HP Operations Manager

Calling OM Incident Web Services from Perl, PHP and Visual Basic

Version: 1.2
Date: October 15, 2010

Manufacturing Part Number: n/a

© Copyright 2010 Hewlett-Packard Development Company, L.P.

Table of Contents

Introduction

HP Operations Manager for UNIX (OMU) and HP Opeations Manager for Windows (OMW) 8.x, the HP Software OM management solution, offer a Web service interface that allows users to control the full event (also known as message or incident by OM) life cycle. These Web services are built based upon the DMTF WS Management (WSMAN) specification. Although some scripting languages offer libraries to access Web services, it is highly likely that these libraries cannot readily be used to communicate with the OM Web services which are implemented according to the WSMAN standard. **This is due to the fact that WSMAN is based upon document style, rather than RPC style Web Services, and makes extensive use of SOAP headers for addressing and control of the WS behavior.** For more information regarding document versus RPC style see: http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/

Several enhancements are necessary to ensure communicate between a Perl, a PHP, or a VB script client and the OM Web service. This document gives an example of how to call the Create() method of the OM WS using a Perl, a PHP or a VB script client.

Structure of this document:

- The chapter WSMAN Web service client using a scripting language describes the components that are necessary to create a WSMAN client. Inside this chapter of the white paper Perl, PHP and Visual Basic (VB) Script are used as an reference scripting language to explain the implementation of the Create() method.
- In the Appendix, all the XML and SOAP examples are listed, as well as the Perl, PHP and VB script code which has been used to implement a client for the Create() method.

Audience

This paper is intended to be used by HP Software consultants, HP Software partners and customers who would like to understand how to implement an OM Web service client using a scripting language.

The whitepaper assumes a general knowledge of

- Perl
- PHP
- VB Script
- SOAP
- XML
- C
- Shared libraries on Unix and Dynamic Link Libraries (DLL) on Windows
- OM
- OM Incident Web service

Scope of the document

This WSMAN Web service white paper covers the Web services from OMU version 8 and OMW version 8.

The latest version of this white paper can be downloaded from
http://support.openview.hp.com/selfsolve/manuals

product “Operations Manager for UNIX” version “8.0”

or

product “Operations Manager for Windows” version “8.1”.

The code attached to this document is just an example of how a client for the Create() method can be implemented using Perl, PHP or VB script. The code does not contain any error handling and is listed without warranty.

WSMAN Web service client using a scripting language

You may want to access the OM Web service using a scripting language such as Perl, PHP or VB script.
Overview of the following chapters:

- Required Components for development and runtime
  This chapter describes the software components which are needed for the development and runtime part.
- An overview of the OM WSMAN Web service
  This chapter describes how the OM WSMAN Web service has been implemented.
- How to retrieve a XML SOAP envelope example?
  This chapter describes how to trace the TCP socket communication and to retrieve the SOAP XML structure.
- Understanding the WSMAN SOAP envelop structure
  This chapter describes the WSMAN Web service and how to create a SOAP envelope manually that can be sent to the Web service
- Implementing an WSMAN Perl client for the Create() method
  This chapter includes a Perl example showing how to implement the Create() method
- Implementing an WSMAN PHP client for the Create() method
  This chapter includes a PHP example showing how to implement the Create() method
- Implementing WSMAN using a Windows VB Script client
  This chapter includes a VB Script example how to implement a WSMAN client using VB script.

Required Components for development and runtime

The following sections include the components and their descriptions which are needed for Perl, PHP and VB script development. At the end of this section there is a section which distinguishes between components needed for development and components needed for runtime only.

- **Perl**
  (http://www.activestate.com)
  You need a Perl Interpreter for the platform on which you are developing and on which you plan to run the client, because the Perl interpreter provides the runtime component for your Perl script. In the following sections it is assumed that the development will be done on a Windows system. To use Perl on a Windows system, download and install Perl from the URL mentioned above.
- **Eclipse 3.3.2 for Java J2EE Developers**
  (http://www.eclipse.org/downloads/packages/release/europa/winter)
  Eclipse is a free Integrated Development Environment (IDE) for which many plug-ins are available. For most development languages, an appropriate Eclipse plug-in is available which offers syntax checking and source-code highlighting or even debugging functionality. It is recommended to download the J2EE package because most of the components which are required to install the plug-ins mentioned below are already included in this package. In addition, this package allows you to create a Java-based WSMAN Web service client which might be helpful to trace the TCP socket communication and to get an idea about the SOAP XML structure.
  - **Eclipse Plugin EPIC 0.5**
    (http://www.epic-ide.org)
    The Eclipse EPIC plug-in provides a development environment for Perl scripts, including syntax check, source-code highlighting, and debug functionality.
  - **Zend PHP Development Tools (PDT) 1.0.3**
    (http://www.zend.com/en/community/pdt)
    The Eclipse PDT plug-in provides a development environment for PHP scripts, including syntax check, source-code highlighting, and debug functionality.

    - **Aptana Studio Eclipse plug-in (optional)**
      (http://www.aptana.com)
      Aptana is a web-development platform which provides a good IDE to develop HTML, XML, PHP and some other Web techniques. If you want to develop Web applications which access the OM WSMAN Web service, it is recommended to use Aptana Studio. Aptana Studio offers a good XML and HTML editor and a synchronization tool, which allows you to synchronize your newly developed files with an existing Web server.
- **TcpTrace 0.8.1**
  (http://www.pocketsoap.com/tcptrace/)
  TcpTrace is used to trace the socket communication between the OM Web service and the client. TcpTrace helps you to debug the generated XML Code and to find errors inside your SOAP messages.
- **PadWalker 1.7**
  (http://search.cpan.org/dist/PadWalker/)
  PadWalker is a Perl module which is available as Source Code only. PadWalker is needed to monitor variables and expressions inside Eclipse during the debugging of Perl code using the EPIC plug-in.
- **UUID-0.03**
  (http://search.cpan.org/~cfaber/UUID/UUID.pm)
  UUID is a Perl module which generates UUIDs. If you use a different method to generate UUIDs, it is up to you whether you use UUID or your own code. For generating SOAP envelopes which conform to the WSMAN specification, a UUID is definitely needed for each generated SOAP envelope.
- **SOAP::Lite 0.71**
  (http://www.soaplite.com)
  SOAP::Lite is the Perl module which performs the socket communication between the Web service client and the server. SOAP::Lite provides necessary functions for sending the SOAP envelope as well as functions to analyze the SOAP envelope returned from the server.
- **Version 0.76**
  (http://search.cpan.org/~jpeacock/version-0.76/lib/version.pod)
  In the most recent version, SOAP::Lite library requires the Perl version module. Most Perl distributions include this module by default. If the required Perl version module is missing, you must download and install the module manually.
- **Libuuid (on Unix/Linux systems)**
  (http://www.rpmseek.com)
  On Unix systems you need a library which provides the necessary system calls for the Perl UUID module to generate and manipulate UUIDs. On Linux there is already a library available which is part of the e2fsprogs suite. On other Unix derivates you can either choose a different library which is already shipped with the OS, or build the libuuid library from the source code. An example in this document explains how this can be realized on a HP-UX 11.23 PA-Risc system. On Windows there is no need to rebuild this library, because the needed UUID functions are already built into the rpcrt4.dll.

Components needed for Perl development

- Perl
- Eclipse 3.3.2 for Java J2EE Developers
- Eclipse Plugin EPIC 0.5
- TcpTrace 0.8.1
- PadWalker 1.7
- UUID-0.03
- SOAP::Lite 0.71
- Version 0.76
- Libuuid (on Unix/Linux systems

Components needed for Perl runtime

- Perl
- UUID-0.03
- SOAP::Lite 0.71
- Version 0.76
- Libuuid (on Unix/Linux systems

Components needed for PHP development

- Eclipse 3.3.2 for Java J2EE Developers
- Zend PHP Development Tools (PDT) plug-in
- TcpTrace 0.8.1

Components needed for PHP runtime

- Web Server with an appropriate PHP module
  For example, on a SuSE Linux Enterprise 9 system the following packages are needed:
  - php4-4.3.4-43.85
  - apache-1.3.29-71.26
  - apache-mod_php4-4.3.4-43.85
  - php4-curl-4.3.4-43.8
  - php4-domxml-4.3.4-43.8

Components needed for Visual Basic Script development & runtime

- Windows Vista or Windows 2008 Server
- Windows XP or Windows 2003 R2 with WS-Management v1.1:
  http://www.microsoft.com/downloads/details.aspx?FamilyID=845289ca-16cc-4c73-8934-dd46b5ed1d33&DisplayLang=en

An overview of the OM WSMAN Web service

As shown in Figure 1, the Web service interface on OM is consistent across both Unix and Windows platforms. The client, written once, can be used for any OM platform.

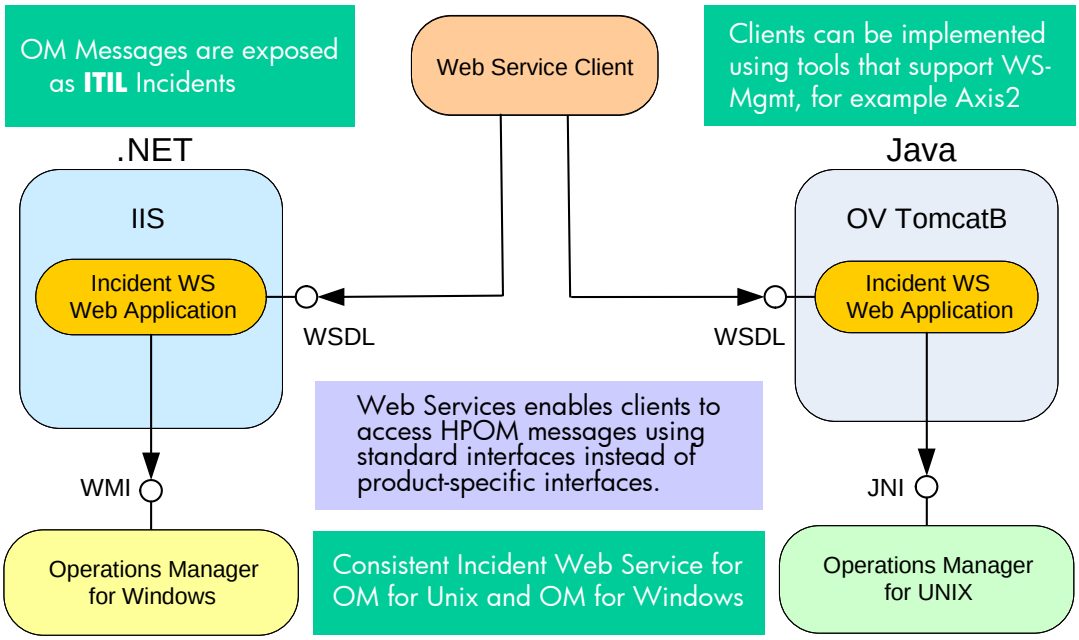

Figure 1: OM Web service overview

To ensure you have a correctly running Web service, try to receive the WSDL by using the following URL:
http://<server>:<port>/opr-webservice/Incident.svc?wsdl
For example, for an OMU system:
http://tcbbn085.deu.hp.com:8081/opr-webservice/Incident.svc?wsdl

The incident Web service was implemented according to the WSMAN standard. The WSMAN SOAP header and body syntax differs from the RPC-style Web service syntax. The Perl SOAP::Lite library and the PHP SOAP extensions only support the RPC-style Web service syntax. Therefore, the Perl SOAP::Lite library and the PHP SOAP extensions are not capable of creating SOAP envelopes which can be understood by the WSMAN Web service. This makes it necessary to manually create SOAP headers and bodies that conform to WSMAN, enclose them in one SOAP envelope, and send the envelope to the WSMAN Web service. Therefore, you use the SOAP::Lite library in Perl only for sending and receiving SOAP envelopes based upon the HTTP protocol. In PHP you use the PHP CURL extensions to send and receive SOAP envelopes based upon the HTTP protocol.

How to retrieve a XML SOAP envelope example?

A good starting point for the retrieval of a SOAP envelope is to build a Java-based WSMAN Web service client, for example: using Axis2 1.4.1 (or a later version) and the WSDL files which are shipped with OM. You should implement the Create() method and trace the TCP socket communication between your Java client and the OM server to get information about the SOAP communication.

The OMU8.0_IncidentWebService.pdf, which is available at http://support.openview.hp.com/selfsolve/manulas, gives you a good idea of how a SOAP envelope for the Create() method must look, if it is to be accepted by the server.

In order to see the actual SOAP envelope exchange you will want to trace the socket communication between the client and the server. To do this, you can use the TcpTrace tool. When communicating via the tool TcpTrace, you must not specify the OM server address directly; instead you can specify 'localhost' and the TCP port TcpTrace is listening on. When starting TcpTrace you will be asked on which port TcpTrace will be listening on and to which server and port the TCP packages being traced should be forwarded. An example is shown in Figure 2. In addition to the listening port, you also specify the destination server and port. In our example, this is the OM server and the Port the incident Web service is listening for incoming requests.

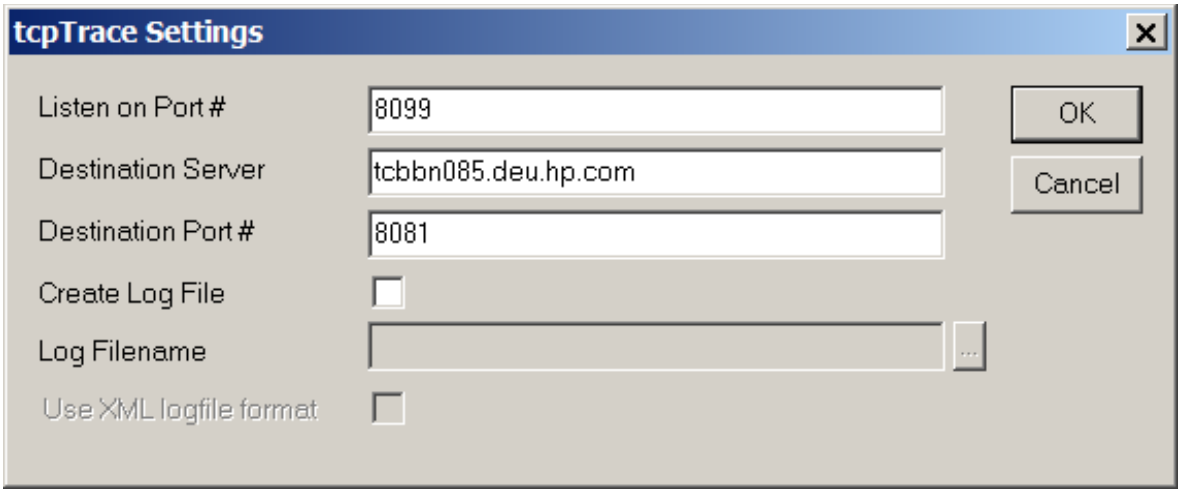

Figure 2: TcpTrace configuration

An example for a traced AXIS2 SOAP envelope is included in chapter AXIS2 generated SOAP envelope example.

Understanding the WSMAN SOAP envelope structure

If you look at the TCP trace and compare it with the SOAP examples in the OMU8.0_IncidentWebService.pdf you can see the structure of the WSMAN SOAP envelope. The SOAP envelope structure is explained below for the Create() method:

1. Each XML tag needs a namespace prefix
2. The namespace prefix can be defined with a short cut in the 'envelope' XML tag. The shortcut must be specified for each XML tag
   Example:
   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <soap:Envelope
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
      <soap:Header>
      .....
      </soap:Header>
      <soap:Body>
       ......
      </soap:Body>
   </soap:Envelope>
   ```
3. The full namespace prefix can be specified in each XML tag. Note that you should not follow this approach, because the XML code becomes unreadable very quickly.
4. The namespace prefix is defined in the parent XML tag and inherited by its child XML tags. If the namespace prefix changes in a child XML tag, the new namespace prefix must be newly defined. Note that it is not recommended to follow this approach, because wrongly inherited namespace prefixes result in errors which are difficult to troubleshoot.

At the beginning you defined all namespaces with their shortcuts. An example of a corresponding SOAP envelope can be found in chapter Create() SOAP TcpTrace example.

WSMAN makes extensive use of *Web Services Addressing (WS-Addressing)*, to address resources exposed by the standard. Therefore, you will need to set several SOAP headers that comply with the WS Addressing as well as the WSMAN standards.

The SOAP Create() method header

The SOAP Create() method header needs the following correctly formulated XML tags:

- <?xml version="1.0" encoding="UTF-8"?>
  Make sure that you use UTF-8 encoding to avoid problems with special characters.
- Required WS Addressing SOAP headers
  - <wsa:Action />
    In this XML tag you specify which action you would like to execute with the Web service call i.e.:
    http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
  - <wsa:To />
    In this XML tag you specify the Web service endpoint. If you are using TcpTrace, you must specify the following URL:
    http://localhost:8099/opr-webservice/Incident.svc
  - <wsa:MessageID />
    In this XML tag you specify the UUID of the SOAP envelope, created from the Perl UUID model:
    b894df7a-854a-47fb-8a81-03288ca1593d

- <wsa:ReplyTo />
  In this XML tag you specify the ReplyTo resource which is always:
  http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous

- Required WSMAN SOAP header
  - <wsman:ResourceURI />
    In this XML tag you define the resource type being acted upon. For the OM Incident WS this is always:
    http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  - <wsman:OperationsTimeout>
    In this XML tag you specify the maximum amount of time the server may take to complete the WS call. If the server cannot complete the call within this amount of time, it is aborted. This header is optional. If not specified, the OM server will use a default of 5 minutes. The following example aborts the Web service call after 10 minutes:
    PT10M0S

An example of a corresponding SOAP envelope can be found in chapter Create() SOAP TcpTrace example.

The SOAP body

The SOAP body contains the required document. For the Create() method this is an XML representation of the Incident you want to create. See the OMU8.0_IncidentWebService.pdf documentation for further details on the Create() and other methods supported. An example of the create document can be found in chapter Create() SOAP TcpTrace example.

Be careful when creating XML tags that may be empty i.e.:

<ns6:NumberOfDuplicates></ns6:NumberOfDuplicates>

Depending of the data type defined in the xsd file, empty XML tags will result in an error processing the request:

<xs:element name="NumberOfDuplicates" type="xs:long" />

is of type long. So if you don't enter at least a '0' in the XML tag, the request will fail, whereas the XML tag

<ns6:Application></ns6:Application>

May be left empty without doing any harm to the processing because of defining the attribute 'Application' as of type string.

<xs:element name="Application" type="xs:string" />

Implementing a WSMAN Perl client for the Create() method

Perl offers a library (SOAP::Lite library) which allows users to connect to a Web service via SOAP, to execute methods and retrieve their results using RPC style Web services. If you implement a Perl script which uses plain SOAP::Lite methods, you will get an error message similar to the following:
406 Not Acceptable
If this happens, you must create the necessary XML SOAP envelopes yourself.

Before you can start writing the Perl code, you must ensure that all necessary components are installed as described in chapter Needed Components for development and runtime. Most components are quite easy to install, only the UUID library is a little more challenging. The following examples describe how the UUID Perl library can be built and installed on different operating systems:

- Installing the Perl 'UUID' library on Linux
- Installing the Perl 'UUID' library on HP-UX
- Installing the Perl UUID library on Windows

The installation of the Perl UUID library also includes information about how to install the optional PadWalk Perl module and the Perl version module, which might be missing on some systems

- Installing optional components

Finally the Perl Create() method is explained in the following chapter

- Creating the Perl client

Installing the Perl UUID library on Linux

On Linux systems all necessary libraries are already available. As a prerequisite you must have a GNU C Compiler, binutils and a Perl interpreter on your system. It is quite easy to install the UUID Perl library using the following steps:

1) Download the source tar ball UUID-0-03.tar.gz
2) Extract it
3) Execute 'perl Makefile.PL'
4) Execute 'make'
5) Execute 'make install'

Installing the Perl UUID library on HP-UX

HP-UX does not include a UUID library as part of the OS; you must compile the UUID library manually. In the following example a HP-UX 11.23 PA-Risc system has been used. The listed components need to be installed before compiling the missing libraries:

- Perl 5.8.8
  http://h21007.www2.hp.com/portal/site/dspp/menuitem.863c3e4cbcdc3f3515b49c108973a801?ciid=2608b3f1eee02110b3f1eee02110275d6e10RCRD
- Gcc 4.1.1
  http://h21007.www2.hp.com/portal/site/dspp/menuitem.863c3e4cbcdc3f3515b49c108973a801?ciid=2a08725cc2f02110725cc2f02110275d6e10RCRD
- GNU binutils 2.16
  http://h21007.www2.hp.com/portal/site/dspp/menuitem.863c3e4cbcdc3f3515b49c108973a801?ciid=2a08725cc2f02110725cc2f02110275d6e10RCRD
- GNU Make 3.81
  http://hpux.connect.org.uk/

Make sure that the $PATH variable contains the appropriate entries so that you can execute the gcc, ar, as, and make executables. If the necessary entries are not in your $PATH variable, add them with the following command:

export PATH=/opt/hp-gcc/bin:/usr/local/bin:$PATH

To create the libuuid, download the e2fsprogs source rpm. Extract it to get the e2fsprogs-1.40.2.tar.bz2 file. Extract this compressed tar file on your HP-UX system and execute the following steps:

1) 'cd e2fsprogs-1.40.2'
2) Execute './configure'
   Note that you might get errors regarding missing components, depending of the packages that are installed on your system. Download and install the missing packages either from the HP website http://www.hp.com or from the 3rd party website http://hpux.connect.org.uk/.
3) Execute 'cd lib/uuid/'
4) Edit the Makefile and do the following changes:
   a. Add –fPIC into the CFLAGS variable
      -fPIC ensures that gcc will generate Position Independent Code which is needed to link shared libraries.
   b. Check whether in some variables '+DAportable' or '+DS2.0' is set. If yes, delete these entries.
5) Execute 'gmake'
6) Execute 'gmake install'
   This command will generate an archive (.a) file which includes the necessary object (.o) files. This archive file is needed to link the UUID library. The root directory of the archive and the include file is /usr/local.

After compiling the libuuid object files, you can build the Perl 'UUID' library.

1) Download the source tar ball
2) Extract the tar ball
3) Execute 'perl Makefile.PL'
4) Edit the Makefile and do the following changes:
   a. Check whether in some variables '+DAportable' or '+DS2.0' is set. If yes, delete these entries.
   b. Ensure that the LD library path contains /usr/local/lib (i.e. add it to the LDDLFLAGS variable in the Makefile with –L/usr/local/lib) otherwise the linker will not be able to find the libuuid.a archive.
5) Execute 'gmake'
6) Execute 'gmake install'

→ Now the UUID Perl library is built.

Installing the Perl UUID library on Windows

On Windows you need to modify the UUID.xs file before you can execute 'perl Makefile.PL'. The C-code inside the UUID.xs files is not suitable for Windows systems because the rpcrt4.dll which is shipped with Windows provides necessary functions to create UUIDs. The modified UUID.xs code for Windows systems can be found in chapter UUID.xs original and Windows modified code.

Make sure that you have a Visual Studio development environment installed on your system.

Execute the following steps:

1) Execute 'VCVARS32.BAT'
   This script will set the necessary environment variables to use Visual Studio executables on the command line.
2) Extract UUID-0.03.tar.gz
3) Edit the file UUID.xs and replace the content with the code you find in chapter UUID.xs original and Windows modified code.
4) Execute 'perl Makefile.PL'
5) Add rpcrt4.dll into the Makefile
   EXTRALIBS = C:\PROGRA~1\MICROS~2\VC98\LIB\RPCRT4.lib
   LDLOADLIBS =C:\PROGRA~1\MICROS~2\VC98\LIB\RPCRT4.lib
6) Execute 'nmake'
7) Execute 'nmake install'

Installing optional components

As mentioned above, the PadWalker library is needed for monitoring variables when debugging your Perl code in Eclipse using the EPIC plugin. In some Perl packages the version library might be missing. The version module is needed from the SOAP:Lite library. If the version library is missing you must install it from the source packages.

Installing the PadWalker library on Windows:

1) Execute 'VCVARS32.BAT'
   This script will set the necessary environment variables to use Visual Studio executables on the command line.
2) Extract PadWalker-1.7.tar.gz
3) Execute 'perl Makefile.PL'
4) Execute 'nmake'
5) Execute 'nmake install'

Installing the PadWalker library on Linux:

1) Extract PadWalker-1.7.tar.gz
2) Execute 'perl Makefile.PL'
3) Execute 'make'
4) Execute 'make install'

Installing the PadWalker library on HP-UX:

1) Make sure that the $PATH variable contains the appropriate entries so that you can execute the gcc, ar, as, and make executables. If the necessary entries are not in your $PATH variable, add them with the following command:
   export PATH=/opt/hp-gcc/bin:/usr/local/bin:$PATH

2) Extract PadWalker-1.7.tar.gz
3) Execute 'perl Makefile.PL'
4) Edit the Makefile and do the following changes:

   Check whether in some variables '+DAportable' or '+DS2.0' is set. If yes, delete these entries

5) Execute 'gmake'
6) Execute 'gmake install'

Installing the version library on Windows:

1) Check if the library already exists, if not continue
2) Execute 'VCVARS32.BAT'
   This script will set the necessary environment variables to use Visual Studio executables on the command line.
3) Extract version-0.76.tar.gz
4) Execute 'perl Makefile.PL'
5) Execute 'nmake'
6) Execute 'nmake install'

Installing the version library on Linux (check first, to see if it already exists):

1) Check if the library already exists, if not continue
2) Extract version-0.76.tar.gz
3) Execute 'perl Makefile.PL'
4) Execute 'make'
5) Execute 'make install'

Installing the version library on HP-UX:

1) Make sure that the $PATH variable contains appropriate entries so that you can execute the gcc, ar, as, and make executables. If the necessary entries are not in your $PATH variable, add them with the following command:
   export PATH=/opt/hp-gcc/bin:/usr/local/bin:$PATH
2) Extract version-0.76.tar.gz
3) Execute 'perl Makefile.PL'
4) Edit the Makefiles in all subdirectories and do the following changes:

   Check whether in some variables '+DAportable' or '+DS2.0' is set. If yes, delete these entries

5) Execute 'gmake'
6) Execute 'gmake install'

Creating the Perl client

After you have completed all the steps above and ensuring that you understand the WSMAN SOAP envelope structure as well as the XML namespace concept, it is easy to write an appropriate Perl client.

The client is divided into two parts:

1) The actual client
   The Perl code can be found in chapter Perl Client
2) A package which creates the SOAP header and body for the Create() method depending on given parameters.
   The Perl code can be found in chapter Perl WSOMU Package

Separating the Create() method in a dedicated package allows you to add additional methods to the package. The package can then be installed separately from the actual client that calls the methods. This example package described in this white paper has been developed for the Create() method only. It has a constructor which instantiates all needed objects and sets the following SOAP::Lite information which you need to provide in every SOAP call:

- Proxy
  $self->{client}->proxy($self->{protocol} . "://" . $self->{server} . ":" . $self->{port} . "/" . $self->{path});
- URI
  $self->{client}->uri('http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd');
- Username and password needed for basic authentication
  $self->{client}->transport->http_request->authorization_basic($self->{username}, $self->{password});
- If you need additional namespaces, you can add them into the <Envelope /> XML tag.
  $self->{serializer}->register_ns($namespaces{$ns}, $ns);

Call the Create() method in your package from the client by using the following statement:

my $result = $client->Create("INCIDENT");

In the WSOMU::WSBASIC->Create() method you must construct the SOAP header and the SOAP body manually. The client hands over all necessary parameters required by the Create() method. After the SOAP header and the SOAP body are created, you can execute the following commands:

```
return $self->{client}->serializer( $self->{serializer} )->readable(1)
    ->call( SOAP::Header->type( 'xml' => $args{header} ),
        SOAP::Data->name('Incident')->prefix('ns3') => SOAP::Data->type( 'xml' => $args{body} ) );
```

You passed over the SOAP header data ($args{header}), the SOAP body data ($args{body}) and the type of the SOAP body including its prefix (SOAP::Data->name('Incident')->prefix('ns3')).

The client receives the resulting SOAP envelope. The UUID of the created Incident can be accessed by using XPATH syntax with the following statement:

```
print $result->valueof('//ResourceCreated/ReferenceParameters/SelectorSet/Selector'), "\n";
```

The complete code examples can be found in chapters:

- Perl Client
- Perl WSOMU Package

Figure 3 shows an example how the Perl client can be started on the command line and the resulting incident UUID on an HP-UX system. The command which has been called was the following:

perl -I /opt/perl_32/SOAP-Lite-0.710.08/lib -I /opt/perl_32 create_incident.pl

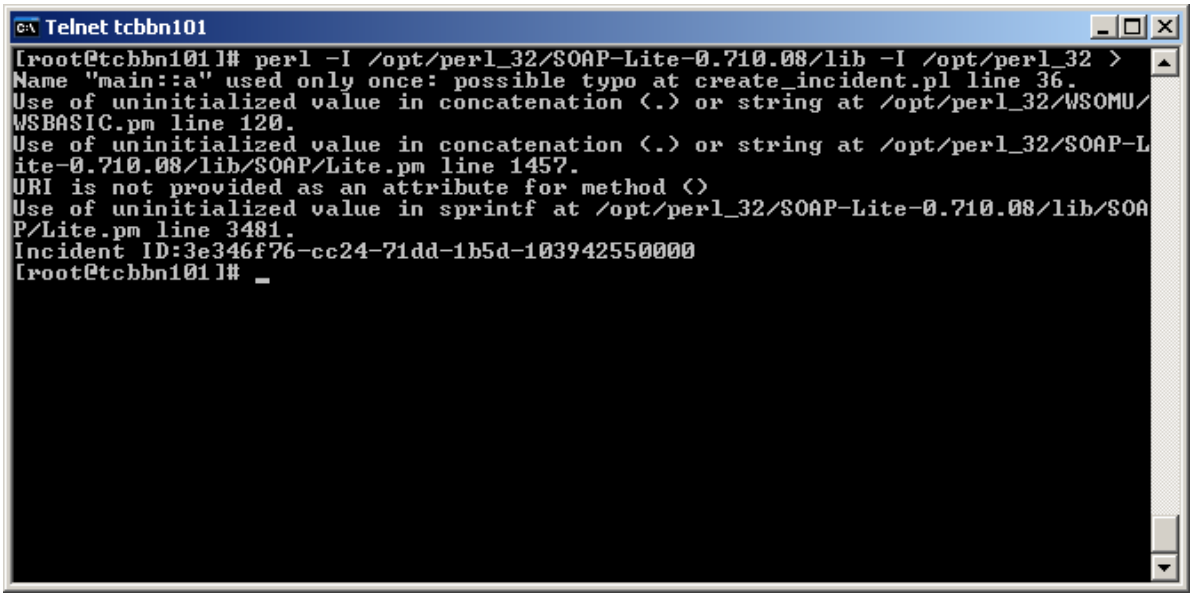

Figure 3: Perl client example on an HP-UX system

Implementing a WSMAN PHP client for the Create() method

After you have installed the PHP development tools mentioned in chapter Components needed for PHP development you might want to use the PHP SOAP library. Unfortunately the PHP SOAP library is not capable of dealing with WSMAN directly. The documentation for the PHP SOAP library can be found at:
http://de.php.net/manual/en/book.soap.php

Because the PHP SOAP library cannot be used, it is necessary to create the SOAP message manually and to send it to the OM Web service using a different mechanism, for example: using CURL (Client for URLs). PHP offers a CURL module that can be used to send the SOAP message to the OM Web service. The PHP CURL documentation can be found at:
http://de.php.net/manual/en/book.curl.php

Because all the necessary modules are included in the Eclipse PDT plug-in, you can start immediately to write the PHP code when you have installed Eclipse and PDT. Because you might want to test your PHP files directly on a web server, you can configure a synchronization directory by using the Aptana Studio Eclipse plug-in. Figure 4 shows an example of a configured synchronization directory.

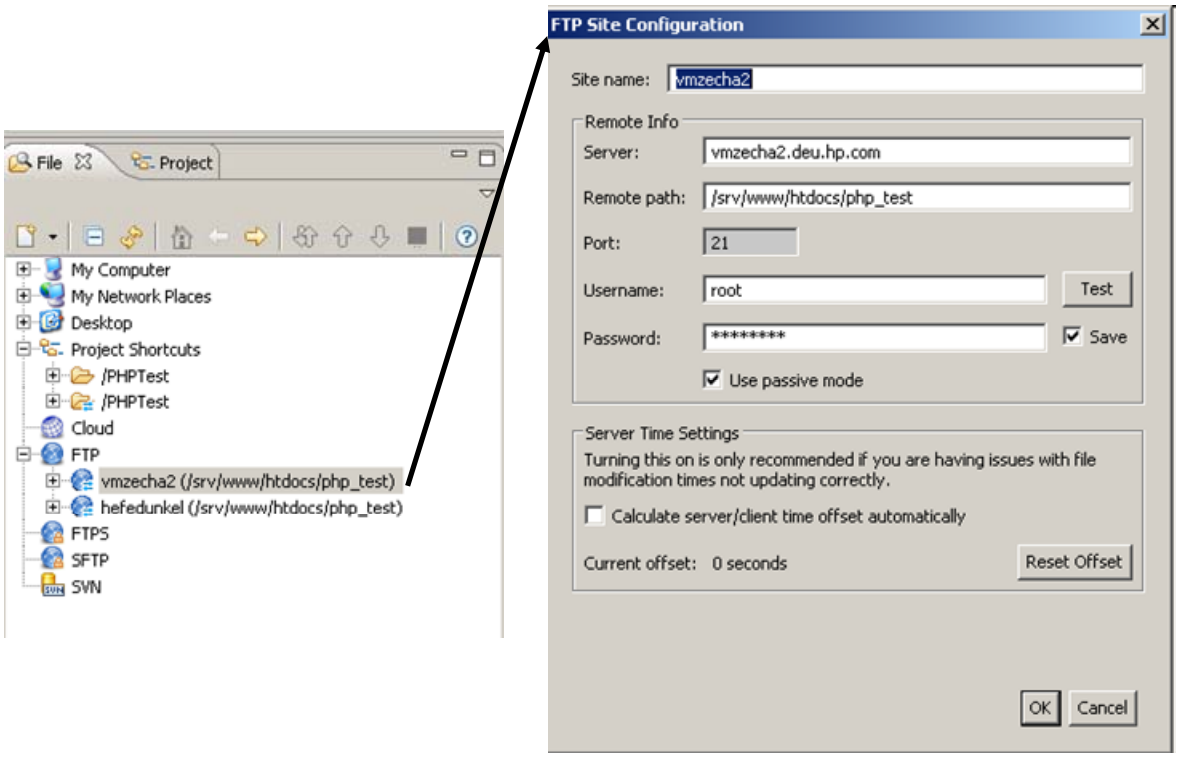

Figure 4: Synchronization Configuration

When you want to upload your PHP file to your server, all you need to do is select the desired file, click the right mouse button, choose 'Synchronize' and select the appropriate synchronization destination.

Initializing the CURL module

Because you use CURL both to send a SOAP message to the OM Web service and receive a SOAP message from the OM Web service, you must initialize the CURL module. The following lines of code initialize the CURL module and set necessary user and password information to access the OM Web service:

```
$omuser = "opc_adm";
$ompassword = "passwd";
$webservice_url = "http://tcbbn085.deu.hp.com:8081/opr-webservice/Incident.svc";
$CurlHandle = curl_init();
curl_setopt($CurlHandle, CURLOPT_URL,            $webservice_url);
curl_setopt($CurlHandle, CURLOPT_POST,            TRUE);
curl_setopt($CurlHandle, CURLOPT_RETURNTRANSFER, TRUE);
curl_setopt($CurlHandle, CURLOPT_USERPWD, $omuser.":".$ompassword);
curl_setopt($CurlHandle, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($CurlHandle, CURLOPT_SSL_VERIFYHOST, FALSE);
```

Detailed descriptions regarding the used curl options can be found at:

http://de.php.net/manual/en/book.curl.php.

Because you send a SOAP message to the OM Web service you must set special options inside the HTTP header to ensure the SOAP message will be understood correctly by the OM Web-service server. The following code shows the added HTTP headers:

```
$HTTPHeader = array (
"TE: deflate,gzip;q=0.3",
"Connection: TE, close",
"Accept: text/xml",
"Accept: multipart/*",
"Accept: application/soap",
"Content-Type: application/soap+xml; charset=utf-8",
"SOAPAction: \"#\"");

curl_setopt($CurlHandle, CURLOPT_HTTPHEADER, $HTTPHeader);
```

Creating and sending the SOAP Message

The PHP code in chapter PHP example send_incident.php shows how a SOAP message for the Create() method can be constructed in PHP. The syntax for other methods is explained in the document OMU8.0_IncidentWebService.pdf.

To trace the socket communication you must not specify the OM server address directly. Iinstead, you can specify 'localhost' and the TCP port TcpTrace is listening on. When starting TcpTrace you will be asked on which port TcpTrace will be listening and to which server and port the TCP packages being traced should be forwarded. An example is shown in Figure 2. In addition to the listening port you also specify the destination server and port. In our example this is the OM server and the Port the incident Web service is listening for incoming requests.

Original OM hostname and port: http://tcbbn085.deu.hp.com:8081/

Modified TcpTrace hostname and port: http://localhost:8099/

When you have created your SOAP message in the variable $SOAPMessage you can send it to the OM Incident Web service by using the following PHP function

```
curl_setopt($CurlHandle, CURLOPT_POSTFIELDS, $SOAPMessage);

$SOAPResponse = curl_exec($CurlHandle);
```

The variable $SOAPResponse contains the SOAP message which has been returned from the OM Web service.

To get the returned incident ID you can either use SimpleXML or DOMXML. Chapter PHP example send_incident.php includes examples showing how to retrieve the incident ID for both PHP modules.

The SimpleXML and DOMXML documentation can be found under:

http://de.php.net/manual/en/book.simplexml.php

http://de.php.net/manual/en/book.domxml.php

Using DOMXML you need the following code to retrieve the incident ID:

1) `$dom = domxml_open_mem($SOAPResponse);`
   creates an DOM tree of the XML document contained in the variable $SOAPResponse
2) `$xpath = $dom->xpath_new_context();`
   creates an XPATH parseable object of the DOM tree
3) `xpath_register_ns($xpath, 'ns6', 'http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd');`
   registers the namespace prefix which is expected in the XML node that contains the incident ID
4) `$result = $xpath->xpath_eval('//ns6:Selector');`
   does an XPATH query which should return the attributes and the values of all XML 'Selector' nodes that have the namespace prefix which has been registered in line 3)
5) `$attributes = $result->nodeset[0]->attributes();`
   gets the attributes of the XML node (only one is expected) and stores it in the variable $attributes
6) `echo $attributes[0]->value().": ".$result->nodeset[0]->get_content();`
   Prints out the attribute and the actual node value

In chapter PHP example send_incident.php you will find the code examples described above including some simple error handling.

Writing a simple PHP based incident submittal web dialog

Using the PHP client created above it is very easy to write a web based incident interface. In the chapter PHP example create_incident.php you will find the code to create the interface shown in Figure 5.

The result of the code example in chapter PHP example send_incident.php which is being called from incident submittal interface is shown in Figure 6.

Make sure, that you use UTF-8 encoding for your PHP files to avoid problems with special characters. To use UTF-8 encoding, make sure that your PHP files contain the following line:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

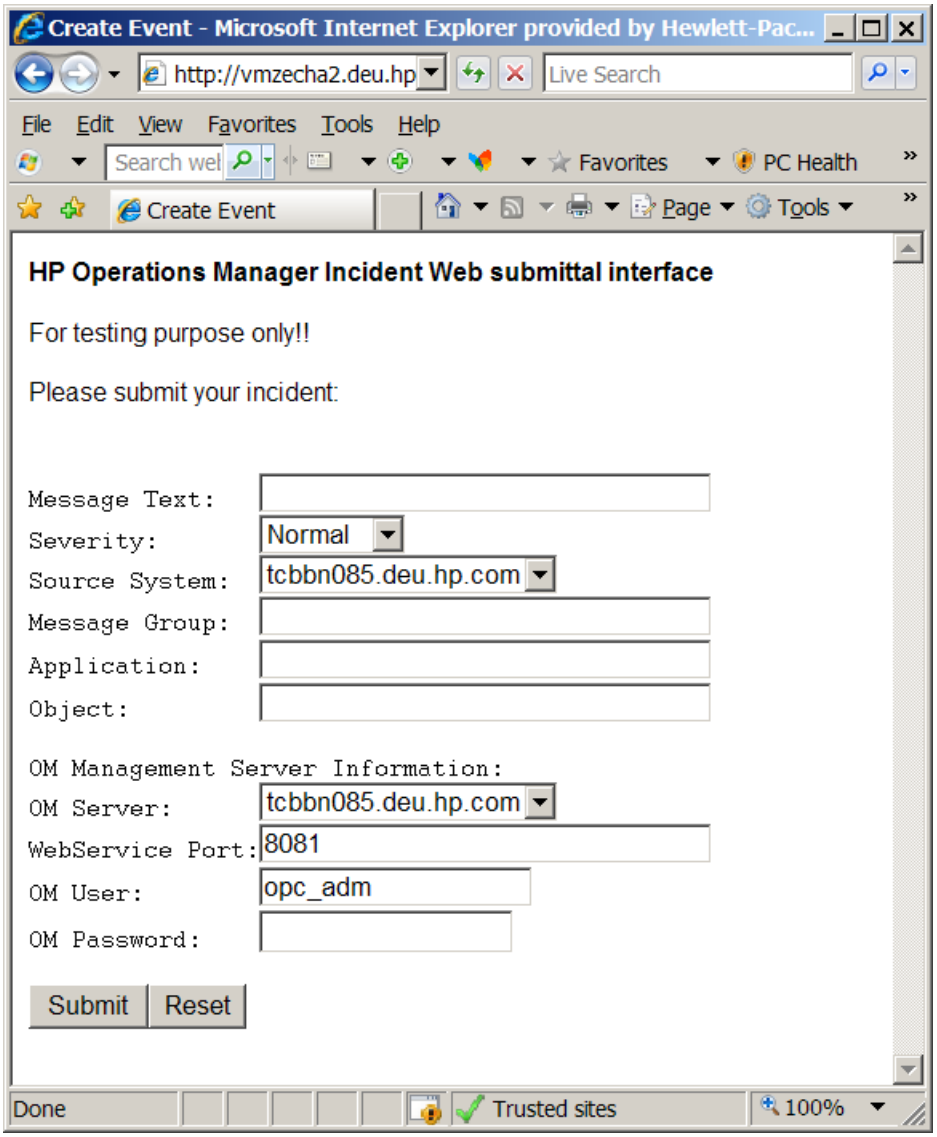

Figure 5: PHP Web Interface

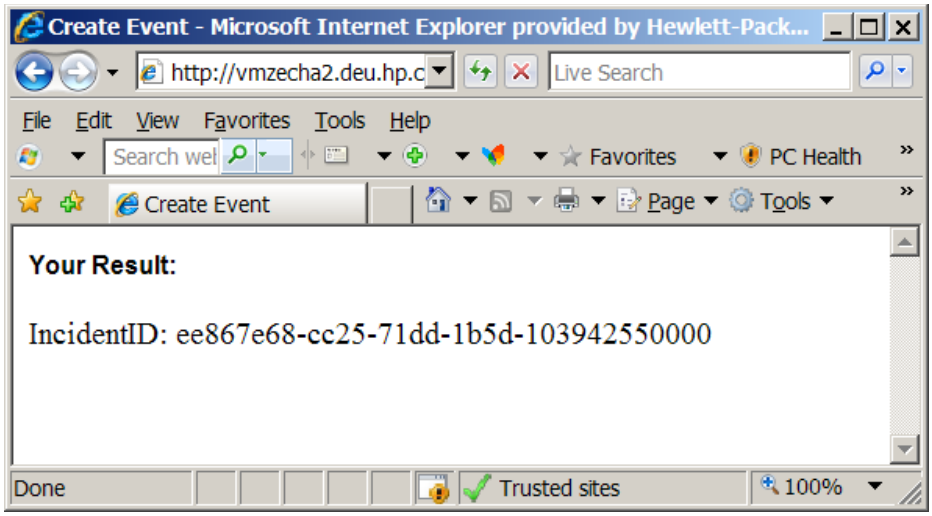

Figure 6: Result

Implementing WSMAN using a Windows VB Script client

Windows Remote Management may be used on Microsoft platforms to create Visual Basic scripts that can access WS Management compliant web services. It is provided by Microsoft on Windows XP, Windows 2003 R2, Windows Vista and Windows 2008. Windows XP and Windows 2003 R2 require Microsoft's WS-Management v1.1 to enable the features needed to communicate with the Wiseman version of WSMAN. See required patches below:

WS-Management v1.1: http://www.microsoft.com/downloads/details.aspx?FamilyID=845289ca-16cc-4c73-8934-dd46b5ed1d33&DisplayLang=en

For general information about winrm see: http://msdn.microsoft.com/en-us/library/aa384426.aspx

For additional tips on Windows Remote Management see:
http://blogs.msdn.com/wmi/archive/2009/01/17/accessing-wmi-data-via-winrm.aspx

WinRM

The Windows Remote Management package provides a command line tool that allows users to easily access WSMAN web services. The command line tool is called winrm. The command line tool is, in itself, a VB script. Users can write their own VB scripts to access the Windows Remote Management services. NOTE: Access to the OMW Incident WS using Windows Remote Management requires version 08.30.008 of the OM Web Services or later.

Help

From the command line help can be obtained on how to use the winrm tool. Just type 'winrm'.Setup

You must first setup the winrm client security on the client:

Allow unencrypted messages:

winrm set winrm/config/client @{AllowUnencrypted="true"}

Allow use of basic authentication:

winrm set winrm/config/client/auth @{Basic="true"}

Set the hosts that are allowed (comma separated list, for example, "localhost,milliways.hp.com,16.57.44.5":

winrm set winrm/config/client @{TrustedHosts="<comma separated host list>"}

Set the MaxEnvelopeSize to a large value if you plan to use Enumeration:

winrm set winrm/config @{MaxEnvelopeSizekb="1000"}

NOTE: This setup will need only be done once on a host. It will survive a reboot.

Incident Enumerate

List all OM Incidents:

winrm enumerate http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident -r:<OM Incident WS address> -a:Basic -username:<username> -password:<password>

OMU Example:

winrm enumerate http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident -r:http://tcivmi17.deu.hp.com:8081/opr-webservice/incident.svc –encoding:utf-8 -a:Basic -username:opc_adm -password:opc_adm

OMW Example:

winrm enumerate http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident -r:http://dmabat3.deu.hp.com/opr-webservice/incident.svc –encoding:utf-8 -a:Basic -username:Administrator -password:password

Incident Get

Get a specific Incident:

winrm get http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident?IncidentID="<OM Incident ID>" -r:<OM Incident WS address> –encoding:utf-8 -a:Basic -username:<username> -password:<password>

For example:

winrm get http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident?IncidentID="ee75154e-f3a6-71dd-1e40-103943920000" -r:http://tcivmi17.deu.hp.com:8081/opr-webservice/incident.svc –encoding:utf-8 -a:Basic -username:opc_adm -password:opc_adm

Incident Create

1. Create an XML file containing the Incident resource you want to create, for example incident.xml contains the following XML:

```
<ns1:Incident
xmlns:ns1="http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
"
xmlns:ns2="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/N
ode">
        <ns1:Title>Test_1234</ns1:Title>
        <ns1:Severity>Major</ns1:Severity>
        <ns1:EmittingNode>
                <ns2:NodeProperties>
                <ns2:DnsName>tcbbn085.deu.hp.com</ns2:DnsName>
                </ns2:NodeProperties>
        </ns1:EmittingNode>
</ns1:Incident>
```

2. Execute the winrm command to create the Incident:

```
winrm create
http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident -r:<OM
Incident WS address> –encoding:utf-8 -a:Basic -username:<username> -
password:<password> -file:incident.xml
```

Incident Subscription

This is used to subscribe to change events on Incidents. After a subscription is created you can then pull the events from the Incident WS. See Incident Pull below. Also a complete script example to subscribe and pull is shown later.

1. Create an XML file to execute the subscription. This is the "document" that will be sent in the body:

```
<wse:Subscribe xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing">
```

2. Execute the following winrm command to create the subscription.
   Output will look something like:
   SubscribeResponse
   SubscriptionManager
   Address = <OM Incident WS address>
   ReferenceParameters
   ResourceURI =
   http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
   Identifier = f76a8408-fb03-4776-bf66-44097e2a9c56
   Expires = 2147483647-12-31T23:59:59.999-14:00
   EnumerationContext = f76a8408-fb03-4776-bf66-44097e2a9c56

Incident Subscription Pull

After a subscription is created, you can use that subscription to pull the events from the server. The following call will block until an event occurs, or a timeout happens.

1. Create a pull.xml file with the context returned. pull.xml:
   <wsen:Pull xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration">
   <wsen:EnumerationContext>f76a8408-fb03-4776-bf66-
   44097e2a9c56</wsen:EnumerationContext>
   <wsen:MaxElements>5</wsen:MaxElements>
   </wsen:Pull>
2. Loop on the following Winrm call:
   winrm invoke http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull
   http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident -r:<OM Incident WS address> -a:Basic -a:Basic -username:<username> -encoding:utf-8 -password:<password> -file:pull.xml

Each time you make the above call it will print out any Incidents that are new because the last call (maximum 5).

WMI Specific

Enable remote access on the server machine:

winrm quickconfig

winrm set winrm/config/service/Auth @{Basic="true"}

winrm set winrm/config/service @{AllowUnencrypted="true"}

Set the client options:

winrm set winrm/config/client @{AllowUnencrypted="true"}

winrm set winrm/config/client/auth @{Basic="true"}

winrm set winrm/config/client @{TrustedHosts="localhost,hpdma,hpdma.deu.hp.com"}

winrm set winrm/config @{MaxEnvelopeSizekb="1000"}

Enumerate Win32_Service resources:

winrm enumerate http://schemas.microsoft.com/wbem/wsman/1/wmi/root/cimv2/Win32_Service -r:localhost

Example VB Script

In chapter Incident Subscription VB Script a sample VB Script is listed that can be used to subscribe to OM Incident events and print them out on the console as they occur within OM.

Appendix

UUID.xs original and Windows modified code

Original Unix/Linux UUID.xs code

```
#include "EXTERN.h"
#include "perl.h"
#include "XSUB.h"

#include <uuid/uuid.h>

#ifndef SvPV_nolen
# define SvPV_nolen(sv) SvPV(sv, na)
#endif

void do_generate(SV *str)
{
        uuid_t uuid;
        uuid_generate( uuid );
        sv_setpvn(str, uuid, sizeof(uuid));
}

void do_unparse(SV *in, SV * out)
{
        uuid_t uuid;
        char str[37];

        uuid_unparse(SvPV_nolen (in), str);
        sv_setpvn(out, str, 36);
}

int do_parse(SV *in, SV * out)
{
        uuid_t uuid;
        char str[37];
        int rc;
```

Modified Windows UUID.xs code

```
#include "EXTERN.h"
#include "perl.h"
#include "XSUB.h"

#include <Rpc.h>
#include <Rpcdce.h>

#ifndef SvPV_nolen
# define SvPV_nolen(sv) SvPV(sv, na)
#endif

void do_generate(SV *str)
{
        UUID uuid;
        UuidCreate( &uuid );
        sv_setpvn(str, &uuid, sizeof(UUID));
}

void do_unparse(SV *in, SV * out)
{
        unsigned char *str;

        UuidToString(SvPV_nolen (in), &str);
        sv_setpvn(out, str, strlen(str));
}

int do_parse(SV *in, SV * out)
{
        UUID uuid;
        unsigned char *str;
        int rc;
```

```
        rc = uuid_parse(SvPV_nolen(in), uuid);
        if (!rc) {
            sv_setpvn(out, uuid, sizeof(uuid));
    }
        return rc;
}

MODULE = UUID        PACKAGE = UUID

void
generate(str)
        SV * str
        PROTOTYPE: $
        CODE:
        do_generate(str);

void
unparse(in, out)
        SV * in
        SV * out
        PROTOTYPE: $$
        CODE:
        do_unparse(in, out);

int
parse(in, out)
        SV * in
        SV * out
        PROTOTYPE: $$
        CODE:
        RETVAL = do_parse(in, out);
        OUTPUT:
        RETVAL
```

```
        rc = UuidFromString(SvPV_nolen(in),
&uuid);
        if (!rc) {
            sv_setpvn(out, &uuid, sizeof(UUID));
    }
        return rc;
}

MODULE = UUID        PACKAGE = UUID

void
generate(str)
        SV * str
        PROTOTYPE: $
        CODE:
        do_generate(str);

void
unparse(in, out)
        SV * in
        SV * out
        PROTOTYPE: $$
        CODE:
        do_unparse(in, out);

int
parse(in, out)
        SV * in
        SV * out
        PROTOTYPE: $$
        CODE:
        RETVAL = do_parse(in, out);
        OUTPUT:
        RETVAL
```

Create() SOAP TcpTrace example

POST /opr-webservice/Incident.svc HTTP/1.1

```
TE: deflate,gzip;q=0.3
Connection: TE, close
Accept: application/soap+xml
Authorization: Basic b3BjX2FkbTpvdnNhcHNwaQ==
Host: localhost:8099
User-Agent: SOAP::Lite/Perl/0.69
Content-Length: 2662
Content-Type: application/soap+xml; charset=utf-8
SOAPAction: "#"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
xmlns:ns4="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/ConfigurationItem"
 xmlns:ns3="http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident"
 xmlns:ns6="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident"
 soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:ns5="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Node"
 xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
   <soap:Header>
      <wsa:Action soap:mustUnderstand="true">
         http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
      </wsa:Action>
      <wsa:To soap:mustUnderstand="true">
         http://localhost:8099/opr-webservice/
      </wsa:To>
      <wsman:ResourceURI soap:mustUnderstand="false">
         http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
      </wsman:ResourceURI>
      <wsman:OperationTimeout soap:mustUnderstand="false">
         P0DT0H10M0S
      </wsman:OperationTimeout>
```

```
    <wsa:MessageID soap:mustUnderstand="true">
      bc050828-35d1-4fd1-8cca-98ca5637dbe2
    </wsa:MessageID>
    <wsa:ReplyTo soap:mustUnderstand="true">
      <wsa:Address>
        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
  </soap:Header>

  <soap:Body>
    <ns3:Incident>
      <ns3:Title>
        Test Incident 2
      </ns3:Title>
      <ns3:Severity>
        Major
      </ns3:Severity>
      <ns3:Category>
        SAP
      </ns3:Category>
      <ns3:CollaborationMode>
        info
      </ns3:CollaborationMode>
      <ns3:EmittingCI>
        <ns4:ConfigurationItemProperties>
          <ns4:ID>
          </ns4:ID>
        </ns4:ConfigurationItemProperties>
      </ns3:EmittingCI>
      <ns3:EmittingNode>
        <ns5:NodeProperties>
          <ns5:DnsName>
            tcbbn085.deu.hp.com
          </ns5:DnsName>
        </ns5:NodeProperties>
```

```
</ns3:EmittingNode>
<ns3:Extensions>
   <ns6:OperationsExtension>
      <ns6:Application>
         SAP
      </ns6:Application>
      <ns6:Object>
         SAP ERP
      </ns6:Object>
      <ns6:NumberOfDuplicates>
         0
      </ns6:NumberOfDuplicates>
      <ns6:ConditionMatched>
         false
      </ns6:ConditionMatched>
      <ns6:AutomaticActionStatus>
      </ns6:AutomaticActionStatus>
      <ns6:OperatorActionStatus>
         notAvailable
      </ns6:OperatorActionStatus>
      <ns6:EscalationStatus>
         notEscalated
      </ns6:EscalationStatus>
      <ns6:CustomAttributes>
         <ns6:CustomAttribute>
            <ns6:Key>
               customer
            </ns6:Key>
            <ns6:Text>
               Test
            </ns6:Text>
         </ns6:CustomAttribute>
      </ns6:CustomAttributes>
      <ns6:NumberOfAnnotations>
         0
      </ns6:NumberOfAnnotations>
```

```
            </ns6:OperationsExtension>
         </ns3:Extensions>
      </ns3:Incident>
   </soap:Body>
</soap:Envelope>
```

SOAP return

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/soap+xml;charset=utf-8
Content-Length: 3246
Date: Fri, 21 Nov 2008 08:47:43 GMT
Connection: close
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:cItem="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Configur
ationItem"
xmlns:inc="http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident"
xmlns:incChg="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident
/Change"
xmlns:incExt="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident"
xmlns:incFil="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/IncidentFilt
er" xmlns:ismCom="http://schemas.hp.com/ism/Common/1/Common"
xmlns:mdo="http://schemas.wiseman.dev.java.net/metadata/messagetypes"
xmlns:mex="http://schemas.xmlsoap.org/ws/2004/09/mex"
xmlns:nItem="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Node"
xmlns:wItem="http://schemas.hp.com/ism/ServiceOperation/Common/1/WorkItem"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:wsmeta="http://schemas.dmtf.org/wbem/wsman/1/wsman/version1.0.0.a/default-
addressing-model.xsd" xmlns:wxf="http://schemas.xmlsoap.org/ws/2004/09/transfer"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <env:Header>
      <wsa:Action env:mustUnderstand="true">
         http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse
      </wsa:Action>
      <wsa:MessageID env:mustUnderstand="true">
         uuid:fd6b6fee-535a-458a-bd6b-92af05572d73
      </wsa:MessageID>
      <wsa:RelatesTo>
```

```
            ff0da6e5-4083-4702-855c-ea66e6289d6f
        </wsa:RelatesTo>
        <wsa:To env:mustUnderstand="true">
            http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
        </wsa:To>
    </env:Header>
    <env:Body>
        <ns5:ResourceCreated xmlns="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns10="http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident"
xmlns:ns11="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Configur
ationItem"
xmlns:ns12="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Node"
xmlns:ns13="http://schemas.hp.com/ism/Common/1/Common"
xmlns:ns14="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident"
xmlns:ns15="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/C
hange" xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:ns3="http://schemas.xmlsoap.org/ws/2004/08/eventing"
xmlns:ns4="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:ns5="http://schemas.xmlsoap.org/ws/2004/09/transfer"
xmlns:ns6="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:ns7="http://schemas.xmlsoap.org/ws/2004/09/mex"
xmlns:ns8="http://schemas.wiseman.dev.java.net/metadata/messagetypes"
xmlns:ns9="http://schemas.hp.com/ism/ServiceOperation/Common/1/WorkItem">
            <ns2:Address env:mustUnderstand="true">
                http://localhost:8099/opr-webservice/
            </ns2:Address>
            <ns2:ReferenceParameters>
                <ns6:ResourceURI>
                    http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
                </ns6:ResourceURI>
                <ns6:SelectorSet>
                    <ns6:Selector Name="IncidentID">
                        1440008e-b7a9-71dd-162b-103942550000
                    </ns6:Selector>
                </ns6:SelectorSet>
            </ns2:ReferenceParameters>
        </ns5:ResourceCreated>
    </env:Body>
</env:Envelope>
```

AXIS2 generated SOAP envelope example

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
   <soapenv:Header xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
      <axis2ns1:ResourceURI
xmlns:axis2ns1="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
soapenv:mustUnderstand="false">
         http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
      </axis2ns1:ResourceURI>
      <axis2ns3:OperationTimeout
xmlns:axis2ns3="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
soapenv:mustUnderstand="false">
         P0DT0H10M0S
      </axis2ns3:OperationTimeout>
      <wsa:To soapenv:mustUnderstand="true">
         http://localhost:8099/opr-webservice/
      </wsa:To>
      <wsa:ReplyTo soapenv:mustUnderstand="true">
         <wsa:Address>
            http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
         </wsa:Address>
      </wsa:ReplyTo>
      <wsa:MessageID soapenv:mustUnderstand="true">
         urn:uuid:C8F0AD534911BAB6C01227170948967
      </wsa:MessageID>
      <wsa:Action soapenv:mustUnderstand="true">
         http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
      </wsa:Action>
   </soapenv:Header>
   <soapenv:Body>
      <ns3:Incident xmlns:ns8="http://schemas.hp.com/ism/Common/1/Common"
xmlns:ns7="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/Ch
ange"
xmlns:ns6="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident"
xmlns:ns5="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Node"
xmlns:ns4="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Configurat
ionItem"
xmlns:ns3="http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident"
xmlns:ns2="http://schemas.hp.com/ism/ServiceOperation/Common/1/WorkItem"
xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">
         <ns3:Title>
            Test_1234
         </ns3:Title>
         <ns3:Severity>
            Major
         </ns3:Severity>
         <ns3:EmittingNode>
            <ns5:NodeProperties>
               <ns5:DnsName>
                  tcbbn085.deu.hp.com
               </ns5:DnsName>
            </ns5:NodeProperties>
```

```
        </ns3:EmittingNode>
      </ns3:Incident>
   </soapenv:Body>
</soapenv:Envelope>
```

Perl Client create_incident.pl

```
#!/usr/bin/perl -w
use strict;
use UUID;
use WSOMU::WSBASIC;    #Import the module.

my %header_opts = (

        path      => "opr-webservice/Incident.svc",
        username  => "opc_adm",
        password  => "my_password",
        port      => "8081",
        server    => "tcbbn085.deu.hp.com",
        namespace => "wsman",
        timeout   => "PT10M0S",
        protocol  => "http"
);

my %body_opts = (
        title           => "Test Incident 3",
        severity        => "Major",
        category        => "SAP",
        collabmode      => "fyi",
        ci_id           => "",
        dnsname         => "tcbbn085.deu.hp.com",
        application     => "SAP",
        object          => "SAP ERP",
        num_of_dup      => "0",
        condition_ma    => "false",
        auto_action_stat => "notAvailable",
        oper_action_stat => "notAvailable",
        escalation_stat  => "notEscalated",
        cma_k           => "customer",
        cma_t           => "Test",
        num_annotations  => "0"
);

my $client = WSOMU::WSBASIC->new( %header_opts, %body_opts );

my $result = $client->Create("INCIDENT");
print "Incident ID:";
print $result->valueof(
        '//ResourceCreated/ReferenceParameters/SelectorSet/Selector'), "\n";
```

Perl WSOMU Package WSBASIC.pm

```
package WSOMU::WSBASIC;
```

```
use strict;
our ( @ISA, @EXPORT );

#=========================<Imports>=================================#
use Exporter;
use SOAP::Lite;
use UUID;

#=========================</Imports>================================#

#=========================<Defaults>================================#
my %namespaces = (
        'wsa'   => "http://schemas.xmlsoap.org/ws/2004/08/addressing",
        'wsman' => "http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd",
        'soap'  => "http://www.w3.org/2003/05/soap-envelope",
        'ns3'   =>
"http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident",
        'ns4'   =>
"http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/ConfigurationItem",
        'ns5'   =>
"http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Node",
        'ns6'   =>
"http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident",
);

@ISA    = ("Exporter");
@EXPORT = qw(&_get_uuid &Create);

#=========================<Defaults>================================#

#    Constructor.
sub new {
        my ( $class, %header_arg ) = @_;

        my $ns;

        my $self = bless {
                client          => SOAP::Lite->new(),
                serializer      => SOAP::Serializer->new(),
                default_class_ns =>
                  { 'INCIDENT' =>
'http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident' },
                protocol  => $header_arg{protocol},
                server    => $header_arg{server},
                port      => $header_arg{port},
                path      => $header_arg{path},
                username  => $header_arg{username},
                password  => $header_arg{password},
                namespace => $header_arg{namespace},
                timeout   => $header_arg{timeout},

                title           => $header_arg{title},
                severity        => $header_arg{severity},
```

```
                category        => $header_arg{category},
                collabmode      => $header_arg{collabmode},
                ci_id           => $header_arg{ci_id},
                dnsname         => $header_arg{dnsname},
                application     => $header_arg{application},
                object          => $header_arg{object},
                num_of_dup      => $header_arg{num_of_dup},
                condition_ma    => $header_arg{condition_ma},
                auto_action_stat => $header_arg{auto_action_stat},
                oper_action_stat => $header_arg{oper_action_stat},
                escalation_stat  => $header_arg{escalation_stat},
                cma_k           => $header_arg{cma_k},
                cma_t           => $header_arg{cma_t},
                num_annotations  => $header_arg{num_annotations}
        }, $class;

        $self->{client}->proxy( $self->{protocol} . "://" . $self->{server} . ":" . $self->{port} . "/" .
$self->{path} );

        $self->{client}->uri('http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd');

        $self->{client}->transport->http_request->header( 'Content-Type' => 'application/soap+xml' );

        $self->{client}->transport->http_request->header( 'Accept' => 'application/soap+xml' );

        $self->{client}->transport->http_request->authorization_basic( $self->{username}, $self-
>{password} );

        foreach $ns ( keys %namespaces ) {
                $self->{serializer}->register_ns( $namespaces{$ns}, $ns );
        }
        return $self;
}

#Method to generate UUIDs.
sub _get_uuid {
        my ( $uuid, $string );
        UUID::generate($uuid);
        UUID::unparse( $uuid, $string );
        return $string;
}

#Generic Create method
sub Create {
        my ( $self, $class_name ) = @_;
        my %args;

        $args{uuid} = _get_uuid;
        if ( !$args{header} ) {
                $args{header} =
                   '<wsa:Action soap:mustUnderstand="true">'
                  . 'http://schemas.xmlsoap.org/ws/2004/09/transfer/Create'
                  . '</wsa:Action>'
```

```
                . '<wsa:To soap:mustUnderstand="true">' . $self->{protocol} . '://'
                . $self->{server} . ":"
                . $self->{port} . "/"
                . $self->{path}
                . '</wsa:To>'
                . '<wsman:ResourceURI soap:mustUnderstand="false">'
                . $self->{default_class_ns}{$class_name}
                . '</wsman:ResourceURI>'
                . '<wsman:OperationTimeout soap:mustUnderstand="false">'
                . $self->{timeout}
                . '</wsman:OperationTimeout>'
                . '<wsa:MessageID soap:mustUnderstand="true">'
                . $args{uuid}
                . '</wsa:MessageID>'
                . '<wsa:ReplyTo soap:mustUnderstand="true">'
                . '<wsa:Address>'
                . 'http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous'
                . '</wsa:Address>'
                . '</wsa:ReplyTo>';
        }

        if ( !$args{body} ) {
                $args{body} =
                   '<ns3:Title>'
                . $self->{title}
                . '</ns3:Title>'
                . '<ns3:Severity>'
                . $self->{severity}
                . '</ns3:Severity>'
                . '<ns3:Category>'
                . $self->{category}
                . '</ns3:Category>'
                . '<ns3:CollaborationMode>'
                . $self->{collabmode}
                . '</ns3:CollaborationMode>'
                . '<ns3:EmittingCI><ns4:ConfigurationItemProperties><ns4:ID>'
                . $self->{ci_id}
                . '</ns4:ID></ns4:ConfigurationItemProperties></ns3:EmittingCI>'
                . '<ns3:EmittingNode><ns5:NodeProperties><ns5:DnsName>'
                . $self->{dnsname}
                . '</ns5:DnsName></ns5:NodeProperties></ns3:EmittingNode>'
                . '<ns3:Extensions><ns6:OperationsExtension>'
                . '<ns6:Application>'
                . $self->{application}
                . '</ns6:Application>'
                . '<ns6:Object>'
                . $self->{object}
                . '</ns6:Object>'
                . '<ns6:NumberOfDuplicates>'
                . $self->{num_of_dup}
                . '</ns6:NumberOfDuplicates>'
                . '<ns6:ConditionMatched>'
                . $self->{condition_ma}
```

```
                        . '</ns6:ConditionMatched>'
                        . '<ns6:AutomaticActionStatus>'
                        . $self->{auto_action_stat}
                        . '</ns6:AutomaticActionStatus>'
                        . '<ns6:OperatorActionStatus>'
                        . $self->{oper_action_stat}
                        . '</ns6:OperatorActionStatus>'
                        . '<ns6:EscalationStatus>'
                        . $self->{escalation_stat}
                        . '</ns6:EscalationStatus>'
                        . '<ns6:CustomAttributes><ns6:CustomAttribute>'
                        . '<ns6:Key>'
                        . $self->{cma_k}
                        . '</ns6:Key>'
                        . '<ns6:Text>'
                        . $self->{cma_t}
                        . '</ns6:Text>'
                        . '</ns6:CustomAttribute></ns6:CustomAttributes>'
                        . '<ns6:NumberOfAnnotations>'
                        . $self->{num_annotations}
                        . '</ns6:NumberOfAnnotations>'
                        . '</ns6:OperationsExtension></ns3:Extensions>';
        }

        return $self->{client}->serializer( $self->{serializer} )->readable(1)
          ->call( SOAP::Header->type( 'xml' => $args{header} ),
                  SOAP::Data->name('Incident')->prefix('ns3') => SOAP::Data->type( 'xml' =>
$args{body} ) );

}
1;
```

PHP example create_incident.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Create Event</title>
<style type="text/css">
<!--
.style1 {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 16px;
        font-weight: normal;
}
.style2 {
        font-family: Courier;
        font-size: 16px;
        font-weight: normal;
}
-->
```

```
</style>
</head>
<body>
<pre class="style1"><strong>HP Operations Manager Incident Web submittal
interface</strong><br>
For testing purpose only!!<br>
Please submit your incident:</pre>
<pre><form name="changefw" method="post" action="./send_event.php" class="style2">
Message Text:   <input type="text" size="30" maxlength="256" name="TITLE" value="">
Severity:       <select name="SEVERITY"><option
selected>Normal</option><option>Minor</option><option>Warning</option><option>Major</o
ption><option>Critical</option></select>
Source System:  <input type="text" size="30" maxlength="64" name="SERVER"
value="tcbbn085.deu.hp.com"></input>
Message Group:  <input type="text" size="30" maxlength="16" name="CATEGORY" value="">
Application:    <input type="text" size="30" maxlength="10" name="APPLICATION" value="">
Object:         <input type="text" size="30" maxlength="10" name="OBJECT" value="">

OM Management Server Information:
OM Server:      <select name="OMSERVER"><option
selected>tcbbn085.deu.hp.com</option><option>sekt.deu.hp.com</option><option>zechaholger.d
eu.hp.com</option></select>
WebService Port:<input type="text" size="5" maxlength="5" name="PORT" value="8081">
Protocol:       <select name="PROTOCOL"><option
selected>http</option><option>https</option></select>
OM User:        <input type="text" size="16" maxlength="32" name="OMUSER" value="opc_adm">
OM Password:    <input type="password" size="16" maxlength="32" name="OMPASSWORD"
value="">

<br><button type="submit" name="send">Submit</button><button type="reset"
name="reset">Reset</button>
</form></pre>
</body>
</html>
```

PHP example send_incident.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Create Event</title>
<style type="text/css">
<!--
.style1 {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 16px;
        font-weight: bold;
}

.style2 {
        font-family: Courier;
```

```
        font-size: 16px;
        font-weight: normal;
}
-->
</style>
</head>
<body>
<pre class="style1">Your Result:</pre>
<?php
$title = $_POST["TITLE"];
$severity = $_POST["SEVERITY"];
$server = $_POST["SERVER"];
$category = $_POST["CATEGORY"];
$application = $_POST["APPLICATION"];
$object = $_POST["OBJECT"];
$webservice_url = $_POST["PROTOCOL"]."://".$_POST["OMSERVER"].":".$_POST["PORT"]."/opr-
webservice/Incident.svc";
$omuser = $_POST["OMUSER"];
$ompassword = $_POST["OMPASSWORD"];

$CurlHandle = curl_init();
curl_setopt($CurlHandle, CURLOPT_URL,           $webservice_url);
curl_setopt($CurlHandle, CURLOPT_POST,           TRUE);
curl_setopt($CurlHandle, CURLOPT_RETURNTRANSFER, TRUE);
curl_setopt($CurlHandle, CURLOPT_USERPWD, $omuser.":".$ompassword);
curl_setopt($CurlHandle, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($CurlHandle, CURLOPT_SSL_VERIFYHOST, FALSE);

$SOAPHeader = array (
"TE: deflate,gzip;q=0.3",
"Connection: keep-alive",
"Accept: text/xml",
"Accept: multipart/*",
"Accept: application/soap+xml;charset=utf-8",
"Content-Type: application/soap+xml; charset=utf-8",
"SOAPAction: \"#\"");

curl_setopt($CurlHandle, CURLOPT_HTTPHEADER, $SOAPHeader);

$uuid = uniqid(rand(), true);

$SOAPMessage  = '<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
xmlns:ns4="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Configurat
ionItem"
xmlns:ns3="http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident"
xmlns:ns6="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident"
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns5="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Node"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
<soap:Header> <wsa:Action
soap:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2004/09/transfer/Create</wsa:Acti
on><wsa:To soap:mustUnderstand="true">'.$webservice_url.'</wsa:To><wsman:ResourceURI
soap:mustUnderstand="false">http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/
1/Incident</wsman:ResourceURI><wsman:OperationTimeout
soap:mustUnderstand="false">PT10M0S</wsman:OperationTimeout><wsa:MessageID
soap:mustUnderstand="true">'.$uuid.'</wsa:MessageID><wsa:ReplyTo
soap:mustUnderstand="true"><wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing
/role/anonymous</wsa:Address></wsa:ReplyTo>  </soap:Header>
<soap:Body><ns3:Incident><ns3:Title>'.$title.'</ns3:Title><ns3:Severity>'.$severity.'</ns3:Severity
><ns3:Category>'.$category.'</ns3:Category><ns3:CollaborationMode>fyi</ns3:CollaborationMo
de><ns3:EmittingCI><ns4:ConfigurationItemProperties><ns4:ID></ns4:ID></ns4:ConfigurationItemP
roperties></ns3:EmittingCI><ns3:EmittingNode><ns5:NodeProperties><ns5:DnsName>'.$server.'</
ns5:DnsName></ns5:NodeProperties></ns3:EmittingNode><ns3:Extensions><ns6:OperationsExten
sion><ns6:Application>'.$application.'</ns6:Application><ns6:Object>'.$object.'</ns6:Object><n
s6:NumberOfDuplicates>0</ns6:NumberOfDuplicates><ns6:ConditionMatched>false</ns6:Conditi
onMatched><ns6:OperatorActionStatus>notAvailable</ns6:OperatorActionStatus><ns6:EscalationSt
atus>notEscalated</ns6:EscalationStatus><ns6:CustomAttributes><ns6:CustomAttribute><ns6:Key>cu
stomer</ns6:Key><ns6:Text>Test</ns6:Text></ns6:CustomAttribute></ns6:CustomAttributes><ns6:
NumberOfAnnotations>0</ns6:NumberOfAnnotations><ns6:Source>OM Incident
Webservice</ns6:Source></ns6:OperationsExtension></ns3:Extensions></ns3:Incident></soap:Bo
dy>
</soap:Envelope>';

curl_setopt($CurlHandle, CURLOPT_POSTFIELDS, $SOAPMessage);

if(!($SOAPResponse = curl_exec($CurlHandle)))
{
        print("ERROR: curl_exec - (" . curl_errno($CurlHandle) . ") " . curl_error($CurlHandle));
        return(FALSE);
}

$parser = xml_parser_create ();
if (!xml_parse ($parser, $SOAPResponse, true)) {
        echo "Error during WebService call!";
        echo $SOAPResponse;
        exit;
}

#SimpleXML example
#$xml = new SimpleXMLElement($SOAPResponse);
#$xml->registerXPathNamespace('ns6','http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd');
#$result = $xml->xpath('//ns6:Selector');
#echo $result[0]->attributes().": ".$result[0];

#DOMXML example
$dom = domxml_open_mem($SOAPResponse);
$xpath = $dom->xpath_new_context();
xpath_register_ns($xpath, 'ns6', 'http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd'); // sets
the prefix "pre" for the namespace
$result = $xpath->xpath_eval('//ns6:Selector');
```

```
$attributes = $result->nodeset[0]->attributes();
echo $attributes[0]->value().": ".$result->nodeset[0]->get_content();
?>
</body>
</html>
```

Incident Subscription VB Script

```
' Module Module1


' Define credentials

private const USERNAME = "opc_adm"

private const PASSWORD = "openview"

private const ADDRESS = "http://omhost:8081/opr-webservice/incident.svc"


private const RESOURCE_URI =
"http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident"


private const SUBSCRIBE_ACTION =
"http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe"

private const SUBSCRIBE_MSG = "<wse:Subscribe
xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'><wse:Delivery
Mode='http://schemas.dmtf.org/wbem/wsman/1/wsman/Pull'/></wse:Subscribe>"


private const PULL_ACTION = "http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull"


sub Main()

' Setup stdin, stdout, and stderr variables

Dim stdIn

Dim stdOut

Dim stdErr


set stdIn = WScript.StdIn

set stdErr = WScript.StdErr

set stdOut = WScript.StdOut


' ---------- Main() ----------


Dim wsmanObj

set wsmanObj = CreateObject("WSMAN.Automation")
```

```
' Setup connection options
Dim objConnectionOptions
set objConnectionOptions = wsmanObj.CreateConnectionOptions
Dim sessionFlags
sessionFlags = wsmanObj.SessionFlagNoEncryption OR
wsmanObj.SessionFlagCredUsernamePassword OR wsmanObj.SessionFlagUseBasic
objConnectionOptions.UserName = USERNAME
objConnectionOptions.Password = PASSWORD
sessionFlags = sessionFlags OR wsmanObj.SessionFlagCredUsernamePassword

' Create the session
Dim session
set session = wsmanObj.CreateSession(ADDRESS, sessionFlags, objConnectionOptions)

' Invoke the request
Dim reply
reply = session.invoke(SUBSCRIBE_ACTION, RESOURCE_URI, SUBSCRIBE_MSG)

' Process the XML document
Dim objXMLDoc
set objXMLDoc = CreateObject("Microsoft.XMLDOM")
objXMLDoc.async = False
objXMLDoc.loadXML(reply)

' Get the Enumeration context to use in the Pull
' This is dirty, but namespace handling is not well documented with VB
Dim NodeList
set NodeList = objXMLDoc.documentElement.getElementsByTagName("wsen:EnumerationContext")
Dim context
context = NodeList(0).text
stdOut.WriteLine "Subscribed to WS-Man Resource com.example:type=CacheControl"

' Now loop on pulling the events
Dim pullXml
Dim notifs
Dim objXMLNotif
```

```
Dim NodeListNotifs
While (True)
    stdOut.WriteLine("*** Waiting for Incident changes (maximum wait is 15 minutes) ***")
    pullXml = "<wsen:Pull
xmlns:wsen='http://schemas.xmlsoap.org/ws/2004/09/enumeration'><wsen:EnumerationContext>
" & context &
"</wsen:EnumerationContext><wsen:MaxTime>PT15M0.000S</wsen:MaxTime><wsen:MaxElemen
ts>5</wsen:MaxElements></wsen:Pull>"
    notifs = session.invoke(PULL_ACTION, RESOURCE_URI, pullXml)
    stdOut.WriteLine notifs
WEnd
End Sub
main
' End Module
```

Glossary

HTML	Hypertext Markup Language, the language web pages are written in.
IDE	Integrated Development Environment, a Software platform used for doing software development.
OM	Short cut for the HP Operation Manager platform in general.
OMU	Operations Manager for Unix, HP Software System Management product on HP-UX and SUN Solaris
OMW	Operations Manager for Windows, HP Software System Management product on Microsoft Windows.
Perl	Practical Extraction and Report Language, a platform independent scripting language.
PHP	Personal Home Page, a scripting language for creating dynamic HTML documents.
RPC	Remote Procedure Call, a inter-process communication technology that allows a computer program to cause a subroutine or procedure to execute in another address space.[1]
SOAP	Simple Object Access Protocol, a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. [1]
URI	Uniform Resource Identifier, a compact string of characters used to identify or name a resource on the Internet. [1]
URL	Uniform Resource Locator, is a type of Uniform Resource Identifier (URI) that specifies where an identified resource is available and the mechanism for retrieving it. [1]
UUID	Universally Unique Identifier, is an identifier standard used in software construction, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE). The intent of UUIDs is to enable distributed systems to uniquely identify information without significant central coordination. [1]
VB script	VB script (short for Visual Basic Scripting Edition) is an Active Scripting language, developed by Microsoft. The language's syntax reflects its origins as a limited variation of Microsoft's Visual Basic programming language.
WSDL	Web Services Description Language, is an XML-based language that provides a model for describing Web services. [1]
XML	Extensible Markup Language, is a general-purpose specification for creating custom markup languages. It is classified as an extensible language, because it allows the user to define the mark-up elements. XML's purpose is to aid information systems in sharing structured data, especially via the Internet, to

encode documents, and to serialize data. [1]

1) Glossary definition has being taken from http://www.wikipedia.org

Index

Related Information

- Incident Web Service Integration Guide

available on http://support.openview.hp.com/selfsolve/manuals

product “Operations Manager for UNIX” version “8.0”

or

product “Operations Manager for Windows” version “8.1”.