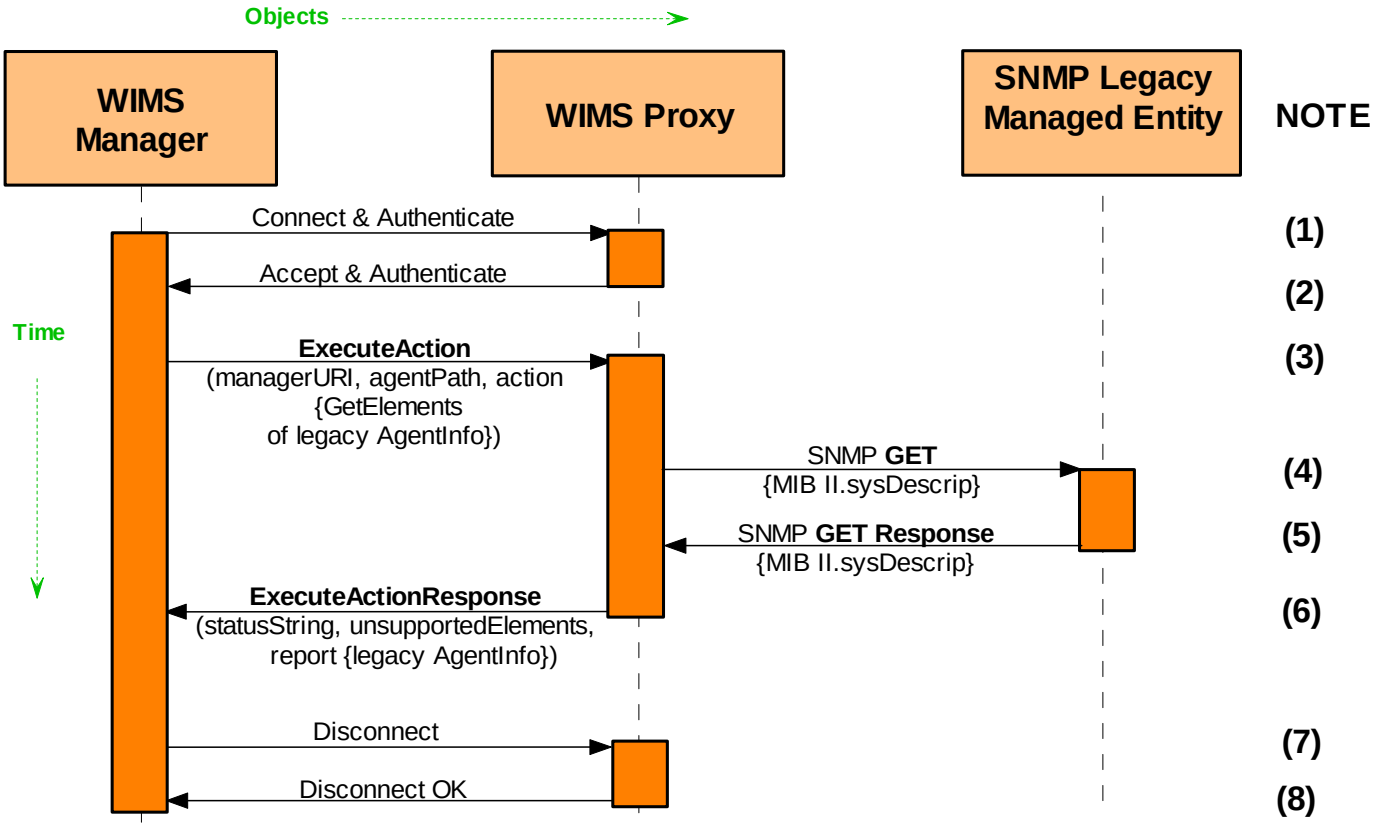

Figure 7 - Enterprise Management - Forwarded Action

4.6 Example of Chained Proxies

For purposes such as single point entry for fleet access, a configuration using chained WIMS Proxies may be used. Because a WIMS Proxy must contain a WIMS Agent, and may contain a WIMS Manager (to communicate with downstream WIMS Agents) and/or a legacy manager (to communicate with downstream legacy agents), the protocol allows the WIMS Agent in one WIMS Proxy to communicate with the WIMS Manager in an up-stream proxy and vice-versa. That is, WIMS Proxy chaining can be used with either management or agent interfaces. A sample configuration is represented in Figure 8. Note that, although this sample uses a relaying of an ExecuteAction operation from WIMS Proxy to WIMS Proxy as an example, Schedule oriented operations could also communicated through chained WIMS Proxies. Implementations are free to decompose the Schedule received by an agent and use the contents of that Schedule to create new Schedule(s) to be passed on to downstream agents. It may be necessary to adjust scheduled action times to compensate for propagation and processing times of the intermediate WIMS Proxy. Alternatively, it is allowed and encouraged that WIMS Proxies execute scheduled actions using recent cached values of states and counters from downstream entities as long as the result is substantively the same as what would have been achieved from forwarding the scheduled action.

The Agent Interface operations provide for chaining WIMS Proxies by specifying the entire path between the Agent operating directly upon the target Managed Entity and the ultimate WIMS Manager. The WIMS Manager Operations provide for this by specifying the entire path between the issuing Manager and the Agent operating directly upon the target Managed Entity. WIMS Proxies do not merely relay operations, the WIMS Managers and WIMS Agent in each proxy perform and respond to Operations; the Agents execute Actions. Proxies do not just pass on Schedules; they retain them and operate on them. Each WIMS Agent, whether in a proxy or in an end Managed Entity, must have inherent knowledge of how to execute the Actions it claims it can perform. Just as the Agent in a Managed Entity must know how to process Actions related to the Managed Entity, an Agent in a WIMS Proxy must know how to set up its associated WIMS Manager to send Schedules to the next Agent in the Path. Similarly, Operations initiated by a terminal Agent must be accepted the first Proxy Manager in the path, which will communicate with its associated Proxy Agent to direct an equivalent Operation to the next Proxy Manager in the path.

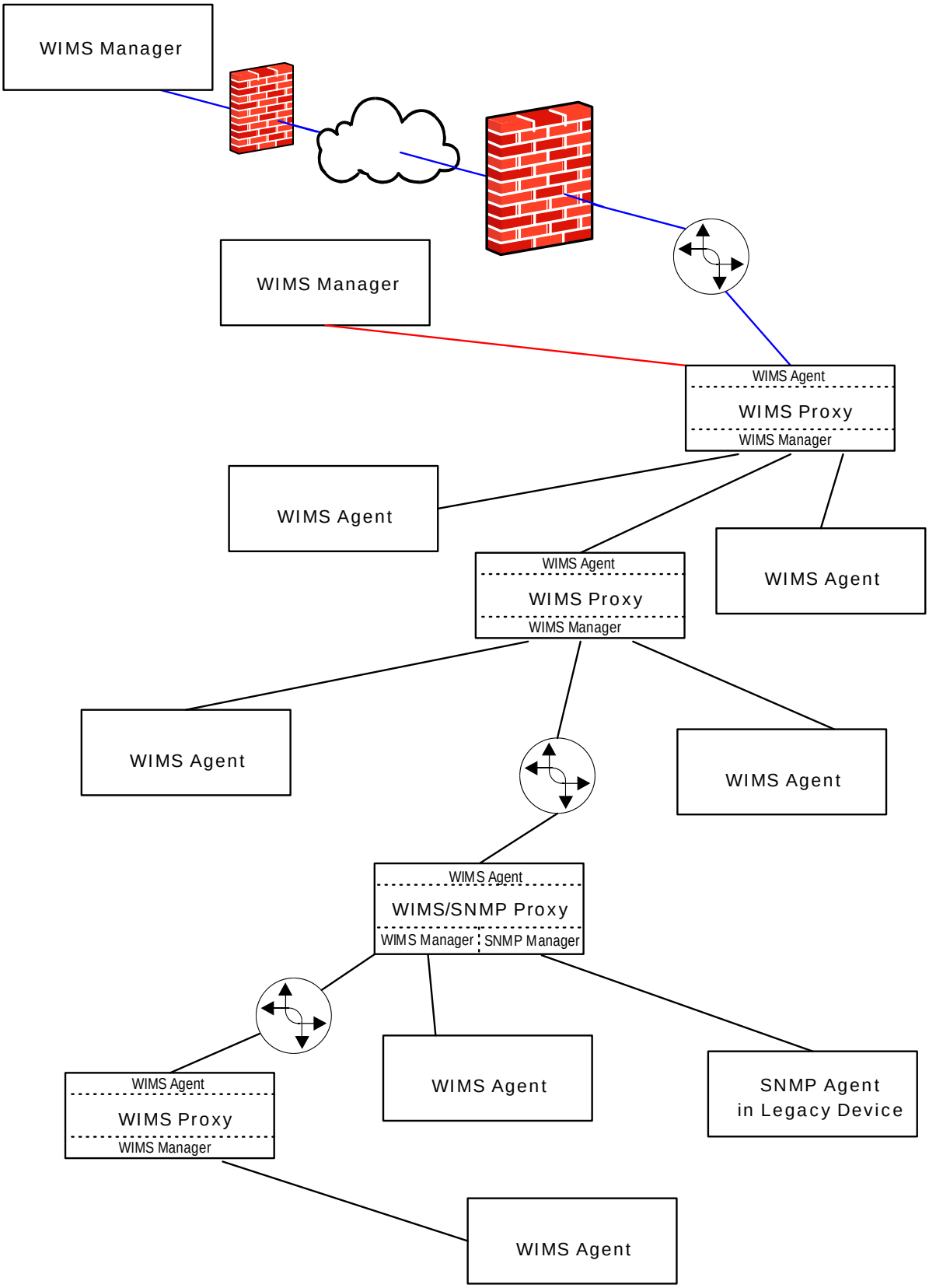

Figure 8 - Example of Chained WIMS Proxies

A sample sequence for an ExecuteAction operation traversing several chained proxies is shown in Figure 9.

(1) WIMS Manager opens downstream connection to first listed WIMS Proxy, including authentication of WIMS Proxy.

(2) WIMS Proxy Agent accepts incoming connection from WIMS Manager and completes authentication.

(3) WIMS Manager sends ExecuteAction request containing GetElements action for AgentInfo element in (mapped) Agent object.

4) WIMS Proxy Agent processes *GetElements* action by having associated WIMS Manager open downstream connection to next WIMS Proxy in the path, including authentication.

(5) Second WIMS Proxy Agent in path accepts incoming connection from WIMS Manager and completes authentication.

(6) WIMS Manager in first proxy rewrites 'AgentPaths' in 'ExecuteAction', removing the WIMS Agent reference it itself (i.e., the first proxy) in each agent path in the array of paths and sends the new ExecuteAction request to the second Proxy Agent in the path.

(7) Second WIMS Proxy Agent processes *GetElements* action by having associated WIMS Manager open downstream connection to next WIMS Proxy in the path, including authentication.

(8) Third WIMS Proxy Agent in path accepts incoming connection from WIMS Manager and completes authentication.

(9) Second WIMS Proxy Manager rewrites 'AgentPaths' in 'ExecuteAction', removing the WIMS Agent reference to itself in each agent path in the array of paths, and sends the new ExecuteAction request to the third Proxy Agent in the path.

(10) Third WIMS Proxy processes *GetElements* action by sending SMNP Get to SNMP Legacy Managed Entity.

(11) SNMP Agent in legacy system replies with SNMP Get response specifying sysDescr to third WIMS Proxy.

(12) Third WIMS Proxy completes processing of *GetElements* action by sending *ExecuteAction* response to Manager in Second WIMS Proxy...

(13) Second WIMS Proxy completes processing of *GetElements* action by sending *ExecuteAction* response to Manager in first WIMS Proxy...

(14) WIMS Manager in second WIMS Proxy sends Disconnect to third Proxy.

(15) Third WIMS Proxy sends disconnect OK to second WIMS Proxy

(16) First WIMS Proxy completes processing of *GetElements* action by sending *ExecuteAction* response to primary WIMS Manager.

(17) WIMS Manager in first WIMS Proxy sends Disconnect to second Proxy.

(18) Second WIMS Proxy sends disconnect OK to first WIMS Proxy

(19) Primary WIMS Manager sends disconnect indication to first WIMS Proxy.

(20) First WIMS Proxy sends disconnect OK to WIMS Manager.

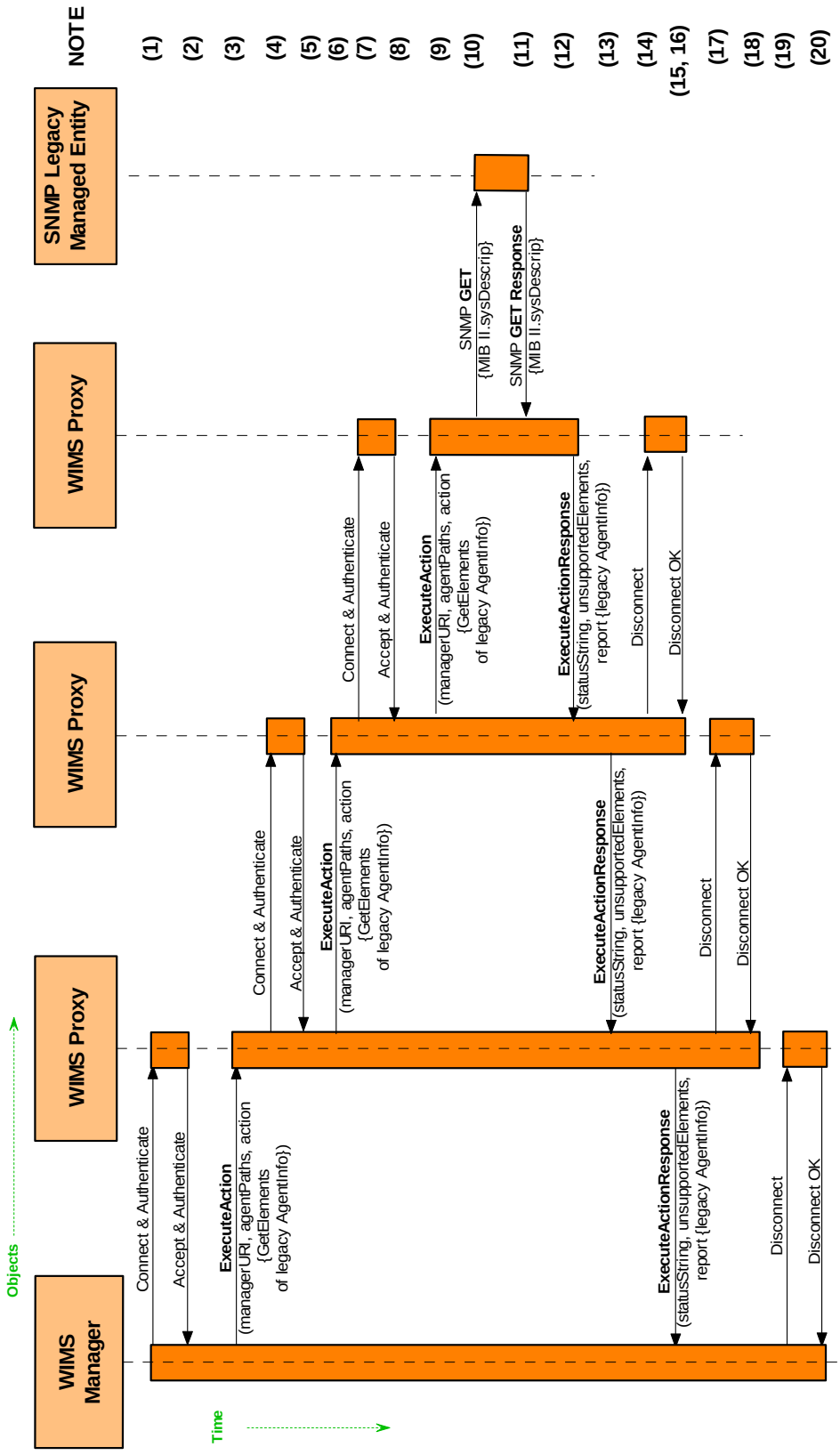

Figure 9- Sequence Diagram of ExecuteAction Operation through Chained Proxies

5 WIMS URI Scheme

5.1 Applicability

The WIMS URI scheme MUST be used to specify only absolute URIs. The WIMS URI scheme MUST NOT be used to specify relative URIs (they would be meaningless). The WIMS URI scheme MUST only be used to specify the use of the WIMS abstract protocol defined in this specification, implemented over one of the protocol bindings defined in paragraph 5.6.2.

Without query parameters, the WIMS URI scheme MUST be used to identify only a WIMS Agent or WIMS Manager as defined in this WIMS specification.

With the WIMS proxy query parameter ('legacy'), the WIMS URI scheme MAY be used to reference other management protocols (SNMP, WBEM, etc.).

Without the WIMS binding query parameter ('binding'), the WIMS URI scheme specifies the WIMS abstract protocol over SOAP/1.2 [SOAP1.2] over HTTP/1.1 [RFC2616] Extensions to the PWG standard query parameters 'auth'(Authentication) and 'sec' (Security) and one new PWG standard query parameter 'binding' (Protocol Stack) are specified below.

The WIMS URI scheme supports communication between WIMS Agents and WIMS Managers:

- In the WIMS Agent Interface, the WIMS Agent always initiates the connection and sends operation requests to the superior WIMS Manager.
- In the WIMS Manager Interface, the WIMS Manager always initiates the connection and sends operation requests to the subordinate WIMS Agent.

The internal communications between the WIMS Agent and the Manager within a WIMS Proxy and the communications between a Legacy Manager in a WIMS Proxy and the legacy agent in a managed entity are not subjects of this standard and are not addressed in this document.

5.2 Associated Port

A WIMS URI that does NOT specify either an alternate port or explicit alternate binding (in the 'binding' query parameter) MUST be resolved to IANA-assigned registered port 4951 for WIMS over HTTP/1.1 [RFC2616], as registered in [IANA-PORTREG].

See: IANA Port Numbers Registry [IANA-PORTREG].

5.3 Associated MIME Types

A WIMS URI that does NOT specify an alternate presentation layer (in the 'binding' query parameter) MUST be used to identify WIMS Agents or WIMS Managers that support the "application/soap+xml" MIME type [RFC3902] (for SOAP bindings) or the "text/xml" MIME type [RFC3023] (for direct XML bindings encoded in UTF-8 [RFC3629]).

See: IANA MIME Media Types Registry [IANA-MIME].

5.4 Character Encoding

A WIMS URI MUST use [RFC3986] encoding. Every character in the "unreserved" set [RFC3986] is equivalent to its percent encoding [RFC3896].

5.5 Syntax

For definitive information on common URI scheme syntax and semantics, see "Uniform Resource Identifier (URI): Generic Syntax and Semantics" [RFC3986].

> Note: The abstract protocol defined in this WIMS specification places a limit of 1023 octets (NOT characters) on the maximum length of a URI. WIMS Agents and WIMS Managers SHOULD be cautious about depending on URI lengths above 255 octets, because older HTTP/1.1 implementations may not properly support these lengths.

A WIMS URI MUST be represented in absolute form, beginning with the scheme name followed by a colon. This WIMS specification imports the normative definitions of 'authority', 'userinfo', 'host', 'port', 'path-absolute', and 'query' from [RFC3986].

The WIMS URI scheme syntax is defined in ABNF as follows:

```
wimsuri = "pwg-wims:" "//" authority [ path-absolute ] [ "?" query ]
```

The 'authority' component is defined by [RFC3986] in ABNF as follows:

```
authority = [ userinfo "@" ] host [ ":" port ]
```

Literal IPv4 or IPv6 addresses SHOULD NOT be used in WIMS URI, for reliability in the context of network reconfigurations, NAT/firewall traversal, and/or mobile managed entities [RFC3986].

The 'port' element SHOULD NOT be used in a WIMS URI for a Legacy Agent (because the port would apply to the WIMS Proxy rather than the actual legacy protocol endpoint). See 'Legacy Agent URI Examples' later in this WIMS specification.

A WIMS URI that does NOT specify either an alternate port or explicit alternate binding (in the 'binding' query parameter) MUST be resolved to IANA-assigned registered port 4951 for WIMS over HTTP/1.1 [RFC2616], as registered in [IANA-PORTREG].

The semantics of a WIMS URI are that it identifies a WIMS endpoint at a WIMS Agent or WIMS Manager listening for incoming connections on the specified host and resolved port, and over HTTP/1.1 [RFC2616] the Request-URI for the identified endpoint is 'path-absolute' (possibly qualified by one or more query parameters).

If the 'path-absolute' element is not present in a WIMS URI, it MUST be given as "/" when used as a Request-URI for a WIMS endpoint (see section 5.1.2 of HTTP/1.1 [RFC2616]).

5.6 Query Parameters

Any of the PWG standard query parameters defined in PSI/1.0 [PWG-5104.2] may be used in specifying a WIMS URI, although some may not always be meaningful (e.g., the 'passive' query parameter is only meaningful for a file transfer binding such as FTP).

Extensions to the PWG standard query parameters 'auth'(Authentication) and 'sec' (Security) and one new PWG standard query parameter 'binding' (Protocol Stack) are specified below.

5.6.1 'auth'

Client authentication mechanism required to access this URI (see section 4.4.2 'uri-authentication-supported' in IPP/1.1 [RFC2911]), for example:

```
pwg-wims://example.com/manager?auth=digest
```

Well-known values are:

```
'none':          no user authentication (i.e., 'anonymous').
'basic':         user authenticated via HTTP basic [RFC2617].
'certificate':   user authenticated via PKI certificate [RFC3280].
'diameter':      user authenticated via Diameter [RFC3588].
'digest':        user authenticated via HTTP digest [RFC2617].
'kerberos':      user authenticated via Kerberos V5 [RFC4120].
'login':         user authenticated via login protocol operation (same as
                 'requesting-user-name' in IPP/1.1 [RFC2911]).
'otp':           user authenticated via One-Time Password [RFC2289].
'radius':        user authenticated via RADIUS [RFC2865].
'sasl':          user authenticated via SASL [RFC2222].
'srp':           user authenticated via Secure Remote Password (SRP)
                 [RFC2945].
```

5.6.2 'binding'

Protocol binding required to access this URI, for example:

```
pwg-wims://manager@example.com?bind=soap11.email
```

Well-known values are:

```
'none':          no protocol binding (i.e., 'not specified').
'soap11.beep':   SOAP/1.1 [SOAP1.1] over BEEP [RFC3080] per [RFC3288].
'soap11.email':  SOAP/1.1 [SOAP1.1] over email (SMTP/IMAP4/POP3).
'soap11.http11': SOAP/1.1 [SOAP1.1] over HTTP/1.1 [RFC2616].
'soap12.beep':   SOAP/1.2 [SOAP1.2] over BEEP [RFC3080] per [RFC3288bis].
'soap12.email':  SOAP/1.2 [SOAP1.2] over email (SMTP/IMAP4/POP3) per [SOAP1.2-
                 EMAIL]
'soap12.http11': SOAP/1.2 [SOAP1.2] over HTTP/1.1 [RFC2616].
'xml.email':     Direct XML [XML] over email (SMTP/IMAP4/POP3).
'xml.http11':    Direct XML [XML] over HTTP/1.1 [RFC2616].
```

5.6.3 'sec'

Security mechanism required to access this URI (see IPP/1.1 [RFC2911], section 4.4.3 'uri-security-supported'), for example:

```
pwg-wims://example.com/manager?auth=digest&sec=tls
```

Well-known values are:

```
'none':          no security mechanism (i.e., 'anonymous').
'pgp':           communications message security via OpenPGP [RFC3156].
'smime':         communications message security via S/MIME [RFC3851].
'ssl3':          communications channel security via SSL3 [SSL].
'tls':           communications channel security via TLS/1.0 [RFC2246].
```


5.7 Examples

5.7.1 WIMS Agent URI Examples

The following are examples of well-formed WIMS URI for WIMS Agents (e.g., for the 'agentReference' parameter in a SetSchedule request):

```
pwg-wims://example.com
pwg-wims://example.com/agent
pwg-wims://example.com/agent&binding=soap11.http11
```

5.7.2 WIMS Manager URI Examples

The following are examples of well-formed WIMS URI for WIMS Managers (e.g., for the 'managerURI' parameter in a GetSchedule request):

```
pwg-wims://example.com
pwg-wims://example.com/manager
pwg-wims://example.com/manager?binding=soap12.beep
```

5.7.3 Legacy Agent URI Examples

The following are examples of well-formed WIMS URI for Legacy Agents (e.g., the 'AgentReference' element of the RequestForManagmenet operation):

```
pwg-wims://example.com?legacy=snmp://bob.com
pwg-wims://example.com/agent?legacy=snmp://bob.com
```

5.8 Normalization and Comparisons

See [RFC3986], section 6 'URI Normalization and Comparison'.

6 WIMS Interface Definition

This section defines the WIMS Interface specifications, independent of specific protocol bindings. Two interfaces are described. These are represented in Figure 10 below.

1. WIMS Agent Interface: (represented by the solid circle in Figure 10). This interface includes all operations initiated by the WIMS Agent, either in a Managed Entity or a WIMS Management Proxy. These operations allow the WIMS Agent to get instructions from a remote WIMS Manager allowing full remote management of entities in an environment where the remote Manager is not allowed direct access to the managed entities (because of firewall restrictions, for example). However, the WIMS Agents are allowed outbound access to the Manager (as HTTP clients with World Wide Web access, for example).

2. WIMS Manager Interface: (represented by the stared circle and dashed line in Figure 10). This interface includes operations initiated by the WIMS Manager, and could be used if there were no Manager to WIMS Management Proxy Agent or Managed Entity Agent access restriction. This interface is more typical of traditional management interfaces.

Note that, in addition to a Management Station (which contains a WIMS Manager) and a Managed Entity (which may contain a WIMS Agent or a Legacy Agent), Figure 10 also shows a WIMS Proxy. A WIMS Proxy includes a WIMS Agent, characterized by a WIMS Agent Interface, and one or more WIMS and/or Legacy Managers. The internal communications between the WIMS Agent and the Manager within a WIMS Proxy, and the communications between a Legacy Manager in a WIMS Proxy and the legacy agent in a Managed Entity are not subjects of this standard and therefore are not addressed in this document.

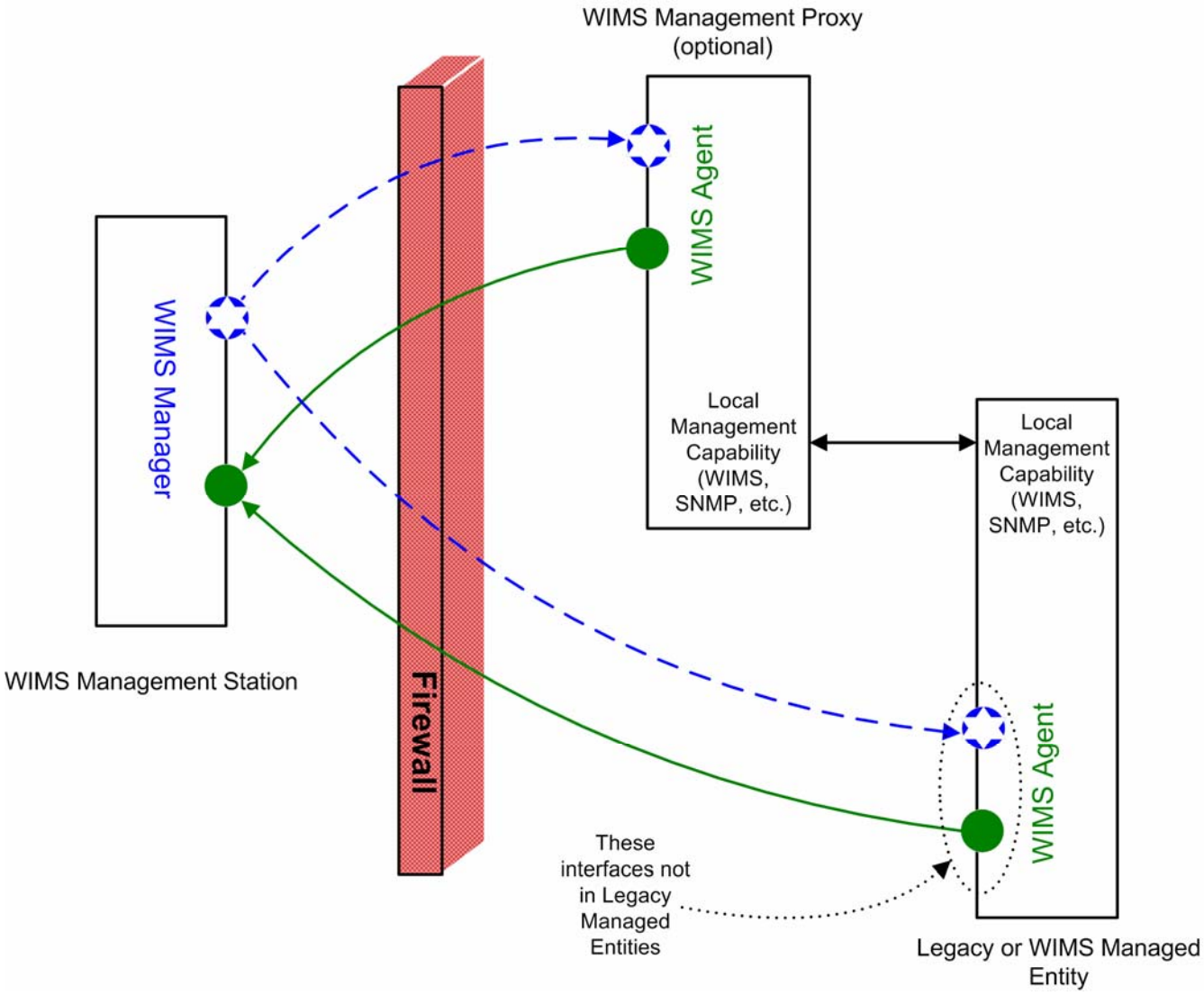

Figure 10 - WIMS Interfaces

Note that the “Operations”, whether initiated by the Agent or by the Manager, establish a relationship between the Manager and the Agent but do not directly affect the Managed Entity. Rather, these operations allow “Actions” to be communicated to the Agent, and it is these actions that dictate the communications with the Managed Entity.

6.1 Operation Parameters and Responses

The WIMS Agent Interface consists of a set of "Operations" that the WIMS Agent can initiate; these are defined in paragraph 6.2. The WIMS Manager Interface consists of a set of "Operations" that the WIMS Manager can initiate; these are identified in paragraph 6.3. Some operations can result in the delivery of a Schedule to a WIMS Agent. A Schedule contains one or more Actions that the WIMS Agent is to take, along with the time or conditions under which the Action is to be taken. The WIMS Management interface ExecuteAction operation defines a single specific action that the WIMS Agent is to process immediately WIMS Actions are defined in paragraphs 6.4, 6.5 and 6.6.

The operations and actions in the following paragraphs are defined in the form:

```
OperationName (OperationParameters) Responses, and

ActionName (ActionParameters)
```

This paragraph provides a detailed definition of all of the Parameters and Responses referenced in the Interface descriptions.

6.1.1 Operation Parameters

`action : Action`
: Single action for immediate execution. This Action instance MUST conform to the XML Schema defined in the PWG Semantic Model [PWG5105.1] module 'Schedule.xsd'.

`actions : WIMSActionsSupported`
: List of WIMS protocol actions supported by this WIMS Agent in the case of Agent Operations, and by the WIMS Manager in the case of Manager Operations.

`agentReference : AgentReference`
: Reference to the target WIMS Agent to be managed. This value MAY use the 'pwg-wims:' URI scheme (see section 'WIMS URI Scheme') or it MAY be an enterprise-specific name, asset number, etc., to protect the privacy of the enterprise network

`agentPaths : AgentPaths`
: Paths to the WIMS Agents or Legacy Agents most directly associated with the Managed Entities being registered for management. This list may include the WIMS Agent identified in the *SenderReference* parameter.

`alerts : Alerts`
: Sets of exception information. Each Alert Instance MUST conform to the XML Schema defined in the PWG Semantic Model [PWG5105.1] module 'Alert.xsd'.

`managerURI : URI`
: URI of the destination WIMS Manager for Agent Operations and the source WIMS Manager for Manager Operations. This value MUST use the 'pwg-wims:' URL scheme (see section ' WIMS URL Scheme') and MUST conform to [RFC2396].

`objects : WIMSObjectsSupported`
: List of objects supported by this WIMS Agent in the case of Agent Operations, and by this WIMS Manager in the case of Manager Operations.

`operations : WIMSOperationsSupported`
: List of WIMS protocol operations supported by this WIMS Agent in the case of Agent Operations, and by this WIMS Manager in the case of Manager Operations.

`reports : Reports`
: Sets of information associated with an ActionPlan. Each Report Instance MUST conform to the XML Schema defined in the PWG Semantic Model [PWG5105.1] module 'Report.xsd'.

`schedule : Schedule`
: Set of planned actions and their timetables. This Schedule Instance MUST conform to the XML Schema defined in the PWG Semantic Model [PWG5105.1] module 'Schedule.xsd'.

`senderReference : AgentReference`
: Reference to the WIMS Agent initiating the operation. The value MUST use the 'pwg-wims:' URL scheme (see section ' WIMS URL Scheme') and MUST conform to [RFC2396].

6.1.2 Operation Responses

Responses are with respect to an operation and the returned values must be considered relative to that operation.

`actions : WIMSActionsSupported`
: List of WIMS protocol actions that may be requested by the WIMS Manager in the case of Agent Operations and of the Actions supported by the Agent in the case of Manager Operations. The list should be a subset of the actions listed in the soliciting operation.

`objects : WIMSObjectsSupported`
: List of WIMS objects that may be accessed by the WIMS Manager in the case of Agent Operations and of the objects supported by the Agent in the case of Manager Operations.. The list should be a subset of the objects listed in the soliciting operation

`operations : WIMSOperationsSupported`
: List of WIMS Protocol operations supported by the WIMS Manager in the case of Agent Operations and by the Agent in the case of Manager Operations. The list should be a subset of the operations listed in soliciting operation.

`schedule : Schedule`
: A set of planned actions and their timetables. This Schedule Instance MUST conform to the XML Schema defined in the PWG Semantic Model [PWG5105.1] module 'Schedule.xsd'.

`statusString : StatusString`
: A keyword, returned by the Manager in the case of Agent Operations and by the Agent in the case of Manager Operations, reflecting its status with respect to the operation, at the time of the response. Conforms to definition in the Semantic Model [PWG5105.1] 'PwgCommon.xsd'.

`unsupportedElements : UnsupportedElements`
: An UnsupportedElements list that contains the elements that, as specified in the action, are not supported by the WIMS Manager. Conforms to definition in the Semantic Model [PWG5105.1] 'PwgCommon.xsd'.

6.1.3 Action Parameters

`agentReference : AgentReference`
: Reference to the WIMS Agent which is to do the GetSchedule

`agentPaths : AgentPaths`
: Paths to the WIMS Agents or Legacy Agents most directly associated with the Managed Entities.

`mandatoryElements : Keyword`
: Fully qualified names of the mandatory elements that MUST all be successfully set or else the action MUST fail.

`mode : PauseModeType`
: Pause mode (after current page, document, or job).

`mode : RestartModeType`
: Restart mode (current configuration cold boot, current configuration warm boot, previous good configuration cold boot, factory default configuration cold boot).

`mode : StartupModeType`
: Startup mode (current configuration cold boot, previous good configuration cold boot, factory default configuration cold boot).

`notifyElements : Keyword`
: Fully qualified names of elements to include in notifications.

`notifyEvents : NotifyEvents`
: List of event names to be delivered via SendAlerts and defined in the PWG Semantic Model [PWG5105.1] module 'Events.xsd'.

`notifyRecipientURI : URI`
: URI of the WIMS Manager to be notified - these value MUST use the 'pwg-wims:' URL scheme and MUST conform to [RFC2396].

`requestedElements : Keyword`
: Fully qualified names of requested elements.

`requestedResources : RequestedResources`
: List of requested resources specified by ResourceId values (keys), ResourceName values (local names), and/or a resource filter (set of resource elements and values that all MUST be satisfied for match)..

`resetMode : RestartModeType`
: Reset mode (none, current configuration cold boot, current configuration warm boot, previous good configuration, factory default configuration). If this parameter is missing or 'None', then the specified target elements are NOT reset to a previous configuration but are instead explicitly set to the newly requested values. This parameter supports reset of selected elements to their previous values without requiring a Restart of the WIMS Agent.

`subscriptionElements : SubscriptionElements`
: Optional fully qualified names and strongly typed values of Subscription object elements defined in the PWG Semantic Model [PWG5105.1] module 'Alert.xsd'. Support by WIMS Agents of 'visible' Subscription objects (can be queried with GetElements) is OPTIONAL.

`subscriptionID : SubscriptionID`
: Unique numeric code identifying the specific subscription. If this is given a value of ‘-1’, the WIMS agent will assign a valid value. In either case, the subscriptionID is returned by the WIMS agent in the SendReports operation to the manager requesting the subscription.

`targetElements : TargetElements`
: Fully qualified names and strongly typed values of target elements to be added or modified by the action.

`targetObjects : TargetObjects`
: A sequence of identifiers (URIs, names, or integers) which specify the desired target objects (i.e., scope) of this GetElements action. (In this case, a Target Object can be a managed entity, document, job, Schedule, or subscription). Particularly if the Manager is external to the enterprise, this term should not include any information about the network and is more reasonable a neutral term such as an asset number.

`targetResources : TargetResources`
: List of target resources specified by ResourceId values (keys), ResourceName values (local names), and/or a resource filter (set of resource elements and values that all MUST be satisfied for match)..

`vendorAction : Keyword`
: Fully qualified name of vendor extension action.

`vendorParameters : VendorParameters`
: Vendor extension parameters from any namespace.

6.1.4 Status Strings

All WIMS operation responses MUST include processing status in the 'statusString' parameter, which contains one of the well-known values defined by the PWG Semantic Model/1.0 [PWG5105.1] in the 'StatusStringWKV' enumeration in 'PwgWellKnownValues.xsd'. The semantics of each value are defined by IETF IPP/1.1 [FC2911] in section 3.1.6.1 'status-code' and in section 13 'Status Codes and Suggested Status Code Messages'.

6.2 WIMS Agent Interface

Each of the following operations is invoked by a WIMS Agent (supporting a service or device) and is sent to a WIMS Manager. This interface allows:

1. Managed Entities to be identified as being manageable by the specified WIMS Manager through a WIMS Agent or Legacy Agent,
2. A WIMS Agent associated with a Managed Entity or with a WIMS Management Proxy to solicit a Schedule of management instructions, and
3. A WIMS Agent to report requested information in a scheduled report or alert.

All operations return status strings that indicate success or identify exceptions.

These operations are designed for WIMS Proxies as well as for WIMS Agents dedicated to single, specific Managed Entities. That is, all agent operations may apply to multiple Managed Entities, with the scope given within the Schedule, Report, or Alert parameter or in the RegisterForManagment AgentReferences list.

The SenderReference parameter within these operations always refers to the WIMS agent originating the operation. The SendReports and SendAlerts operations contain, within the Reports or Alert parameters, separate terms that identify the agent most directly associated with the specific Managed Entity to which the report or alert applies. This agent may or may not be the same as the Sender, and may or may not be a WIMS agent.

The WIMSAgent, using the RegisterForManagement operation, may make an entity manageable by multiple managers. This capability must be used judiciously to avoid conflicting instructions. However, the RegisterForManagement operation does allow the Agent to state what Operations, Actions and Objects are supported. The Agent could provide full management control to one Manager while limiting others to monitoring actions.

The majority of the operations are initiated by the Agent, largely to get instructions (the Schedule) from the Manager and make reports to the Manager. The Schedule contains instructions indicating the actions that the Agent is to perform, and is the mechanism by which the Manager indirectly interacts with the managed entities. WIMS allows WIMS Agents to:

1. Accept Schedules from multiple Managers.
2. Accept multiple Schedules from any given Manager. Each Schedule has an ID. The interaction of the Agent with a Manager is then defined by the sum of all the requested actions in all Schedules from that Manager.
3. Accept revised Schedules from a given Manager, where a Schedule submission has the same ID as a previously submitted Schedule. In this case, the later Schedule will replace the earlier Schedule with the same ID.

6.2.1 RegisterForManagement

```
RegisterForManagement (senderReference : AgentReference, managerURI :
URI   agentPaths : AgentPaths,
      operations :WIMSOperationsSupported,
      actions : WIMSActionsSupported, objects : WIMSObjectsSupported),
      statusString : StatusString, operations :
WIMSOperationsSupported,
```



```
actions : WIMSActionsSupported, objects : WIMSObjectsSupported,
schedule : Schedule
```

Description:

WIMS Agent identifies Managed Entities as being WIMS-manageable by the target WIMS Manager. WIMS Agent also identifies the specific operations, objects and actions supported for that the WIMS Agent supports (but which may or may not be supported in one or more of the Managed Entities.) WIMS Manager returns operations it is capable of, and the objects and actions it may include in the Schedule, along with an initial Schedule. The WIMS Agent will exercise the supplied Schedule.

Note that RegisterForManagement lists are cumulative: AgentReferences are removed from the ManagedEntity list only by an UnRegisterForManagment operation.

Method Support:

WIMS Agent - REQUIRED to support
WIMS Manager - REQUIRED to support

6.2.2 UnregisterForManagement

```
UnregisterForManagement (senderReference : AgentReference,
     managerURI : URI, agentPaths : AgentPaths)  statusString :
StatusString
```

Description:

WIMS Agent sends a request to cancel management registration for Managed Entities identified by their associated *agentReferences.* That is, the Management Entities uniquely associated with the listed *agentReferences* will no longer be manageable by the target WIMS Manager.

Method Support:

WIMS Agent - REQUIRED to support
WIMS Manager - REQUIRED to support

6.2.3 SendReports

```
SendReports (senderReference : AgentReference, managerURI : URI,
     reports : Reports) statusString : StatusString,
     unsupportedElements : UnsupportedElements
```

Description:

Sends Reports (information requested in or associated with a scheduled or immediate action) to a WIMS Manager from a WIMS Agent. A WIMS Agent performs a SendReports in response to Monitoring Actions (paragraph 0) such as GetElements or GetResources, as well as all Management Actions (paragraph 6.5). Note that a given SendReports operation may include reports associated with more than one Managed Entity.

Method Support:

WIMS Agent - REQUIRED to support
WIMS Manager - REQUIRED to support

6.2.4 SendAlerts

```
SendAlerts (senderReference : AgentReference, managerURI : URI,
     alerts : Alerts) statusString : StatusString,
     unsupportedElements : UnsupportedElements,
```

Description:

Sends Alerts (each Alert being a set of exception information) to a WIMS Manager from a WIMS Agent. A WIMS Agent performs a SendAlerts in response to a specified event in a SubscribeForAlerts action. A given SendAlerts may information associated with more than one Managed Entity. That is, the WIMS Agent may be programmed to collect low level alerts from several Managed Entities, and to perform a SendAlerts operation containing the collected and/or moderated information.

Method Support:

WIMS Agent - REQUIRED to support
WIMS Manager - REQUIRED to support

6.2.5 GetSchedule

```
GetSchedule (senderReference : AgentReference, managerURI : URI)
      statusString : StatusString,  schedule : Schedule
```

Description:

Sends a request for a Schedule (a set of planned actions with their timetables) to a WIMS Manager from a WIMS Agent. The WIMS Agent MUST subsequently perform the actions identified in the returned Schedule. In the case of a WIMS Management Proxy, a Schedule may include planned actions for more than one managed entity

Method Support:

WIMS Agent - REQUIRED to support
WIMS Manager - REQUIRED to support

6.3 WIMS Manager Interface

Each of the following operations is invoked by a WIMS Manager and is sent to a WIMS Agent in a service, device or WIMS Proxy. The exercise of the Manager Interface requires that the WIMS Agent support a reverse transport method such as SMTP.

6.3.1 BeginManagement

```
BeginManagement (managerURI : URI, agentReference : AgentReference,
      operations : WIMSOperationsSupported,
      actions : WIMSActionsSupported, objects : WIMSObjectsSupported,
       schedule : Schedule) statusString : StatusString,
      operations : WIMSOperationsSupported,
      actions : WIMSActionsSupported, objects : WIMSObjectsSupported)
```

Description:

WIMS Manager sends a request to create a management association with the target WIMS Agent. That is, the target WIMS Agent will now be managed by the source WIMS Manager. WIMS Manager also identifies the specific operations, objects and actions it supports (but which may or may not be supported in the WIMS Agent) and sends an initial Schedule. WIMS Agent returns the operations, objects, and actions it supports. The WIMS Agent will subsequently perform the actions identified in the supplied Schedule.

Method Support:

WIMS Agent - OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.3.2 EndManagement

```
EndManagement (managerURI : URI, agentReference : agentReference)
       statusString : StatusString
```

Description:

WIMS Manager sends a request to cancel a management association with the target WIMS Agent. That is, the target WIMS Agent will no longer be managed by the source WIMS Manager.

Method Support:

WIMS Agent - OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.3.3 Set Schedule

```
SetSchedule (managerURI : URI, agentReference : AgentReference,
     schedule : Schedule) statusString : StatusString
     unsupportedElements : UnsupportedElements
```

Description:

Sends a Schedule (a set of planned actions with their timetables) to a WIMS Agent (service or device) from a WIMS Manager. The WIMS Agent (service or device) MUST subsequently perform the actions in the received Schedule.

Method Support:

WIMS Agent - OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.3.4 ExecuteAction

```
ExecuteAction (managerURI : URI, agentReference : AgentReference,
     action : Action) statusString : StatusString,
     unsupportedElements : UnsupportedElements, reports : Reports
```

Description:

Sends a request to process a single WIMS action to a WIMS Agent, expecting an immediate report.

The receiving WIMS Agent:

1. MUST immediately validate this action request.
2. MUST immediately attempt to execute any well-formed action requested.
3. MUST include a Report of results of this action in the response to the request.

The receiving WIMS Agent MUST NOT generate a SendReports or SendAlerts in direct reaction to this request; explanatory information may be in the statusString contained in the response. .

Method Support:

WIMS Agent - OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.4 WIMS Monitoring Actions

Actions are performed by the WIMS agent and are invoked by an ExecuteAction operation or by inclusion in a Schedule. A Schedule is transferred to a WIMS Agent in:

a) a GetSchedule response (WIMS Agent Interface); or

b) a RegisterForManagement response (WIMS Agent Interface), for the initial Schedule only; or

c) a SetSchedule request (WIMS Manager Interface).

This section defines WIMS monitoring actions that are benign and intended to not disrupt the normal operation of the Managed Entity. Therefore, support for WIMS Monitoring actions is REQUIRED for all WIMS Agents and WIMS Managers. This does not preclude the local configuration of policies disallowing certain actions, particularly with respect to certain elements or values. For example, certain elements or resources may be may be regarded as private information and may be made inaccessible.

All Monitoring actions specify that the WIMS Agent perform some action, and confirm the success or failure of the action by the WIM Agent sending a Report either in a WIMS SendReports operation or an ExecuteAction response.

The SubscribeForAlerts action requires that the Agent assign the subscriptionID. If this action is within a Schedule, the Manager SHOULD Schedule a SubscribeForAlerts with a mode of oneshot and an interval of '0', resulting in immediate execution. In response to a scheduled SubscribeForAlerts action, the Agent MUST execute a SendReports operation including the assigned Subscription ID in the Report. If the SubscribeForAlerts action is requested by ExecuteAction, the assigned Subscription ID must be in the Report included in the response to the request. The appropriate subscriptionID may be specifically identified in an UnsubscribeForAlerts. If the subscriptionID value is '-1' in the UnsubscribeForAlerts, all subscriptions associated with that manager will be cancelled.

Note that second-order failures (e.g., failure to get an element in a GetElements action) will be identified in the GetElements SendReports, or in the status strings of the ExeuteAction response, not as a separate SendAlerts.

6.4.1 GetElements

```
GetElements (agentPaths : AgentPaths,
      targetObjects : TargetObjects, requestedElements : Keyword,
      vendorParameters : VendorParameters)
```

Description:

Read specified elements of service or device. Agent MUST include these values in a SendReports to the WIMS Manager identified in the Schedule or in the Report included in the ExecuteAction response.

See: Get-Printer-Attributes - section 3.2.5 [RFC2911].

Action Support:

WIMS Agent- REQUIRED to support
WIMS Manager - REQUIRED to support

6.4.2 GetResources

```
GetResources (agentPaths : AgentPaths,
      targetObject : TargetObject,
      requestedResources : RequestedResources,
      vendorParameters : VendorParameters)
```

Description:

Read specified resources (e.g., fonts, forms, logos, drivers, firmware) of Managed Entity. WIMS Agent MUST include these values in a SendReports to the WIMS Manager identified in the Schedule or in the Report included in the ExecuteAction response. See: Get-Resources - section 4.1.3 [IPP-RES].

Action Support:

WIMS Agent- REQUIRED to support
WIMS Manager - REQUIRED to support

6.4.3 SubscribeForAlerts

```
SubscribeForAlerts (agentReference : AgentReference,
      notifyRecipientURI : URI, notifyEvents : NotifyEvents,
      notifyElements : Keyword,
      subsciptionElements : SubscriptionElements,
      vendorParameters : VendorParameters)
```

Description:

Create subscription for specified events delivered via SendAlerts operations. The Manager should Schedule a subscribeForAlerts for immediate execution. The Agent MUST include the subscription object in a SendReports operation to the WIMS Manager identified in the Schedule or in the Report included in the ExecuteAction response.

See: Create-Printer-Subscriptions - section 11.1.2 [RFC3995].

Action Support:
WIMS Agent- REQUIRED to support
WIMS Manager - REQUIRED to support

6.4.4 UnsubscribeForAlerts

```
UnsubscribeForAlerts (agentPaths : AgentPaths,
      subscriptionID : SubscriptionID,
      vendorParameters : VendorParameters)
```

Description:
Cancel subscription for event delivery via SendAlerts.

Action Support:
WIMS Agent- REQUIRED to support
WIMS Manager - REQUIRED to support

6.4.5 UpdateSchedule

```
UpdateSchedule (agentReference : AgentReference,
      vendorParameters : VendorParameters)
```

Description:
Indicates that WIMS Agent MUST initiate a GetSchedule operation to the WIMS Manager identified in the Schedule. The UpdateSchedule action must not be included as an ExecuteAction; use SetSchedule instead.

Action Support:
WIMS Agent- REQUIRED to support
WIMS Manager - REQUIRED to support

6.5 WIMS Management Actions

This paragraph describes WIMS management actions. A management action will cause a configuration change in the Managed Entity. Each of the following actions is invoked by an ExecuteAction operation or by inclusion in a Schedule. A Schedule is transferred to a WIMS Agent in:

1. a GetSchedule response (WIMS Agent Interface); or
2. a RegisterForManagement response (WIMS Agent Interface), for the initial Schedule only; or
3. a SetSchedule request (WIMS Manager Interface).
4. A BeginManagement request (WIMS Manager Interface)

Because WIMS management actions may indirectly cause state changes or disrupt operation of the Managed Entity, support for WIMS management actions is OPTIONAL for all WIMS Agents and WIMS Managers. WIMS management actions all result in the WIMS Agent sending a Report either in a WIMS SendReports operation or ExecuteAction response to confirm action success or failure.

6.5.1 Vendor

```
Vendor (vendorAction : Keyword, agentPaths : AgentPaths,
      vendorParameters : VendorParameters)
```

Description:
Vendor extension action.

Action Support:
WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.5.2 SetElements

```
SetElements (agentPaths : AgentPaths,
      targetObjects : TargetObjects,
      targetElements : any, mandatoryElements : Keyword,
      resetMode : restartMode, vendorParameters : VendorParameters)
```

Description:

Write specified elements of service or device. WIMS Agent MUST reflect the success or failure of this action in a SendReports to the WIMS Manager identified in the Schedule or in the Report included in the ExecuteAction response.

See: Set-Printer-Attributes - section 4.1 [RFC3380].

Action Support:

WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.5.3 DeleteResources

```
DeleteResources (agentPaths : AgentPaths,
      targetObjects : TargetObjects
      requestedResources : RequestedResources,
      vendorParameters : VendorParameters)
```

Description:

Delete specified resources of service or device. WIMS Agent MUST reflect the success or failure of this action in a SendReports to the WIMS Manager identified in the Schedule or in the Report included in the ExecuteAction response.

See section 5.3.2 GetResources above.

Action Support:

WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.5.4 SetResources

```
SetResources(agentPaths : AgentPaths,
      targetObjects : TargetObjects
      targetResources : TargetResources, targetElements : any,
      mandatoryElements : Keyword, vendorParameters :
VendorParameters)
```

Description:

Create or modify specified resources on service or device. WIMS Agent MUST reflect the success or failure of this action in a SendReports to the WIMS Manager identified in the Schedule or Schedule or in the Report included in the ExecuteAction response. See SetElements above.

Action Support:

WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.6 WIMS Administration Actions

This section defines WIMS administration actions. Each of the administration actions is invoked by an ExecuteAction operation or by inclusion in a Schedule. A Schedule is transferred to a WIMS Agent in:

1. a GetSchedule response (WIMS Agent Interface); or
2. a RegisterForManagement response (WIMS Agent Interface), for the initial Schedule only; or
3. a SetSchedule request (WIMS Manager Interface).

4. A BeginManagement request (WIMS Manager Interface)

WIMS administration actions all cause immediate state changes and may disrupt operation of the Managed Entity. Therefore, support for WIMS administration actions is OPTIONAL for all WIMS Agents and WIMS Managers.

All WIMS Administrative actions require that the WIM Agent send a Report either in a WIMS SendReports operation or the ExecuteAction response to confirm action success or failure.

Throughout this section, references to [RFC3998] and [RFC2911] are informational.

6.6.1 Disable

```
Disable (agentPaths : AgentPaths,
      targetObjects : TargetObjects,
      vendorParameters : VendorParameters)
```

Description:

Stop input jobs and data.

See: Disable-Printer - section 3.1.1 [RFC3998].

Action Support:

WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.6.2 Enable

```
Enable (agentPaths : AgentPaths, targetObjects : TargetObjects,
       vendorParameters : VendorParameters
```

Description:

Start input jobs and data.

See: Enable-Printer - section 3.1.2 [RFC3998].

Action Support:

WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.6.3 Pause

```
Pause (agentPaths : AgentPaths, mode : PauseMode,
      targetObjects : TargetObjects,
      vendorParameters : VendorParameters)
```

Description:

Stop output jobs and data.

See: Pause-Printer - section 3.2.7 [RFC2911].

Note: 'Deactivate-Printer' as defined in section 3.4.1 of [RFC3998] (a compound operation) can be accomplished in a Schedule by a 'Pause' action followed by a 'Disable'action.

Action Support:

WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.6.4 Resume

```
Resume (agentPaths : AgentPaths, targetObjects : TargetObjects,
       vendorParameters : VendorParameters)
```

Description:

Resume paused and new output jobs and data.

See: Resume-Printer - section 3.2.8 [RFC2911].

Note: 'Activate-Printer' defined in section 3.4.2 of [RFC3998] (a compound operation) can be accomplished in a Schedule by a 'Resume' action followed by an 'Enable' action.

Action Support:
WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.6.5 PurgeJobs

```
PurgeJobs (agentPaths : AgentPaths,
      targetObjects : TargetObjects,
      vendorParameters : VendorParameters)
```

Description:
Remove all jobs (regardless of state).

See: Purge-Jobs - section 3.2.9 [RFC2911].

Action Support:
WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.6.6 Restart

```
Restart (agentPaths : AgentPaths,
      targetObjects : TargetObjects,
      mode : RestartMode, vendorParameters : VendorParameters)
```

Description:
Restart service or device (reboot software instance). May be implemented with PrtGeneralReset in the Printer MIB (RFC1759). See: Restart-Printer - section 3.5.1 [RFC3998].

Action Support:
WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.6.7 Shutdown

```
Shutdown (agentPaths : AgentPaths,
      targetObjects : TargetObjects,
      vendorParameters : VendorParameters)
```

Description:
Shutdown service or device (terminate software instance).

See: Shutdown-Printer - section 3.5.2 [RFC3998].

Action Support:
WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

6.6.8 Startup

```
Startup (agentPaths : AgentPaths,
      targetObjects : TargetObjects,
      mode : StartupMode, vendorParameters : VendorParameters)
```

Description:
Startup service or device (new software instance).

See: Startup-Printer - section 3.5.3 [RFC3998].

Action Support:
WIMS Agent- OPTIONAL to support
WIMS Manager - OPTIONAL to support

7 Conformance

The WIMS includes two types of actors: The WIMS Agent and the WIMS Manager. The description of Operations and Actions in Section 6 identified under the “Method Support” heading, which were required for conformance to this speciation and which were optional. This section summarizes the Operations and Actions that must be supported by each actor for compliance with this specification.

> **Note Well: Compliance to this specification also requires that the applicable requirements of Section 10, Security Considerations, are met.**

Note that the WIMS Management Proxy is a composite entity rather than a distinct actor. The Proxy must contain a compliant WIMS Agent, and may contain a WIMS Manager, both of which must be compliant. Other functionality that may be in the WIMS Management Proxy, such as alternate managers, mapping of asset identifications to managed entity URLs, and user interfaces to set up the management access are implementation dependent and are not appropriate subjects of the WIMS protocol specification.

7.1 WIMS Agent Mandatory Support

7.1.1 WIMS Agent Interface Operations

RegisterForManagement
UnregisterForManagement
SendReports
SendAlerts
GetSchedule

7.1.2 WIMS Manager Interface

None

7.1.3 WIMS Monitoring Actions

GetElements
SubscribeForAlerts
UnsubscribeForAlerts
UpdateSchedule

7.1.4 WIMS Management Actions

none

7.1.5 WIMS Administration Actions

none

7.2 WIMS Manager Mandatory Support

7.2.1 WIMS Agent Interface Operations

RegisterForManagement
UnregisterForManagement

SendReports
SendAlerts
GetSchedule

7.2.2 WIMS Manager Interface

None

7.2.3 WIMS Monitoring Actions

GetElements
SubscribeForAlerts
UnsubscribeForAlerts
UpdateSchedule

7.2.4 WIMS Management Actions

none

7.2.5 WIMS Administration Actions

none

8 PWG and IANA Registration Considerations

This document does not require any new IANA registration support.

This WIMS/1.0 document adds the following standard objects to the existing definitions in the PWG Semantic Model [PWG5105.1]. These objects are defined in Section 4 of this document.

1. Agent
2. Alert
3. Device
4. Manager
5. Report
6. Resource
7. Schedule
8. Service
9. Subscription
10. Subunit
11. System

9 Internationalization Considerations

WIMS/1.0 implementations conform to all the best practice recommendations in "IETF Policy on Character Sets and Languages" [RFC2277].

WIMS/1.0 implementations exchange [XML] information based on XML Schema [XSD] defined in the PWG Semantic Model [PWG5105.1].

The [XML] 'encoding' attribute (which MUST be supplied in WIMS/1.0 requests and responses) SHOULD be set to 'utf-8' [RFC3629], as recommended by [RFC2277].

The [XML] 'lang' attribute (which SHOULD be supplied in WIMS/1.0 requests and responses) SHOULD be set to a value conforming to [RFC3066], as recommended by [RFC2277].

10 Security Considerations

This section defines security conformance requirements applicable to all WIMS implementations. These requirements are in addition to the basic conformance requirements specified in Section 6 'WIMS Interface Definition'. This section conforms to all of the best practice recommendations in "Guidelines for Writing RFC Text on Security Considerations" [RFC3552]. The security considerations include:

(a) Descriptions of the security threats applicable to all WIMS implementations;
(b) Discussions of these security threats by reference to applicable underlying protocol specifications (e.g., HTTP/1.1 [RFC2616]);
(c) Definitions of the corresponding security conformance requirements applicable to all WIMS implementations.

10.1 Internet Threat Model

The following discussion of threats is adapted from section 3 of [RFC3552].

In the Internet threat model, we assume that the end-systems engaging in a protocol exchange have not themselves been compromised. Protecting against an attack when one of the end-systems has itself been compromised is extraordinarily difficult.

By contrast, we assume that the attacker has nearly complete control of the communications channel over which the end-systems communicate. This means that the attacker can read any PDU (Protocol Data Unit) on the network and undetectably remove, change, or inject forged packets onto the wire. This includes being able to generate packets that appear to be from a trusted machine. Thus, even if the end-system with which you wish to communicate is itself secure, the Internet environment provides no assurance that packets which claim to be from that system in fact are.

It's important to realize that the meaning of a PDU is different at different layers. At the IP layer [RFC791], a PDU means an IP packet. At the TCP layer [RFC793], it means a TCP segment. At higher layers it means some kind of higher layer PDU. For instance, at the SOAP layer [SOAP1.2], it means a single SOAP request or response message, while at the HTTP layer [RFC2616], it means a single HTTP request or response message.

10.1.1 Passive Attacks

In a passive attack, the attacker reads packets off the network but does not write them. The simplest way to mount such an attack is to simply be on the same LAN as the victim. On most common LAN configurations, including Ethernet, 802.3, and FDDI, any machine on the wire can read all traffic destined for any other machine on the same LAN. Note that switching hubs make this sort of sniffing substantially more difficult, since traffic destined for a machine only goes to the network segment that machine is located on.

Wireless communications channels deserve special consideration, especially with the recent and growing popularity of wireless-based LANs, such as those using 802.11. Since the data is simply broadcast on well known radio frequencies, an attacker simply needs to be able to receive those transmissions. Such channels are especially vulnerable to passive attacks. Although many such channels include cryptographic protection, it is often of very poor quality.

Passive attacks described in [RFC3552] and considered in WIMS are:

1. Confidentiality Violations - exposure of private business data:

    a. WIMS implementations over HTTP/1.1 [RFC2616] or BEEP [RFC3080] MUST support TLS/1.0 [RFC2616]; and

    b. WIMS implementations over email (SMTP/IMAP4/POP3) MUST support S/MIME [RFC3851] or OpenPGP [RFC3156].

2. Password Sniffing - exposure of passwords by an attacker who can monitor messages on the wire:

    a. WIMS implementations MUST NOT transfer clear text passwords over unencrypted channels; and

    b. WIMS implementations SHOULD throttle login attempts from a given user (e.g., no more than three attempts per minute).

3. Offline Cryptographic Attacks - dictionary attacks on password-based challenge/response protocols such as HTTP Digest defined in [RFC2617] by an attacker who can record a sequence of messages off the wire:

    a. WIMS implementations over HTTP/1.1 [RFC2616] MUST support TLS/1.0 [RFC2246]; and

    b. WIMS implementations over BEEP [RFC3080] MUST support TLS/1.0 [RFC2246] and SASL [RFC2222].

10.1.2 Active Attacks

In an active attack, the attacker writes packets to the network and may read responses from the network. Active attacks that involve sending forged packets but not receiving any responses are called 'blind attacks'.

When IP [RFC791] is used without IPSec [RFC2401], there is no authentication for the sender address. Active attacks that involve forging an IP packet with a false source address are called 'spoofing attacks'.

An IP packet with a forged source address may sometimes be screened out by the network infrastructure. For instance, many packet filtering firewalls screen out all packets with source addresses on the internal network that arrive on the external interface. Note that this provides no protection against an attacker who is inside the firewall. In general, protocol designers should assume that attackers can forge IP packets.

Not all active attacks require forging addresses. For example, the TCP SYN denial of service attack does not require disguising the sender's address. However, it is common practice to disguise one's address in order to conceal one's identity if an attack is discovered.

Active attacks described in [RFC3552] and considered in WIMS are:

1) Message Replay Attacks - transaction replays by an attacker who can record a sequence of messages off the wire.
2) Message Insertion Attacks - message insertion by an attacker who can monitor a sequence of messages on the wire.
3) Message Deletion Attacks - message deletion by and attacker who can intercept messages on the wire.
4) Message Modification Attacks - message modification by an attacker who can intercept messages on the wire.

For protection against attacks (1) to (4) above:

a) All WIMS implementations MUST validate WIMS application protocol sequence numbers;

b) WIMS implementations over HTTP/1.1 [RFC2616] or BEEP [RFC3080] MUST support TLS/1.0 [RFC2246] and SHOULD always use TLS data integrity services; and

c) WIMS implementations over email (SMTP/IMAP4/POP3) MUST support S/MIME [RFC3851] or OpenPGP [RFC3156].

5) Denial-Of-Service Attacks - resource blocking by an attacker who can generate high-volume message traffic (for example, numerous incoming TCP [RFC793] connections from the same remote host or domain):
   a) All WIMS implementations MUST take the active measures against denial-of-service attacks recommended for their implemented protocol stack(s); and

   b) All WIMS implementations SHOULD report denial-of-service attacks to network management stations and/or management personnel.
6) Man-In-The-Middle Attacks - session hijacking by an attacker who can intercept and insert messages on the wire:
   a) All WIMS implementations SHOULD perform mutual authentication during session startup;

   b) WIMS implementations over HTTP/1.1 [RFC2616] MUST support TLS/1.0 [RFC2246] and SHOULD perform both client and server authentication during TLS startup using public key certificates;

   c) WIMS implementations over BEEP [RFC3080] MUST support TLS/1.0 [RFC2246] and SHOULD perform server authentication during TLS startup and client authentication via SASL using public key certificates; and

   d) WIMS implementations over email (SMTP/IMAP4/POP3) MUST support S/MIME [RFC3851] or OpenPGP [RFC3156] and SHOULD perform both initiator and responder authentication via public key certificates.

10.2 Enterprise Threat Model

In the enterprise threat model, we can no longer assume that the end-systems engaging in a protocol exchange have not themselves been compromised. Physical security of workstations and imaging systems in an enterprise network is often the weakest point of security defenses. For example, print jobs typically produce hardcopy, which an attacker can simply steal.

Network security problems are actually worse in an enterprise network. Firewalls and border routers no longer provide any useful protection.

Users and administrators who deploy WIMS-enabled products in enterprise networks:

a) SHOULD enforce the use of mutual authentication by all deployed WIMS-enabled products;

b) SHOULD enforce the use of TLS/1.0 by WIMS implementations over HTTP/1.1 [RFC2616] or BEEP [RFC3080]; and
c) SHOULD enforce the use of S/MIME [RFC3851] or OpenPGP [RFC3156] by WIMS implementations over email (SMTP/IMAP4/POP3).

10.3 Mobile Threat Model

In the mobile threat model, we can no longer defend against attackers based on physical network topology. Mobile clients may access home, business, or commercial WIMS-enabled products via:

1) Public Wireless - Cellular ISPs, IEEE 802.11 wireless Ethernet 'hot spots' in airports, etc.
2) Local Wireless - Bluetooth/IRDA-enabled laptops and printers, etc.

Users and administrators who deploy WIMS-enabled products in mobile networks:

a. SHOULD enforce the use of mutual authentication by all deployed WIMS-enabled products;
b. SHOULD enforce the use of TLS/1.0 by WIMS implementations over HTTP/1.1 [RFC2616] or BEEP [RFC3080];
c. SHOULD enforce the use of S/MIME [RFC3851] or OpenPGP [RFC3156] by WIMS implementations over email (SMTP/IMAP4/POP3); and
d. SHOULD consider the use of IPSec [RFC2401] which offers significant security advantages in mobile networks.

10.4 HTTP Threat Model

WIMS implementations over HTTP/1.1 [RFC2616] are vulnerable to the threats described in section 15 'Security Considerations' of [RFC2616].

10.5 BEEP Threat Model

WIMS implementations over BEEP [RFC3080] are vulnerable to the threats described in section 9 'Security Considerations' of [RFC3080].

10.6 Email Threat Model

WIMS implementations over email are vulnerable to threats described in:

a) SMTP [RFC2821], section 7 'Security Considerations';
b) Internet Message Format [RFC2822], section 5 'Security Considerations';
c) IMAP4 [RFC3501], section 11 'Security Considerations'; and
d) POP3 [RFC1939]. Section 13 'Security Considerations'.

11 References

11.1 Normative References

[ISO10175] ISO. "Information Technology - Document Printing Application (DPA) Part 1: Abstract Service Definition and Procedures", ISO 10175, May 1995.

[PWG5105.1] Zehler, P., Hastings, T., Albright, S., “PWG Semantic Model”, Candidate Standard PWG 5105.1-2004. January 2004. See ftp://ftp.pwg.org/pub/pwg/candidates/cs-sm10-20040120-5105.1.pdf

[RFC1939] Myers, J., Rose, M. "Post Office Protocol V3 (POP3)", RFC1939, May 1996.

[RFC2119] Bradner. "Key words for use in RFCs to Indicate Requirement Levels", RFC2119, March 1997.

[RFC2222] Myers, J. "Simple Authentication and Security Layer (SASL)", RFC2222, October 1997.

[RFC2277]. H. Alvestrand, "IETF Policy on Character Sets and Languages", RFC2277, January, 1998

[RFC2289] Haller, N., Metz, C., Nesser, P., Straw, M. "One-Time Password System", RFC2289, February 1998.

[RFC2616] Fielding, Gettys, Mogul, Frystyk, Masinter, Leach, Berners-Lee. "Hypertext Transfer Protocol - HTTP/1.1", RFC2616, June 1999.

[RFC2790] S. Waldbusser, P. Grillo. "Host Resources MIB v2", RFC 2790, March 2000.

[RFC2821] Klensin, J. "Simple Mail Transfer Protocol", RFC2821, April 2001.

[RFC2822] Klensin, J. "Internet Message Format", RFC2822, April 2001.

[RFC2865] Rigney, C., Willens, S., Rubens, A., Simpson, W. "Remote Authentication Dial In User Service (RADIUS)", RFC2865, June 2000.

[RFC2911] Hastings, Herriot, deBry, Isaacson, Powell. "Internet Printing Protocol/1.1: Model and Semantics", RFC2911, September 2000.

[RFC3023] Murata, M., St. Laurent, S., Kohn, D. "XML Media Types", RFC3023, January 2001.

[RFC3080] Rose, M. "Blocks Extensible Exchange Protocol Core", RFC3080, March 2001.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., Roessler, T. "MIME Security with OpenPGP", RFC3156, August 2001.

[RFC3231] D. Levi, J. Schoenwaelder, "Definitions of Managed Objects for Scheduling Management Operations", RFC 3231, January 2002.

[RFC3280] Polk, W., Ford, W., Solo, D. "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC3280, April 2002.

[RFC3288bis] Rose, M. "Using the Simple Object Access Protocol (SOAP) in Blocks Extensible Exchange Protocol (BEEP)", <draft-mrose-rfc3288bis-02.txt>, May 2005 (IESG approved).

[RFC3380] Hastings, Herriot, Kugler, Lewis. "Internet Printing Protocol (IPP): Job and Printer Set Operations", RFC 3380, September, 2002.

[RFC3501] Crispin, M. "Internet Message Access Protocol V4R1 (IMAP4)", RFC3501, March 2003.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko J. "Diameter Base Protocol", RFC 3588, September 2003.

[RFC3629] Yergeau, F. "UTF-8, a transformation format of ISO 10646", RFC3629, November 2003.

[RFC3805] R. Bergman, H. Lewis, I. McDonald, "Printer MIB v2", RFC3805, June 2004

[RFC3851] Ramsdell, B. "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

[RFC3986] Berners-Lee, Fielding, Masinter. "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.

[RFC3995] Herriot, Hastings. "Internet Printing Protocol (IPP): Event Notifications and Subscriptions", RFC 3995, March 2005.

[RFC3998] Kugler, Lewis, Hastings. "Internet Printing Protocol (IPP):Job and Printer Administrative Operations", RFC 3998, March, 2005.

[RFC4120] Neuman, C., Yu, T., Hartman, S., Raeburn, K. "Kerberos Network Authentication Service (V5)", RFC4120, July 2005.

[SOAP1.2] See [SOAP12-1] and [SOAP12-2].

[SOAP1.2-1] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen. "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, June 2003.

[SOAP1.2-2] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen. "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation, June 2003.

[SOAP1.2-EMAIL] "SOAP Version 1.2 Email Binding", W3C Note, July 2002.

11.2 Informative References

[RFC1514] P. Grillo, S. Waldbusser, "Host Resources MIB", RFC1514, September 1993

[RFC1759] R. Smith, F. Wright, T. Hastings, S. Zilles, J. Gyllenskog, "Printer MIB", RFC 1759, March 1995. (obsoleted by RFC 3805)

[RFC2566] R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell, "Internet Printing Protocol/1.0: Model and Semantics", RFC2566, April 1999; R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell, "Internet Printing Protocol/1.1: Model and Semantics", RFC2911, September 2000.

[RFC3288] O'Tuathail, E., Rose, M. "Using the Simple Object Access Protocol (SOAP) in Blocks Extensible Exchange Protocol (BEEP)", RFC3288, June 2002.

[RFC3411] D. Harrington, R. Presuhn, B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", RFC3411, December 2002

[RFC3413] D. Levi, P. Meyer, B. Stewart, "Simple Network Management Protocol (SNMP) Applications", RFC3413, December 2002

[RFC3902] Baker, M., Nottingham, M. "The 'application/soap+xml' Media Type", RFC3902, September 2004.

[SOAP1.1] Defacto Industry Standard: W3C Note "Simple Object Access Protocol (SOAP) 1.1", Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Nielsen, Satish Thatte, Dave Winer, 8 May 2000. (See http://www.w3.org/TR/SOAP/)

[SOAP1.2-0] Box, Ehnebuske, Kakivaya, Layman, Mendelsohn, Nielsen, Thatte, Winer, "SOAP Version 1.2 Part 0: Primer", W3C Recommendation, June 2003.

12 Authors Addresses

Harry Lewis
IBM
6300 Diagonal Hwy
Boulder, CO 80301
Phone: +1 303-924-5337
Email: harryl@us.ibm.com

Ira McDonald
High North Inc
PO Box 221
Grand Marais, MI 49839
Phone: +1 906-494-2434
Email: imcdonald@sharplabs.com

William A. Wagner
Technical Interface Consulting
214 Graniteville Road
Chelmsford, MA 01824
Email: wamwagner@comcast.net

