

# TECHNICAL GUIDE TO WEB SERVICES SOFTWARE

MONDAY, 15 APRIL 2013



Version: 2.00

Reference: ELECMARKDEV-9-551 © 2013 Australian Energy Market Operator Ltd (AEMO). All rights reserved.

## Important Notice

AEMO has prepared this Technical Guide to Web Services Software (Guide) to provide guidance on the use of the Web Services Client Software under the National Gas or Electricity Rules (Rules), as at the date of publication.

## No reliance or warranty

This Guide does not constitute legal or business advice, and should not be relied on as a substitute for obtaining detailed advice about the National Gas or Electricity Law, the Rules or any other applicable laws, procedures or policies. While AEMO has made every effort to ensure the quality of the information in this Guide, neither AEMO, nor any of its employees, agents and consultants make any representation or warranty as to the accuracy, reliability, completeness, currency or suitability for particular purposes of that information.

## Limitation of liability

To the maximum extent permitted by law, AEMO and its advisers, consultants and other contributors to this Guide (or their respective associated companies, businesses, partners, directors, officers or employees) are not liable (whether by reason of negligence or otherwise) for any errors, omissions, defects or misrepresentations in this document, or for any loss or damage suffered by persons who use or rely on the information in it.

## Copyright

Copyright 2013 Australian Energy Market Operator Limited. The material in this publication may be used in accordance with the [copyright permissions](#) on AEMO's website.

## Trademark notices

No trademark notices.

## Documents made obsolete

The release of this document changes any version of the Web Services User Guide and earlier versions of Technical Guide to Web Services Software.

## Distribution

Available to the public.

## Prepared by

IMT Documentation Team

Last update: 15/04/2013 2:48 PM

## Notes

MSATS version 3.0 – software release 46.81. Complies with the energy market systems single web portal interface.

## Further information

For further information, please visit [www.aemo.com.au](http://www.aemo.com.au) or contact:

AEMO Information and Support Hub

Phone: 1300 AEMO 00 (1300 236 600) and follow the prompts.

Email: [supporthub@aemo.com.au](mailto:supporthub@aemo.com.au)

# Contents

<b>1 Introduction</b> .....	<b>1</b>		
1.1 Purpose .....	1		
1.2 Audience .....	1		
1.3 What's in this guide .....	1		
1.4 Related resources .....	2		
<b>2 Context</b> .....	<b>3</b>		
2.1 Web services interface .....	3		
2.2 RESTful architecture .....	3		
2.3 What AEMO's Web Services are for	4		
2.4 How do you use AEMO's Web			
Services .....	4		
2.5 Who can use AEMO's Web Services	5		
2.6 System requirements .....	5		
<b>3 Standards</b> .....	<b>7</b>		
3.1 aseXML .....	7		
3.2 aseXML version .....	7		
3.3 Security and authentication .....	8		
3.4 User access .....	8		
3.5 HTTPS requests .....	8		
3.5.1 URL .....	8		
3.5.2 Headers .....	9		
3.5.3 File size limits .....	9		
3.5.4 Performing a GET request			
from Internet Explorer .....	10		
3.6 HTTPS responses .....	11		
<b>4 Web Services Client Software</b> .....	<b>13</b>		
4.1 Introduction to Web Services			
Client Software .....	13		
4.2 Java Platform software .....	14		
4.3 Microsoft .NET Framework-based			
software .....	15		
4.4 Java Platform web services client			
software setup .....	15		
4.4.1 Java web services client			
software quick start guide .....	15		
4.4.2 Downloading the Java			
web services client software ...	16		
4.4.3 Extracting the Java			
distribution file .....	16		
4.4.4 Configuring the Java			
.properties file .....	18		
4.4.5 Running the Java web			
services client software .....	22		
4.4.6 Creating Java web			
services client software			
instances .....	25		
4.5 Microsoft .NET Framework-based			
web services client software setup .....	25		
4.5.1 .NET Framework-based			
web services client software			
quick start guide .....	25		
4.5.2 Downloading the .NET			
Framework-based web			
services client software .....	26		
4.5.3 Extracting the .NET			
Framework-based distribution			
file .....	26		
4.5.4 Editing the			
WebServicesGet.exe.config file	28		
4.5.5 Editing the			
WebServicesPost.exe.config			
file .....	31		
4.5.6 Running the			
WebServicesGet.exe .....	35		
4.5.7 Running the			
WebServicesPost.exe .....	36		
4.5.8 Creating .NET			
Framework-based application			
instances .....	38		
4.6 Maintenance .....	38		
<b>5 Library Functions</b> .....	<b>39</b>		
5.1 Reference documentation .....	39		
5.2 C4 NMI Master report library			
functions .....	40		
5.2.1 Contents .....	40		
5.2.2 Get C4 NMI Master			
Report .....	40		

5.2.3 Post C4 NMI Master Report .....	41
5.3 MSATS Limits library functions ....	42
5.3.1 Contents .....	42
5.3.2 Get MSATS Limits .....	42
5.3.3 Post MSATS Limits .....	42
5.4 NMI Detail library functions .....	42
5.4.1 Contents .....	42
5.4.2 Get NMI Detail .....	43
5.4.3 Post NMI Detail .....	43
5.5 NMI Discovery library functions ...	44
5.5.1 Contents .....	44
5.5.2 Get NMI Discovery by DPID .....	44
5.5.3 Post NMI Discovery by DPID .....	44
5.5.4 Get NMI Discovery by meter serial .....	45
5.5.5 Post NMI Discovery by meter serial .....	45
5.5.6 Get NMI Discovery by address .....	46
5.5.7 Post NMI Discovery by address .....	48
5.6 NMI Discovery 3 library functions .	48
5.6.1 Contents .....	48
5.6.2 Get NMI Discovery Type 3 .....	48
5.6.3 Post NMI Discovery Type 3 .....	49
5.7 Participant System Status library functions .....	49
5.7.1 Contents .....	49
5.7.2 Get Participant System Status .....	49
5.7.3 Post Participant System Status .....	50
<b>6 Response Codes .....</b>	<b>51</b>
6.1 Contents .....	51
6.2 Console and log file responses .....	51

6.3 Java software response codes .....	52
6.4 Microsoft .NET Framework-based web services client software response codes .....	53
<b>6 Needing help? .....</b>	<b>56</b>
6.5 Authentication errors .....	56
6.6 .NET Framework-based web services client software .....	56
6.7 Other errors .....	57
6.8 AEMO's Information and Support Hub .....	57
6.8.1 Contacting AEMO's Information and Support Hub ..	57
6.8.2 What can I check before requesting IT assistance from AEMO? .....	57
6.8.3 Information to provide AEMO .....	58
<b>6 References .....</b>	<b>59</b>
6.9 Rules, Law, and Government Bodies .....	59
6.10 Oracle .....	59
6.11 AEMO's website .....	59
6.12 Feedback .....	61
<b>6 Index .....</b>	<b>62</b>

## Figures

Figure 2-1: web services infrastructure	5
Figure 3-1: schema version parameter example	7
Figure 3-2: URL example	8
Figure 3-3: Response header example	12
Figure 4-1: AEMO's Web Services Client Software interface to participant gateways	14

Figure 4-2: HTTP Processing section in the sample.properties file	19
Figure 4-3: Figure 5: Logging configuration section in the sample.properties file	22
Figure 4-4: WebServicesGet.exe.config file	29
Figure 4-5: WebServicesPost.exe.config file	32
Figure 6-1: console response example	51
Figure 6-2: log file response example	52

## Tables

Table 3-1: URL parameters	8
Table 3-2: Headers	9
Table 3-3: Response headers	11
Table 4-1: Java .properties file parameters	19
Table 4-2: WebServicesGet.exe config properties for the log4net section	29
Table 4-3: WebServicesGet.exe properties	30
Table 4-4: WebServicesPost.exe config properties for the log4net section	33
Table 4-5: WebServicesPost.exe properties	33
Table 6-1: Java software response codes	52
Table 6-2: MS .NET Framework-based software response codes	53

## Glossary

These abbreviations, symbols, and special terms assist the reader's understanding of the terms used in this document. For definitions of these terms, the reader should always refer to the applicable market Rules.

### A

#### **AEMC**

Australian Energy Market Commission

#### **API**

Application Programming Interface

#### **aseXML**

A Standard for Energy Transactions in XML. The eXtensible mark-up language standard used by energy companies.

### C

#### **CSV**

Comma-separated values; a file format for exchanging data.

### E

#### **EMMS**

Electricity Market Management System; software, hardware, network and related processes to implement the wholesale National Electricity Market (NEM).

#### **energy market systems web portal**

Single web portal interface to access AEMO's IT systems.

### H

#### **HTTP**

Hypertext Transfer Protocol

#### **HTTPS**

Hypertext Transfer Protocol Secure. A combination of the Hypertext Transfer Protocol (HTTP) with SSL/TLS protocol to provide encrypted communication and secure identification.

### J

#### **JRE**

Java Runtime Environment

---

**M**

**MarketNet**

AEMO's secure private data network connection.

**MSATS**

Market Settlements And Transfer Solution; software, hardware, network and related processes to implement the retail national electricity market (NEM).

---

**N**

**NEM**

National Electricity Market

**NER**

National Electricity Rules

**NGR**

National Gas Rules

---

**R**

**REST**

Representational State Transfer

**RESTful**

Web services designed using the Representational State Transfer (REST) paradigm.

**Rules**

National electricity or gas rules.

---

**S**

**SOAP**

Simple Object Access Protocol

---

**T**

**TLS/SSL**

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are protocols providing communication security over the Internet.

---

**U**

**URI**

Unified Resource Identifier

**URM**

User Rights Management for AEMO's participant systems.

---

**W**

**WSDL**

Web Services Description Language

---

**X**

**XML**

Extensible Mark-up Language

---

**Z**

**ZIP**

Files containing business data with the filename extensions .zip are compressed. They usually contain the XML coded message data.

# 1 Introduction

In this chapter:

---

<b>1.1 Purpose</b> .....	<b>1</b>
<b>1.2 Audience</b> .....	<b>1</b>
<b>1.3 What's in this guide</b> .....	<b>1</b>
<b>1.4 Related resources</b> .....	<b>2</b>

## 1.1 Purpose

---

This technical guide explains the setup and use of AEMO's Web Services Client Software.

## 1.2 Audience

---

This document is intended for participants' technical and software development staff, responsible for implementing AEMO's systems.

## 1.3 What's in this guide

---

This technical guide describes the:

- Context and standards of AEMO's Web Services.
- Available AEMO Web Services.
- Available Web Services Client Software.
- Setup of the Web Services Client Software.
- Details of the library functions used in the web services client software.
- Details of the web service response codes.
- How to ask AEMO for IT assistance.

This document assumes you have knowledge of:

- The Java™ Platform version 7 and above or the Microsoft.NET. Framework 4® environments.
- The operating system you are using.
- AEMO's systems and how they operate from an external perspective.
- The extensible mark-up language (XML).

## 1.4 Related resources

---

As well as the resources listed in "References" on page 59 the following resources may be useful:

- [Guide to Web Services.](#)
- [RESTful Web Services: The basics.](#)
- [Guide to Transition of aseXML](#)

[Text in this format](#) indicates a direct hyperlink with details of the resource listed in "References" on page 59.



## 2 Context

In this chapter:

---

<b>2.1 Web services interface</b> .....	<b>3</b>
<b>2.2 RESTful architecture</b> .....	<b>3</b>
<b>2.3 What AEMO's Web Services are for</b> .....	<b>4</b>
<b>2.4 How do you use AEMO's Web Services</b> .....	<b>4</b>
<b>2.5 Who can use AEMO's Web Services</b> .....	<b>5</b>
<b>2.6 System requirements</b> .....	<b>5</b>

### 2.1 Web services interface

---

In addition to AEMO’s web portals, and batch or file interface options for system-to-system interaction, AEMO’s Web Services offers a new layer of interaction. It takes advantage of existing architecture and is extensible into a larger set of web services, applying to both wholesale and retail systems. The approach, built on top of aseXML, provides for comprehensive coverage of AEMO’s systems into the future.

AEMO’s Web Services use existing messaging standards such as aseXML and CSV, maintaining maximum flexibility and consistency for participants who are free to specify data formats or payloads that suit the target system.

The market systems standard is the transfer of aseXML documents between participant gateways and the market systems. As the full MSATS interface is based on existing aseXML standard documents, the addition of a web service only requires the transfer of such documents. As an extension, the web services are enhanced with some URI parameterised requests to help in the development of gateway interfaces.

### 2.2 RESTful architecture

---

AEMO chose RESTful (REST) for its web services architecture because of its lightweight nature and ability to transmit data directly over HTTP—REST is an alternative to SOAP and WSDL.

The REST architecture makes it possible to start small, developing what is required with available resources, and scaling up as the number of services increase. The REST approach uses the features of HTTP to make requests and follows these design principles:

- Services are provided using HTTPS over MarketNet.

- HTTP requests are stateless.
- Directory structure-like URIs, for example, `https://<web service host>/<system>/<business_function>`.
- Transfer of XML or JavaScript Object Notation (JSON), or both.

Resources are addressed by mapping to a location within a hierarchy of URIs. For example, the root of the hierarchy might represent the web service application and provide a listing of the resources available. Drilling down one level then provides specific information about a particular resource, and further levels provide data from specific resource records. For participants, the operations performed on resources are mapped to HTTP methods:

HTTP Method	Operation
GET	Retrieve data
POST	Update data, retrieve data

AEMO's goal in implementing a RESTful web services approach is to achieve the following:

- Performance: quality of responsiveness.
- Scalability: many users can simultaneously use the systems.
- Generality: solve a wide variety of problems.
- Simplicity: no complex interactions, easy to prove the system is doing as it is supposed to.
- Modifiability: extensible in the face of new requirements and technologies.

## 2.3 What AEMO's Web Services are for

---

AEMO's Web Services offers a new layer of interaction with AEMO's systems. It is an additional option to AEMO's web portals, and batch or file interfaces for system-to-system interaction with AEMO's systems.

## 2.4 How do you use AEMO's Web Services

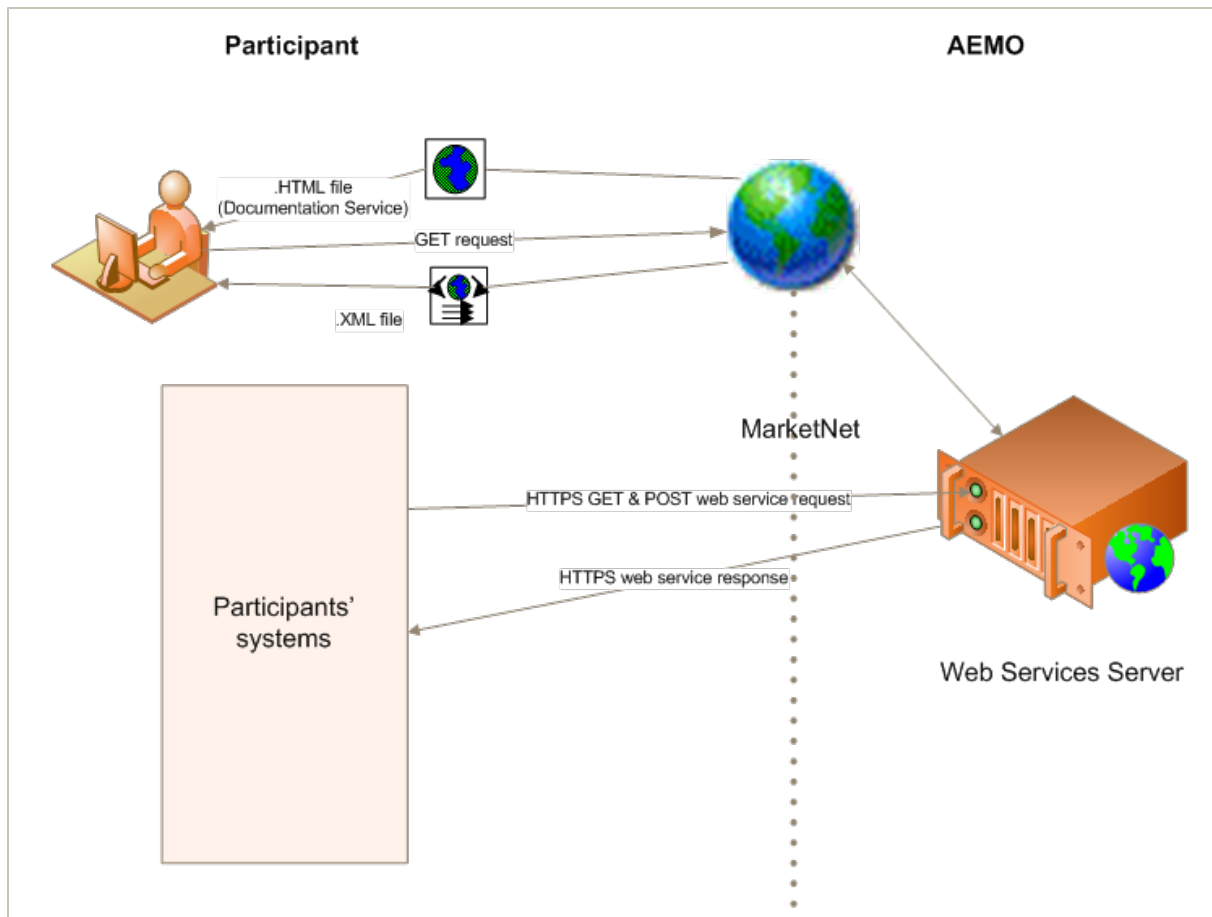
---

AEMO's Web Services provides supported and secure web services to participants' systems using their secure MarketNet connection.

The web service is provided by AEMO's web services server responding to a HTTPS request. The request is stateless, or self-contained, having no dependency on any prior or future request. It contains all data required to both define the request and authenticate the requestor. The response is a block of data sent to the participant's machine making the request. The response is not required to be readable by a person.

Errors are flagged using HTTP response codes, not textual descriptions of the error. A response code of 200 indicates the request was processed successfully. All other response codes indicate the server was unable to process the request, see "Response Codes" on page 51

Figure 2-1: web services infrastructure



## 2.5 Who can use AEMO's Web Services

- Web Services and its software is for use by participants' technical and software development staff, responsible for implementing AEMO's systems.

## 2.6 System requirements

- The supported Internet browser is Microsoft Internet Explorer version 7 or later, although the recommended version is Microsoft Internet Explorer 8.
- AEMO's Web Services are accessed using your MarketNet connection.

- A user ID and password, provided by your system administrator, set up with access to AEMO's Web Services is required.

For participant user access to web services, participant administrators select the appropriate entity in the **Maintain Rights** menu. User accounts and user administration is done in the energy market systems web portal, see the [Guide to User Rights Management](#).

- The Web Services Client Software runs on either the Java™ Platform, Standard Edition 6 (developer version 1.6.0), or the Microsoft .NET Framework 4®.

## 3 Standards

In this chapter:

---

<b>3.1 aseXML</b> .....	<b>7</b>
<b>3.2 aseXML version</b> .....	<b>7</b>
<b>3.3 Security and authentication</b> .....	<b>8</b>
<b>3.4 User access</b> .....	<b>8</b>
<b>3.5 HTTPS requests</b> .....	<b>8</b>
<b>3.6 HTTPS responses</b> .....	<b>11</b>

### 3.1 aseXML

---

For MSATS transactions, the architecture supports the aseXML format. aseXML defines an XML format for data exchange specific to the electricity and gas industries in Australia and is already in use, and well known to participants and AEMO. For more information about aseXML, see [aseXML Standards](#).

### 3.2 aseXML version

---

Most requests produce an aseXML response and clients often require a specific schema version. An optional `aseXML_version` parameter in the `Accept` header meets this requirement. If the requested aseXML version is not supported, a 406 HTTP response code returns.

If the `aseXML_version` is not provided, the participant’s aseXML schema for the current or superseded schema is used. To see your participant schema version, sign in to the MSATS web portal, select Participants and then Participant Schema. For help see the *Guide to MSATS Web Portal* on the [MSATS Participant User Interface Guides](#) web page.

When participants update their aseXML schema version in the MSATS web portal, the change is reflected after 4:00 AM the next day (the schema change for batch services is immediate).

Figure 3-1: schema version parameter example

```

GET https://msats.prod.nemnet.net.au/msats/ws/NMIDetail/NEMMCO?transactionId=TX123&nmi=1234567890&checksum=0 HTTP/1.0
Host: 127.0.0.1
Authorization: Basic dXNlcmlkOnBhc3N3b3Jk
Accept: application/zip; aseXML_version=r25
    
```

### 3.3 Security and authentication

---

To provide encrypted communication and secure identification, interactions between participant systems and AEMO's MarketNet are secured using HTTPS.

### 3.4 User access

---

For access to web services, participant administrators select the relevant entity in the "Maintain Rights" menu and assign the right to their participant users. For help, see [Guide to User Rights Management](#).

### 3.5 HTTPS requests

---

All HTTP requests contain a method, a URL, headers, and optional file attachments. The method is either GET or POST. The HTTP response code 405 returns if the requested method is not supported.

- GET requests require all parameters in the URL, see § 3.5.1 "URL".
- POST requests allow the parameters to exist in an attached file. The file format is either an .XML or a .ZIP file. Posted files have a size limit of 1 MB. If the limit is exceeded, a response code of 413 is returned.
- URL parameters are case sensitive.
- Request character sets are UTF-8 encoded.
- The .XML file must only contain one request.

#### 3.5.1 URL

GET Requests conform to the following URL pattern:

<protocol>://<server>/<application>/ws/<serviceName>/<participantId>?<parameters>

*Figure 3-2: URL example*

```
https://127.0.0.1/msats/ws/NMIDiscovery/NEMMCO?jurisdictionCode=NSW&transactionId=TX123&deliveryPointIdentifier=12345
```

---

Note: URL parameters are case sensitive.

---

*Table 3-1: URL parameters*

URL Parameter	Description
<protocol>	HTTPS

URL Parameter	Description
<server>	Names the server hosting the service or an external proxy. See AEMO's "Interfaces" in <a href="#">Technical Guide to Electricity IT Systems</a> .
<application>	The AEMO system providing the service (for example MSATS or EMMS).
<serviceName>	Names the required web service.
<participantId>	The participant requesting the service. If no <participantId> is provided then HTTP response code 400 returns.
<parameters>	A list of "name=value" pairs, separated by "&".

### 3.5.2 Headers

Table 3-2: Headers

Header	Description
Authorization	userid:password encoded by the Base64 algorithm.  If this header is not provided then HTTP response code 401 returns with the header: WWW-Authenticate: Basic realm="<applicationId>".  If the header is provided but the user credentials cannot be decoded and authenticated, then HTTP response code 403 returns.
Host: <web service host>	URL of the server hosting the service.
Accept: application/zip;	application/zip requests the ase:XML response compressed and returned as a .ZIP file.
Accept: text/xml;	text/xml requests the ase:XML response returned as an XML file.
Accept: application/zip; aseXML_version=r25	Adding aseXML_version=r25 indicates the client is requesting the response as r25 XML schema.  If no schema is specified, the participant's schema version is sent.  If the version specified is not supported, then HTTP response code 406 is returned.
Content-Length: nnn	The length of the attached request file.
Content-Type: text/xml	The format of the attached request file (application/zip or text/xml).

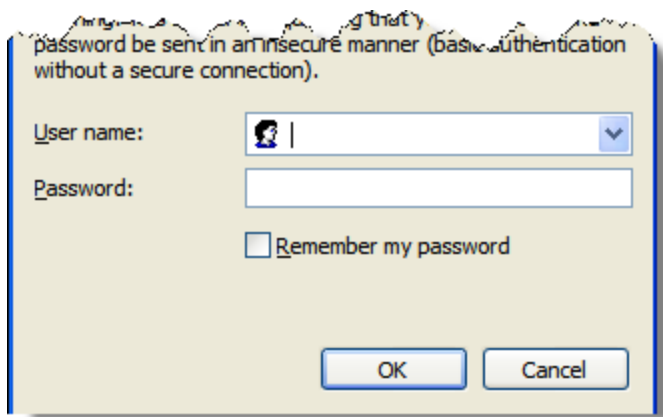
### 3.5.3 File size limits

- Files are limited to a size of 1 MB.

### 3.5.4 Performing a GET request from Internet Explorer

For testing purposes, as well as using the web services client software, GET requests can be made from Internet Explorer, by entering the parameters in the URL. The following is an example of a NMI Discovery, search by meter serial, GET request to the MSATS pre-production web services server.

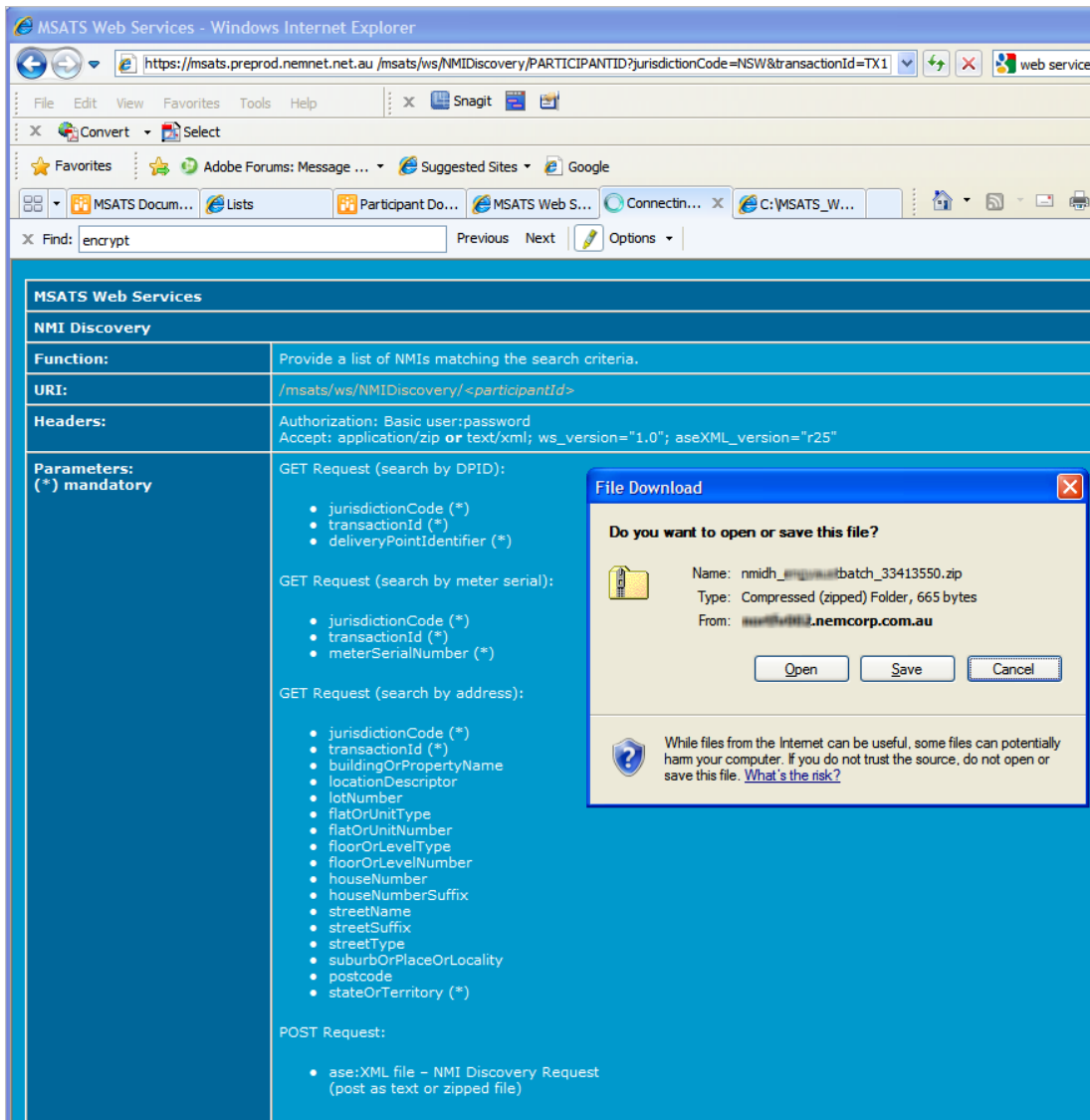
1. Enter the URL including the NMI Discovery parameters in the address bar. Substituting your participant ID for PARTICIPANTID, and the correct parameters for jurisdictionCode, transactionId, and meterSerialNumber. For example:  
https://msats.-  
preprod.nemnet.NET.au/msats/ws/NMIDiscovery/PARTICIPANTID?jurisdictionCode=NSW&transactionId=TX123&meterSerialNumber=12345
2. If the authentication dialog box displays, enter your MSATS user ID and password.



3. If you have entered the GET parameters correctly the **File Download** dialog box displays, where you can **Open** or **Save** the file.

Otherwise, the **MSATS Web Services** web page displays a description of the web service along with the parameter information.





### 3.6 HTTPS responses

A successful request to the web services server, indicated by the HTTP response code 200, returns an .XML file to the client. All other codes indicate a failure, see "Response Codes" on page 51.

The response character set is UTF-8 encoded. The following headers are always returned.

Table 3-3: Response headers

Response Header	Description
HTTP/1.1 and response code	Indicates the HTTP response code (see "Response Codes")
Date:	Current date and time
Server	Web services server

Response Header	Description
Content-Length:	length of response
Cache-Control: no-cache	Response is not cached
Expires:	Response completion date and time
Content-Disposition	Response type and filename
Connection: close	Indicates connection to web services server closed.
Content-Type:	Mime type of response
WWW-Authenticate: Basic realm="<applicationId>"	Returned if the authorization header is not supplied

Figure 3-3: Response header example

```

HTTP/1.1 200 OK
Date: Fri, 05 Aug 2011 05:20:53 GMT
Server: Oracle-Application-Server-10g/10.1.2.0.0 Oracle-HTTP-Server
Content-Length: 5570
Cache-Control: no-cache
Expires: Fri, 05 Aug 2011 05:20:55 GMT
Content-Disposition: attachment; filename=nmidh_userid_33433417.xml
Connection: close
Content-Type: text/xml; charset=UTF-8
    
```

## 4 Web Services Client Software

In this chapter:

---

<b>4.1 Introduction to Web Services Client Software</b> .....	<b>13</b>
<b>4.2 Java Platform software</b> .....	<b>14</b>
<b>4.3 Microsoft .NET Framework-based software</b> .....	<b>15</b>
<b>4.4 Java Platform web services client software setup</b> .....	<b>15</b>
<b>4.5 Microsoft .NET Framework-based web services client software setup</b> .....	<b>25</b>
<b>4.6 Maintenance</b> .....	<b>38</b>

### 4.1 Introduction to Web Services Client Software

---

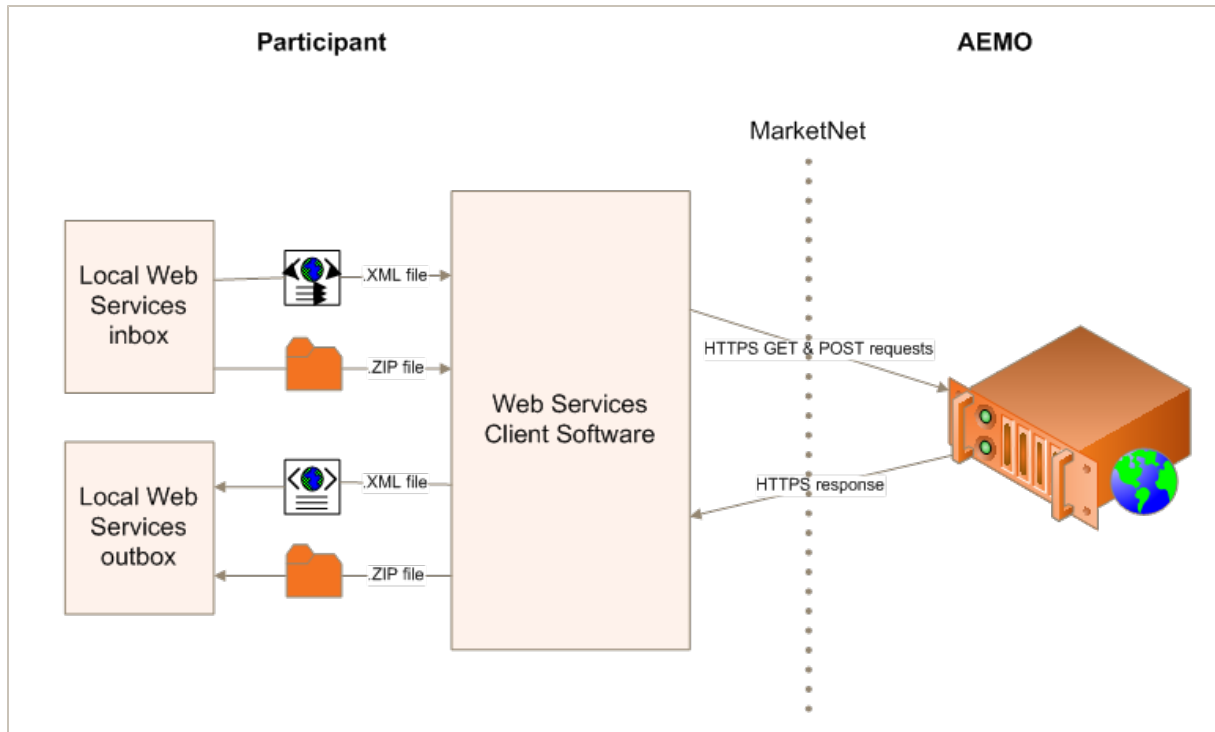
To demonstrate the web services function correctly at participant sites, and to facilitate integration of AEMO’s web services into participant systems, AEMO provides the following web services client software environments:

- Java™ Platform, Standard Edition 7, see 4.2 "Java Platform software" on next page.
- Microsoft .NET Framework 4®, see 4.3 "Microsoft .NET Framework-based software" on page 15.

Participants can use the approach taken in the web services client software to implement their own custom web client into their gateway system.

The software supports participants’ HTTPS protocol requests and operates as a batch process, similar to the *MSATS Participant Batch Software*—not requiring a graphical user interface. Configuration of the software is done in the `.properties` or `.exe.config` files, depending on the software you are using.

Figure 4-1: AEMO's Web Services Client Software interface to participant gateways



Participants can also perform Get requests using their web browser, see § 3.5.4 "Performing a GET request from Internet Explorer".

Download and install one of the following web services client software suited to your participant environment.

## 4.2 Java Platform software

- The `webServices_Java_vn.n.zip` provides a Java interface to invoke web services from within a Java application. The software runs under Java SE 7. Participants require the Java JDK 7 available from [Oracle Downloads](#).

The application takes .ZIP or .XML file requests in the participant's local inbox directory (`inbox`). A successful request places the response files in the participant's local outbox (`outbox`) directory. The Java web services client software can process several web service request types within one instance, for example, NMI Discovery and NMI Detail. For more details, see § 4.4.6 "Creating Java web services client software instances".

To download and install the Java web services client software, see § 4.4 "Java Platform web services client software setup".

## 4.3 Microsoft .NET Framework-based software

---

- The `WebServices_MSdotNet_vn.n.zip` is a .NET Framework-based application used to invoke web services.

The `WebClient.dll` library provides library routines for performing Get and Post requests to the web service and returning the .XML file as a stream.

While running, the `WebServicesPost.exe` application consumes .ZIP and .XML file requests in the participant local inbox (`pli`) directory. When the web service responds, depending on the outcome of the request, the files are moved to either the `pli\Error` or `pli\Done` directories. Successful web service responses are placed in the participant local outbox (`plo`) directory. The `WebServicesPost.exe` only processes one type of web service request (for example, NMI Discovery) within one instance, see § 4.5.8 "Creating .NET Framework-based application instances".

The `WebServicesGet.exe` makes Get requests to the web services server and returns the response to the participant local outbox (`plo`) directory. The Get requests are defined in the `webServicesGet.exe.config` file.

To download and install the sample .NET Framework-based web services client software, see § 4.5 "Microsoft .NET Framework-based web services client software setup".

## 4.4 Java Platform web services client software setup

---

### 4.4.1 Java web services client software quick start guide

The steps required to install, configure, run the web services sample, and convert to a working instance are:

1. Download the latest version of the `WebServices_Java_vn.n.zip` file from [Using Energy Market Information Systems](#).
2. Extract the .ZIP file to `C:\MSATS_WSC` (see § 4.4.3 "Extracting the Java distribution file"). Do not extract the file to a network drive.
3. Configure the `.properties` file in the root directory where you extracted the distribution file (see § 4.4.4 "Configuring the Java .properties file").

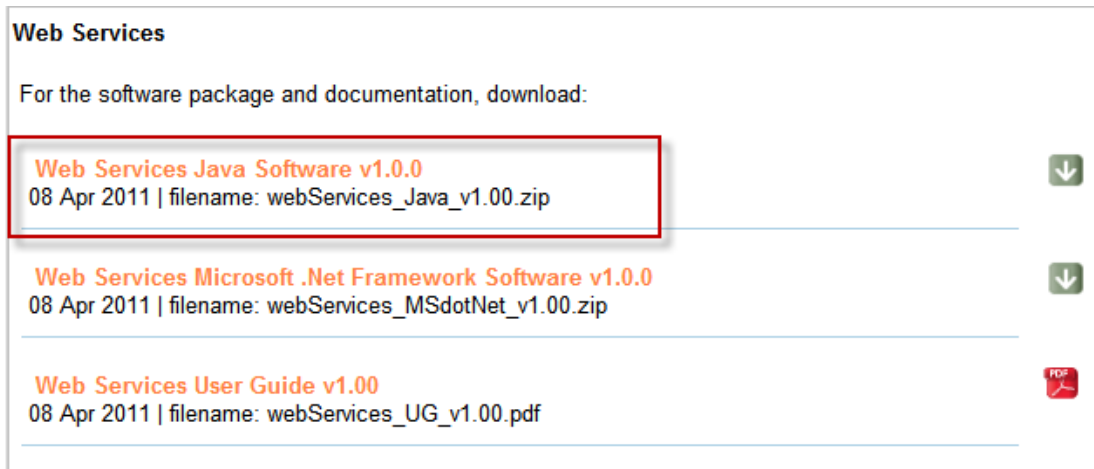
If the file is extracted to the `c:\MSATS_WSC` directory, the sample application (`sample.cmd`) runs and connects to the production web services server with the editing of the following properties in the `sample.properties` file:

- `http.username`
  - `http.password`
  - `participant`
4. Run the `sample.cmd` (see § 4.4.5 "Running the Java web services client software").

5. Create new instances (see § 4.4.6 "Creating Java web services client software instances").
6. Perform regular maintenance to keep the web services client software running smoothly (see § 4.6 "Maintenance").

#### 4.4.2 Downloading the Java web services client software

The latest version of the application is in a single .ZIP file, available from the secure [Using Energy Market Information Systems](#). It looks similar to the following:

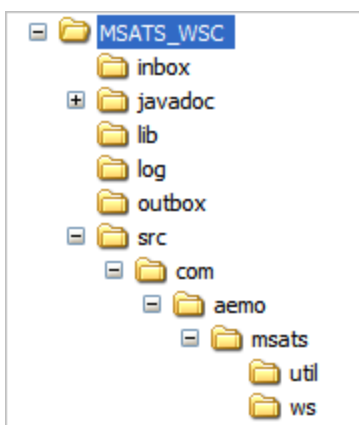


#### 4.4.3 Extracting the Java distribution file

1. Extract the WebServices\_Java\_vn.n.zip file into a directory called C:\MSATS\_WSC.

Extract the file to your C:\ drive only; do not extract it to a network drive.

2. The extraction creates the following directory structure:



3. Contained in the root directory are the following subdirectories and files:

Name	Size	Type
inbox		File Folder
javadoc		File Folder
lib		File Folder
log		File Folder
outbox		File Folder
src		File Folder
README.txt	1 KB	Text Document
sample.cmd	3 KB	Windows NT Command Script
sample.properties	2 KB	PROPERTIES File
setEnv.cmd	1 KB	Windows NT Command Script

#### 4.3.0.1 Java root directory

Contents	Description
inbox	The directory to place your request files in XML or a ZIP format. The distribution file comes with sample requests for each of the available web services.
javadoc	Reference documentation for Java programmers.
lib	All .JAR files.
log	Log and monitor files.
outbox	The directory containing your web service responses.
src	Directory containing all source files, for more details see "Web Services Client Software" on page 13 see "Web Services Client Software" on page 13.
README.txt	Contains the web services version number, list of files, and upgrade information.
sample.cmd	Used to run the sample files included in the distribution package.
sample.properties	Modify this sample properties file to setup your own web services instance. The default location for the JRE is C:/jdk1.6.o_16/jre, if the JRE is installed elsewhere, the setenv.cmd must be amended to match the location.
setEnv.cmd	Sets values for environment variables required by other scripts. The default location for the JRE is C:/jdk1.6.o_16/jre, if the JRE is installed elsewhere, the setenv.cmd must be amended to match the location.

#### 4.3.0.2 Java SRC directory and subdirectories

Contents	Description
ApplicationException.java	This class is a wrapper to any errors that occur during program execution. Details of errors are written to the log file.
Constants.java	This class provides access to static and configurable settings. Configurable settings are loaded by calling the static method setConstants(...).

Contents	Description
ApiTest.java	This class provides a wrapper to access web services. The actual web service requested is controlled by command line arguments. If no arguments are provided a usage message is displays: usage: java com.aemo.msats.ws.ApiTest <properties> <service> <options...>  Run the sample.cmd script to test web services.
NMIDetail.java	Public methods for all NMI details requests.
NMIDiscovery.java	Public methods for all NMI search requests.
WebServices.java	Common functions required for all web requests.

#### 4.4.4 Configuring the Java .properties file

This section describes the properties you are required to change in the `sample.properties` file to have a working web services software set-up. If the file is extracted to `C:\MSATS_WSC`, the sample application (`sample.cmd`) runs and connects to the production web services server with the editing of the following properties in the `sample.properties` file:

- `http.username`
- `http.password`
- `participant`

---

#### Important Notes:

---

- All web services software instances must have a corresponding `.properties` file in the web services root directory.
- A change to the `.properties` file requires an application restart, as the `.properties` file is only read when the application is started.
- For Unix-like systems, the Windows file paths must be converted to Unix paths.
- All file paths in the `.properties` file use forward slashes e.g. `C:/MSATS_WSC/inbox`.



4.4.0.1 HTTP Processing

Figure 4-2: HTTP Processing section in the sample.properties file

```
# AEMO MSATS web services Client Properties File
#
# Please refer to the MSATS Web Services Guide for more details
#
#####
# HTTP Processing
# passwords are either plain or encrypted
#####

#javax.net.debug=all
#javax.net.debug=ssl

#http.protocol=http
http.protocol=https
#http.host=msats.preprod.nemnet.net.au
http.host=msats.prod.nemnet.net.au
#http.port=80
http.port=443

#proxy.host=
#proxy.port=

# value in milliseconds. Zero is interpreted as an infinite timeout.
http.timeout=10000

#http.accept=text/xml; ws_version=1.0; asexML_version=r25
http.accept=application/zip
http.username=myuserid
http.password=mypassword

participant=myParticipantID

keystore.type=jks
keystore=C:/jdk1.6.0_16/jre/lib/security/cacerts
keystore.password=changeit

truststore.type=jks
truststore=C:/jdk1.6.0_16/jre/lib/security/cacerts
truststore.password=changeit

inbox=C:/MSATS_WSC/inbox
outbox=C:/MSATS_WSC/outbox

#####
# Logging configuration section
#
```

Use the property examples in Table 4-1 below to configure your .properties file. Properties with an asterisk (\*) are required.

Table 4-1: Java .properties file parameters

Property	Description	Example
javax.NET.debug	Leave as the default unless debugging your set-up.	javax.NET.debug=all javax.NET.debug=ssl
http.protocol *	Use the secure HTTPS protocol.	http.protocol=https

Property	Description	Example
http.host *	Specifies the AEMO server to connect to.	Pre-production: http.host=msats.preprod.nemnet.NET.au Production: http.host=msats.prod.nemnet.NET.au
http.port	Specify the port number to connect to, if the port number is not specified the application defaults to the correct port number.	http.port=443
proxy.host	Optional - for use with a proxy server only.	#proxy.host=
proxy.port	Only required if proxy.host is specified. For use with a proxy server only.	#proxy.port=
http.timeout *	Adjust the timeout value so under normal conditions it does not produce errors. A value of 10,000 equivalent to 10 seconds is a good starting point.	http.timeout=10000
http.accept *	Requests the format and version of the response. If the accept header is not specified the default response format is returned.	To request the response returned as an XML file: http.accept=text/xml; aseXML_version=r25  To request the response compressed and returned as a ZIP file: http.accept=application/zip
http.username *	Enter your User ID.	http.username=myMarketNetUserID
http.password *	Enter your password.	http.password=myMarketNetPassword
participant *	Enter your MarketNet participant ID.	participant=myParticipantID
keystore.type *	The default keystore type is "jks".	keystore.type=jks
keystore *	If required, change the keystore property to the location of your Java installation's cacerts file.	keystore=C:/jdk1.6.0_16/jre/lib/security/cacerts

Property	Description	Example
<code>keystore.password*</code>	The default Java keystore password is “changeit”. If you have modified yours using the Java “keytool”, enter your modified password.	<code>keystore.password=changeit</code>
<code>truststore.type *</code>	The default truststore type is “jks”.	<code>truststore.type=jks</code>
<code>truststore *</code>	Truststore is the file where the certificates are stored. You can save the file to any location and change the password as long as you configure the truststore property to find file.	<code>truststore=C:/jdk1.6.0_16/jre/lib/security/cacerts</code> Default certificate file stored by Java.
<code>truststore.password *</code>	The default Java truststore password is “changeit”. If you have modified yours using the Java “keytool”, enter your modified password.	<code>truststore.password=changeit</code>
<code>inbox *</code>	Enter the path to your local web services inbox.	<code>inbox=C:/MSATS_WSC/inbox</code>
<code>outbox *</code>	Enter the path to the local web services.	<code>outbox=C:/MSATS_WSC/outbox</code>

#### 4.4.0.2 Logging configuration

The logging properties in this section are for the following logging purposes:

- `ConsoleHandler`: a simple handler for writing formatted records to `System.err`
- `FileHandler`: a handler that writes formatted log records either to a single file, or to a set of rotating log files.

For more details about logging facilities, see

<http://docs.oracle.com/javase/7/docs/api/java/util/logging/package-summary.html>.

- For the sample configuration, leave the logging properties in this section as the default.
- For new instances, change the `java.util.logging.ConsoleHandler.pattern` path to the new location.

Figure 4-3: Figure 5: Logging configuration section in the sample.properties file

```
inbox=C:/MSATS_WSC/inbox
outbox=C:/MSATS_WSC/outbox

#####
# Logging configuration section
#####

# "handlers" specifies a comma separated list of log Handler classes.
handlers=java.util.logging.FileHandler, java.util.logging.ConsoleHandler

# Default global logging level.
.level=CONFIG

# Handler specific properties.
java.util.logging.FileHandler.level=ALL
java.util.logging.FileHandler.formatter=java.util.logging.SimpleFormatter
java.util.logging.FileHandler.limit=100000000
java.util.logging.FileHandler.count=20
java.util.logging.FileHandler.pattern=C:/MSATS_WSC/log/sample_%g.log
java.util.logging.FileHandler.append=true

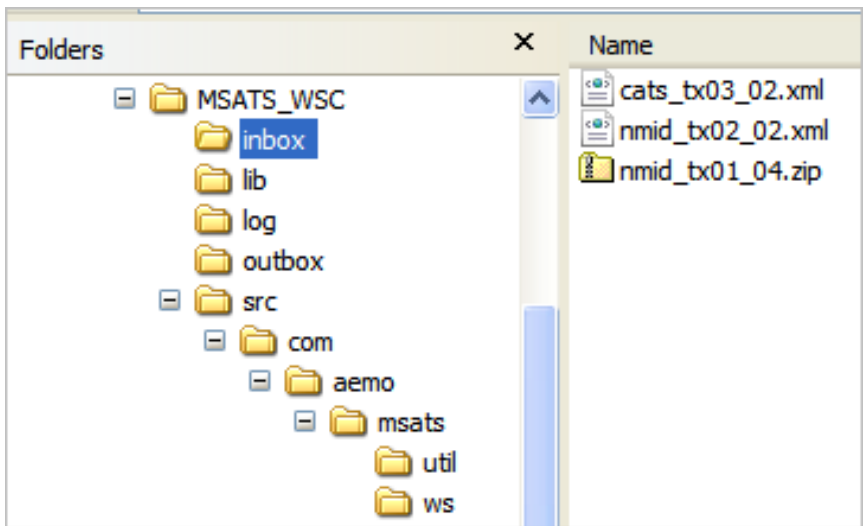
java.util.logging.ConsoleHandler.level=CONFIG
java.util.logging.ConsoleHandler.formatter=java.util.logging.SimpleFormatter

# Logger specific properties.
com.aemo.level=ALL
```

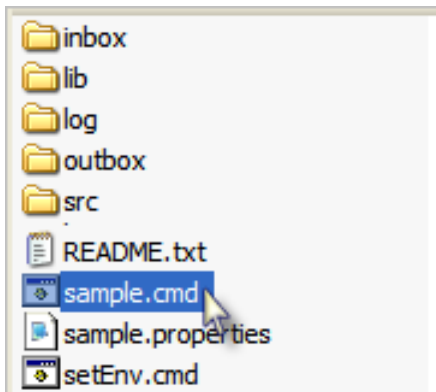
#### 4.4.5 Running the Java web services client software

To run the Java web services client software:

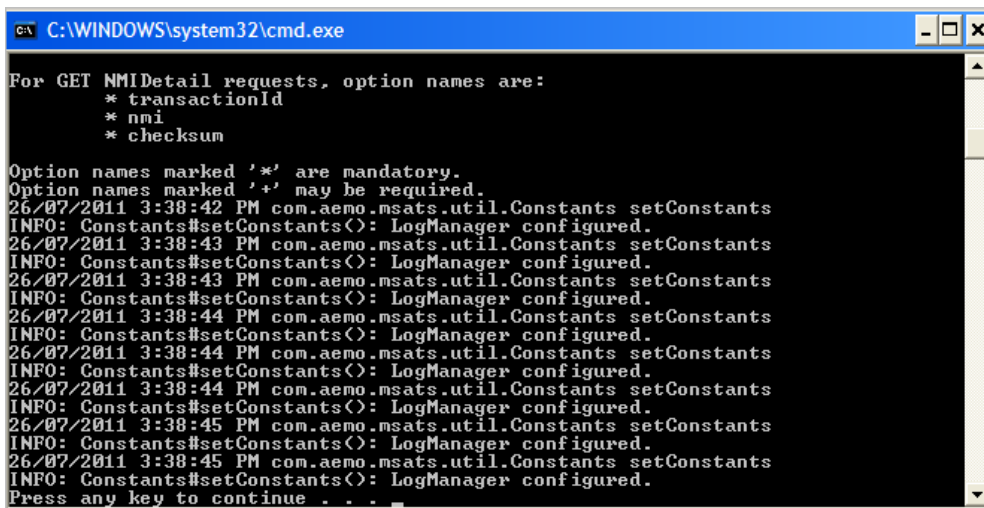
1. The distribution file comes with sample requests for each of the available web services in the inbox. The sample files demonstrate the transaction of .XML or .ZIP files.



2. From the root directory or the command line, run the sample.cmd.



3. The web service script runs, watch for any errors.



4. When the script completes, check the .LOG files in the web services root directory for errors. **Response Code : 200 = OK** indicates a successful request. Other response codes indicate an error, see "Response Codes" on page 51.

Fix any problems and run the `sample.cmd` again until you have a working web services sample, see "Needing help?" on page 56.



#### 4.4.6 Creating Java web services client software instances

Participants can have multiple instances of the Java web services client software in different locations, or in the same location with different `.properties` filenames (for example, one instance for production and one for pre-production). Each instance must have corresponding `.properties` and `web_services.cmd` files in the web services root directory. Each instance is configured differently and can only connect to one AEMO server with one User ID.

To convert the working sample to another instance:

1. Copy the `MSATS_WSC` directory to another location on your participant systems.
2. Configure the new `.properties` file (see § 4.4.4 "Configuring the Java `.properties` file"), for example:
  - In the `http.host` property, specify the environment.
  - In the `http.accept` property, specify the format for responses.
  - Change the `inbox` and `outbox` paths to the new location.
  - Change the Logging configuration path (see § 4.4.0.2 "Logging configuration").
3. Place your `.XML` or `.ZIP` files in the `inbox` directory.
4. Run the web services `.CMD` file.

### 4.5 Microsoft .NET Framework-based web services client software setup

---

#### 4.5.1 .NET Framework-based web services client software quick start guide

The steps required to install, configure, run the web services sample, and convert to a working instance are:

1. Download the latest version of the `WebServices_MSdotNet_vn.n.zip` file from [Using Energy Market Information Systems](#).
2. Extract the `.ZIP` file to `C:\WebServices` (see § 4.5.3 "Extracting the .NET Framework-based distribution file"). Do not extract the file to a network drive.
3. Edit the `WebServicesGet.exe.config` and `WebServicesPost.exe.config` files in the root directory where you extracted the distribution file.

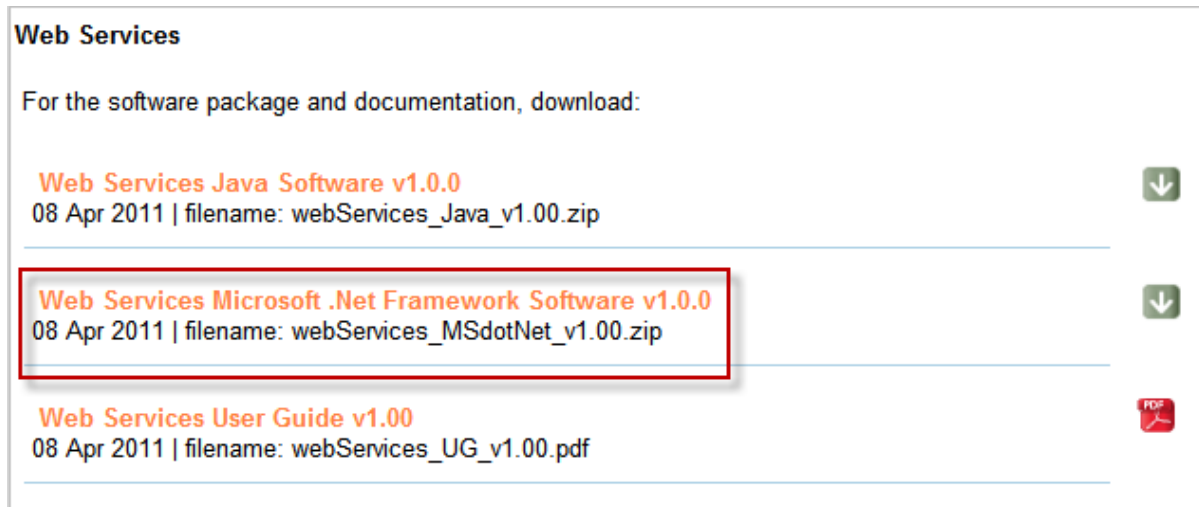
If the distribution file is extracted to the `WebServices` directory, the sample applications, `WebServicesGet.exe` and `WebServicesPost.exe`, run and connect to the pre-production server with the editing of only a few parameters (see § 4.5.4 "Editing the `WebServicesGet.exe.config` file" and see § 4.5.5 "Editing the `WebServicesPost.exe.config` file").

4. Run the `WebServicesPost.exe` or the `WebServicesGet.exe` (see § 4.5.6 "Running the `WebServicesGet.exe`" and see § 4.5.7 "Running the `WebServicesPost.exe`").

5. Create new instances (see § 4.5.8 "Creating .NET Framework-based application instances").
6. Perform regular maintenance to keep the web services client software running smoothly (see § 4.6 "Maintenance").

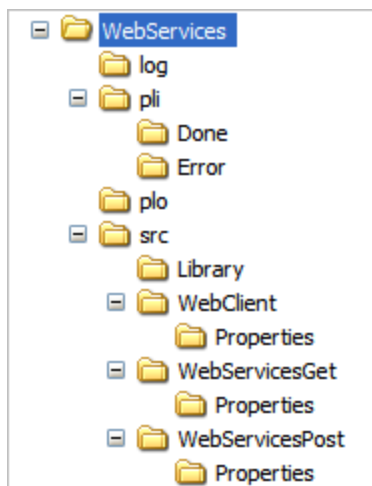
#### 4.5.2 Downloading the .NET Framework-based web services client software

The latest version of the application is in a single .ZIP file, available from [Using Energy Market Information Systems](#). It looks similar to the following:



#### 4.5.3 Extracting the .NET Framework-based distribution file

1. Extract the `webServices_MSdotNet_vn.n.zip` file into `C:\WebServices`.  
Extract the file to your `C:\` drive only; do not extract it to a network drive.
2. The extraction creates the following directory structure:



3. Contained in the root directory are the following subdirectories and files:



Name	Size	Type
log		File Folder
pli		File Folder
plo		File Folder
src		File Folder
log4net.dll	264 KB	Application Extension
log4net.xml	1,334 KB	XML Document
WebClient.dll	17 KB	Application Extension
WebClient.pdb	28 KB	PDB File
WebClient.xml	13 KB	XML Document
WebServices.chm	150 KB	Compiled HTML Help file
WebServicesGet.exe	15 KB	Application
WebServicesGet.exe.config	4 KB	CONFIG File
WebServicesGet.pdb	24 KB	PDB File
WebServicesGet.xml	5 KB	XML Document
WebServicesPost.exe	16 KB	Application
WebServicesPost.exe.config	3 KB	CONFIG File
WebServicesPost.pdb	26 KB	PDB File
WebServicesPost.xml	6 KB	XML Document

### 5.3.0.1 .NET Framework root directory

Contents	Description
log	Log and monitor files.
pli	Participant local inbox where .ZIP and .XML files are placed. The directory contains a Done and an Error sub-directory. The Done directory contains the files that were successfully posted. The Error directory contains the files that failed posting.
plo	Participant local outbox containing your web service responses.
src	Directory containing release notes, all source files, and programming information about the classes, interfaces, and value types, for more details see "Web Services Client Software" on page 13.
log4net.dll	Library of logging routines.
log4net.xml	Appends logs to a database.
webClient.dll	Dynamic link library file for web services Get and Post.
webClient.pdb	Microsoft debug file
webClient.xml	Documentation of interface routines.
webServices.chm	Reference documentation for .NET Framework application programmers. This file must be read from a local drive.
webServicesGet.exe	Sample application to test Get web services.
webServicesGet.exe.config	Modify this sample configuration file to setup your own web services instance.

Contents	Description
webServicesGet.pdb	Microsoft debug file
webServicesGet.xml	Documentation of interface routines.
webServicesPost.exe	Sample application to test Post web services.
webServicesPost.exe.config	Modify this sample configuration file to setup your own web services instance.
webServicesPost.pdb	Microsoft debug file
webServicesPost.xml	Documentation of interface routines.

### 5.3.0.2 .NET Framework SRC directory and subdirectories

Contents	Description
Library	Provides access to the logging library.
webClient	Library of routines for web service Get and Post calls.
webServicesGet	Example of Get routine calls.
webServicesPost	Example of Post routine calls.

### 4.5.4 Editing the webServicesGet.exe.config file

This section describes the properties you are required to change in the `webServicesGet.exe.config` file to have a working Get web services software set-up. If the distribution file is extracted to the `C:\webServices` directory, the sample application (`WebServicesGet.exe`) runs and connects to the pre-production server with the editing of the following properties in the `WebServicesGet.exe.config` file:

- Host
- Username
- Password
- Participant
- Get requests

---

#### Important Notes:

---

- All web services software instances must have a corresponding `webServicesGet.exe.config` file in the web services root directory.
- A change to the `webServicesGet.exe.config` file requires an application restart, as the `.exe.config` file is only read at application start-up.

Figure 4-4: WebServicesGet.exe.config file

```
<?xml version="1.0"?>
<configuration>
  <!-- AEMO Copyright Notice
  // The Copyright of this work is vested in the Australian Energy Market
  // operator and the software is issued in confidence for the purpose only
  // for which it is supplied. It must not be reproduced in whole or in
  // part or used for tendering or manufacturing purposes except under an
  // agreement or with the consent in writing of the Australian Energy
  // Market operator and then only on the condition that this notice is
  // included in any such reproduction. No information as to the contents
  // or subject matter of this software or any part thereof arising directly
  // or indirectly there from shall be given orally or in writing or
  // communicated in any manner whatsoever to any third party being an
  // individual firm or company or any employee thereof without the prior
  // consent in writing of the Australian Energy Market operator.
  // Copyright AEMO Limited. 2011
  -->
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,Log4net"/>
  </configSections>
  <log4net>
    <root>
      <level value="DEBUG"/>
      <appender-ref ref="LogFileAppender"/>
    </root>
    <appender name="LogFileAppender" type="log4net.Appender.RollingFileAppender">
      <param name="File" value="c:\webServices\log\webServicesGet.log"/>
      <param name="AppendToFile" value="true"/>
      <rollingstyle value="Date"/>
      <datePattern value="'.yyyyMMdd'.log'"/>
      <layout type="log4net.Layout.PatternLayout">
        <param name="ConversionPattern" value="%d{dd/MM/yyyy HH:mm:ss}, %-5p, - %m%n"/>
      </layout>
    </appender>
  </log4net>
  <appSettings>
    <add key="Protocol" value="https"/>
    <add key="Host" value="msats.preprod.nemnet.net.au"/>
    <!-- <add key="ProxyAddress" value="192.168.0.104"/> -->
    <!-- <add key="ProxyPort" value="8080"/> -->
    <!-- <add key="Timeout" value="10000"/> -->
    <add key="UserName" value="myUsername"/>
    <add key="Password" value="myPassword"/>
    <!-- this is just a test password! -->
    <add key="Participant" value="myParticipant"/>
    <add key="Accept" value="text/xml"/>
    <!-- this determined the kind of data requested, either xml or zipped_xml -->
    <add key="Plo" value="c:\webServices\plo"/>
    <!-- 5 kinds of GET calls -->
    <add key="DONmiDetail" value="TX123, 1234567890, 0"/>
    <!-- transactionId, Nmi, checksum -->
    <add key="DONmiDiscoveryDpid" value="NSW, TX123, 12345"/>
    <!-- jurisdiction, transactionId, dpid -->
    <add key="DONmiDiscoveryMeterSerial" value="NSW, TX123, 12345"/>
    <!-- jurisdiction, transactionId, meterserialnumber -->
    <!-- jurisdictionCode, transactionId, buildingorPropertyName,
    locationDescriptor, lotNumber, flatOrUnitType, flatOrUnitNumber, floorOrLevelType,
    floorOrLevelNumber, houseNumber, houseNumbersSuffix, streetName, streetsuffix,
    streetType, suburbOrPlaceOrLocality, postcode, stateOrTerritory) -->
    <add key="DONmiDiscoveryAddress" value="NSW, TX123, , , , , , , , GEORGE, , , , 2000, NSW"/>
    <add key="DOMsatsLimits" value="TX123"/>
    <!-- transactionId -->
    <add key="Console" value="true"/>
  </appSettings>
</configuration>
</startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/></startup></configuration>
```

5.4.0.1 webServicesGet.exe.config log4net parameters

If required, edit the parameter value in Table 4-2 below, in the log4net section of the webServicesGet.exe.config file.

Table 4-2: WebServicesGet.exe.config properties for the log4net section

Parameter name	Description	Value example
File *	Enter the path to the directory where you store your log files.	<param name="File" value="c:\webServices\log\webServicesGet.log"/>

5.4.0.2 webServicesGet.exe.config appSettings properties

Edit the properties in [Table 4-3](#) below in the appSettings section of the webServicesGet.exe.config file. Properties with an asterisk (\*) are required.

Table 4-3: WebServicesGet.exe properties

Property	Description	Example
Protocol *	Use the secure HTTPS protocol.	<code>&lt;add key="Protocol" value="https"/&gt;</code>
Host *	Specifies the AEMO server to connect to. Can optionally end in 443 to specify the port number of the web service. If the port number is not specified, the application defaults to the correct port number.	Pre-production: <code>&lt;add key="Host" value="msats.preprod.nemnet.NET.au:443"/&gt;</code>  Production: <code>&lt;add key="Host" value="msats.prod.nemnet.NET.au:443"/&gt;</code>
ProxyAddress	Optional	<code>&lt;add key="ProxyAddress" value="192.168.0.104"/&gt;</code>
ProxyPort	Only needed if ProxyAddress is specified. Default is 8080. Must be an integer between 1 and 65,000.	<code>&lt;add key="ProxyPort" value="8080"/&gt;</code>
Timeout	Adjust the timeout value so under normal conditions it does not produce errors. A value of 10,000 equivalent to 10 seconds is a good starting point. The value must be a positive integer.	<code>&lt;add key="Timeout" value="10000"/&gt;</code>
UserName *	Enter your MSATS User ID.	<code>&lt;add key="UserName" value="yourMarketNetUserID"/&gt;</code>
Password *	Enter your MSATS password.	<code>&lt;add key="Password" value="yourMarketNetPassword"/&gt;</code>
Participant *	Enter your MSATS participant ID. Usually upper case.	<code>&lt;add key="Participant" value="YourParticipantID"/&gt;</code>

Property	Description	Example
Accept *	Determines the format of data requested, either XML or ZIP.	The <code>.exe.config</code> file must contain one of the following accept formats.  To request the response returned as an XML file: <code>&lt;add key="Accept" value="text/xml"&gt;</code>  To request the response compressed and returned as a ZIP file: <code>&lt;add key="Accept" value="application/zip"&gt;</code>
Plo *	Set to the full path of your participant local outbox (plo) directory.	<code>&lt;add key="Plo" value="c:-\webServices\plo"/&gt;</code>
Console	Controls if the application puts log messages on the console window when running - must be a Boolean value.	<code>&lt;add key="Console" value="true"/&gt;</code>
Add one of the five kinds of Get calls:		
DoNmiDetail	NMI Detail	<code>&lt;add key="DoNmiDetail" value="TX123, 1234567890, 0"/&gt;</code>
DoNmiDiscoveryDpid	NMI Discovery by DPID	<code>&lt;add key="DoNmiDiscoveryDpid" value="NSW, TX123, 12345"/&gt;</code>
DoNmiDiscoveryMeterSerial	NMI Discovery by Meter Serial	<code>&lt;add key="DoNmiDiscoveryMeterSerial" value="NSW, TX123, 12345"/&gt;</code>
DoNmiDiscoveryAddress	NMI Discovery by Address	<code>&lt;add key="DoNmiDiscoveryAddress" value="NSW, TX123, , , , , , , , , GEORGE, , , , 2000, NSW"/&gt;</code>
DoMsatsLimits	MSATS Limits	<code>&lt;add key="DoMsatsLimits" value="TX123"/&gt;</code>

#### 4.5.5 Editing the `webServicesPost.exe.config` file

This section describes the properties you are required to change in the `webServicesPost.exe.config` file to have a working Post web services software set-up. If the distribution file is extracted to the `C:\webServices` directory, the sample application (`WebServicesPost.exe`) runs and connects to the pre-production server with the editing of the following properties in the `webServicesPost.exe.config` file:

- Host
- Participant
- Username
- Password

---

Important Notes:

---

- All web services software instances must have a corresponding `WebServicesPost.exe.config` file in the web services root directory.
- A change to the `WebServicesPost.exe.config` file requires an application restart, as the `.exe.config` file is only read at application start-up.

Figure 4-5: `WebServicesPost.exe.config` file

```
<?xml version="1.0"?>
<configuration>
  <!-- AEMO Copyright Notice
  // The Copyright of this work is vested in the Australian Energy Market
  // Operator and the software is issued in confidence for the purpose only
  // for which it is supplied. It must not be reproduced in whole or in
  // part or used for tendering or manufacturing purposes except under an
  // agreement or with the consent in writing of the Australian Energy
  // Market Operator and then only on the condition that this notice is
  // included in any such reproduction. No information as to the contents
  // or subject matter of this software or any part thereof arising directly
  // or indirectly there from shall be given orally or in writing or
  // communicated in any manner whatsoever to any third party being an
  // individual firm or company or any employee thereof without the prior
  // consent in writing of the Australian Energy Market Operator.
  // Copyright AEMO Limited. 2011
  -->
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,Log4net"/>
  </configSections>
  <log4net>
    <root>
      <level value="DEBUG"/>
      <appender-ref ref="LogFileAppender"/>
    </root>
    <appender name="LogFileAppender" type="log4net.Appender.RollingFileAppender">
      <param name="File" value="c:\webServices\log\WebServicesPost.log"/>
      <param name="AppendToFile" value="true"/>
      <rollingStyle value="Date"/>
      <datePattern value=".\'yyyyMMdd\'.log"/>
      <layout type="log4net.Layout.PatternLayout">
        <param name="ConversionPattern" value="%d{dd/MM/yyyy HH:mm:ss}, %-5p, - %m%n"/>
      </layout>
    </appender>
  </log4net>
  <appSettings>
    <add key="Protocol" value="https"/>
    <add key="Host" value="nortfv002.test.nemnet.net.au:443"/>
    <!-- value="10.20.104.139"/ -->
    <add key="Timeout" value="10001"/>
    <!--add key="ProxyAddress" value="10.20.28.74"/>
    <add key="ProxyPort" value="8080"/>
    <add key="Accept" value="text/xml"/>
    <!--; ws_version=1.0; aseXML_version=r25"/>
    <!--text/xml"/ -->
    <!-- application/zip -->
    <add key="Participant" value="NEMMCO"/>
    <add key="Resource" value="/msats/ws/NMIDiscovery/">
    <add key="Pli" value="c:\webServices\pli"/>
    <add key="Plo" value="c:\webServices\plo"/>
    <add key="MaskZip" value="*.zip"/>
    <add key="MaskXml" value="*.xml"/>
    <add key="UserName" value="mikemuir"/>
    <add key="Password" value="Blondie02"/>
    <!-- this is just a test password! -->
    <add key="cycleInSec" value="10"/>
    <add key="console" value="true"/>
  </appSettings>
  <startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/></startup></configuration>
```

### 5.5.0.1 `WebServicesPost.exe.config` log4net parameters

If required, edit the parameter value in Table 4-4 on the facing page in the log4net section of the `WebServicesPost.exe.config` file.

Table 4-4: *WebServicesPost.exe* config properties for the *log4net* section

Parameter name	Description	Value example
File	Enter the path to the directory where you store your log files.	<code>&lt;param name="File" value="c:-\WebServices\log\webServicesPost.log"/&gt;</code>

### 5.5.0.2 *webServicesPost.exe.config* appSettings properties

Edit the properties in Table 4-5 below in the *appSettings* section of the *WebServicesPost.exe.config* file. Key names with an asterisk (\*) are required.

Table 4-5: *WebServicesPost.exe* properties

Property	Description	Example
Protocol *	Use the secure HTTPS protocol.	<code>&lt;add key="Protocol" value="https"/&gt;</code>
Host *	Specifies the AEMO server to connect to. Can optionally end in 443 to specify the port number of the web service. If the port number is not specified, the application defaults to the correct port number.	Pre-production: <code>&lt;add key="Host" value="msats.preprod.nemnet.NET.au:443"/&gt;</code> Production: <code>&lt;add key="Host" value="msats.prod.nemnet.NET.au:443"/&gt;</code>
Timeout	Adjust the timeout value so under normal conditions it does not produce errors. A value of 10,000 equivalent to 10 seconds is a good starting point. The value must be a positive integer.	<code>&lt;add key="Timeout" value="10000"/&gt;</code>
ProxyAddress	Optional	<code>&lt;add key="ProxyAddress" value="10.20.28.74"/&gt;</code>
ProxyPort	Only needed if ProxyAddress is specified. Default is 8080. Must be an integer between 1 and 65,000.	<code>&lt;add key="ProxyPort" value="8080"/&gt;</code>

Property	Description	Example
Accept *	Requests the format and version of the response.	<p>The <code>.exe.config</code> file must contain one of the following accept formats:</p> <p>To request the response returned as an XML file:  <code>&lt;add key="Accept" value="text/xml"&gt;</code></p> <p>To request the response compressed and returned as a ZIP file:  <code>&lt;add key="Accept" value="application/zip"&gt;</code></p>
Participant *	Enter your MSATS participant ID.	<code>&lt;add key="Participant" value="YourParticipantID"/&gt;</code>
Resource *	<p>The resource must start and end with a forward slash (/).</p> <p>If you want to process two types of web service requests, e.g. NMI Discovery and NMI Detail, create two instances of the program with different masks (see <code>MaskZIP</code> and <code>MaskXML</code>) or <code>pli</code> directory to distinguish the input files (see § see "Web Services Client Software").</p>	<code>&lt;add key="Resource" value="/msats/ws/NMIDiscovery/"&gt;</code>
Pli *	Set to the full path of your participant local inbox ( <code>pli</code> ) directory.	<code>&lt;add key="Pli" value="c:-\webServices\pli"/&gt;</code>
Plo *	Set to the full path of your participant local outbox ( <code>plo</code> ) directory.	<code>&lt;add key="Plo" value="c:-\webServices\plo"/&gt;</code>
MaskZip *	Used against the <code>Pli</code> directory to select <code>.ZIP</code> request files for posting.	<code>&lt;add key="MaskZip" value="*.zip"/&gt;</code>
MaskXml *	Used against the <code>Pli</code> directory to select <code>.XML</code> format request files for posting.	<code>&lt;add key="MaskXml" value="*.xml"/&gt;</code>
UserName *	Enter your MSATS User ID.	<code>&lt;add key="UserName" value="yourMarketNetUserID"/&gt;</code>
Password *	Enter your MSATS password.	<code>&lt;add key="Password" value="yourMarketNetPassword"/&gt;</code>
CycleInSec	Controls how long to wait between checks after the <code>Pli</code> directory is empty - must be an integer.	<code>&lt;add key="CycleInSec" value="10"/&gt;</code>
Console	Controls if the application puts log messages on the console window when running - must be a Boolean value.	<code>&lt;add key="Console" value="true"/&gt;</code>



### 4.5.6 Running the WebServicesGet.exe

To run the sample WebServicesGet.exe:

1. From the root directory or the command line, run the `WebServicesGet.exe`.

Be sure to edit the `webServicesGet.exe.config` file first, see § 4.5.4 "Editing the `WebServicesGet.exe.config` file".

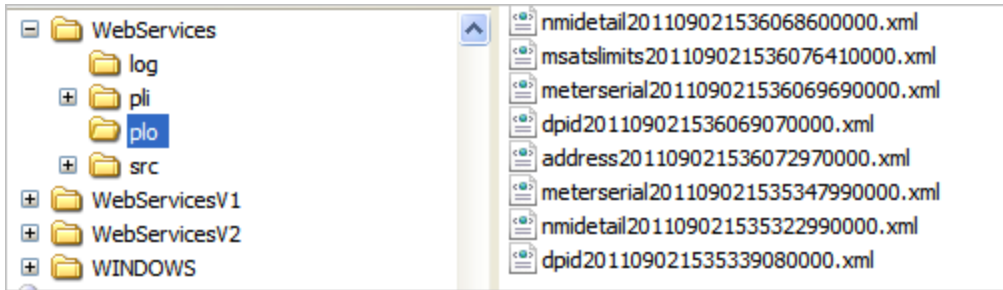
Name	Size	Type
log		File Folder
pli		File Folder
plo		File Folder
src		File Folder
log4net.dll	264 KB	Application Extension
log4net.xml	1,334 KB	XML Document
WebClient.dll	17 KB	Application Extension
WebClient.pdb	28 KB	PDB File
WebClient.xml	13 KB	XML Document
WebServices.chm	150 KB	Compiled HTML Help file
<b>WebServicesGet.exe</b>	15 KB	Application
WebServicesGet.exe.config	4 KB	CONFIG File
WebServicesGet.pdb	24 KB	PDB File
WebServicesGet.xml	5 KB	XML Document
WebServicesPost.exe	16 KB	Application
WebServicesPost.exe.config	3 KB	CONFIG File
WebServicesPost.pdb	26 KB	PDB File
WebServicesPost.xml	6 KB	XML Document

2. The web service script runs, watch for any errors.

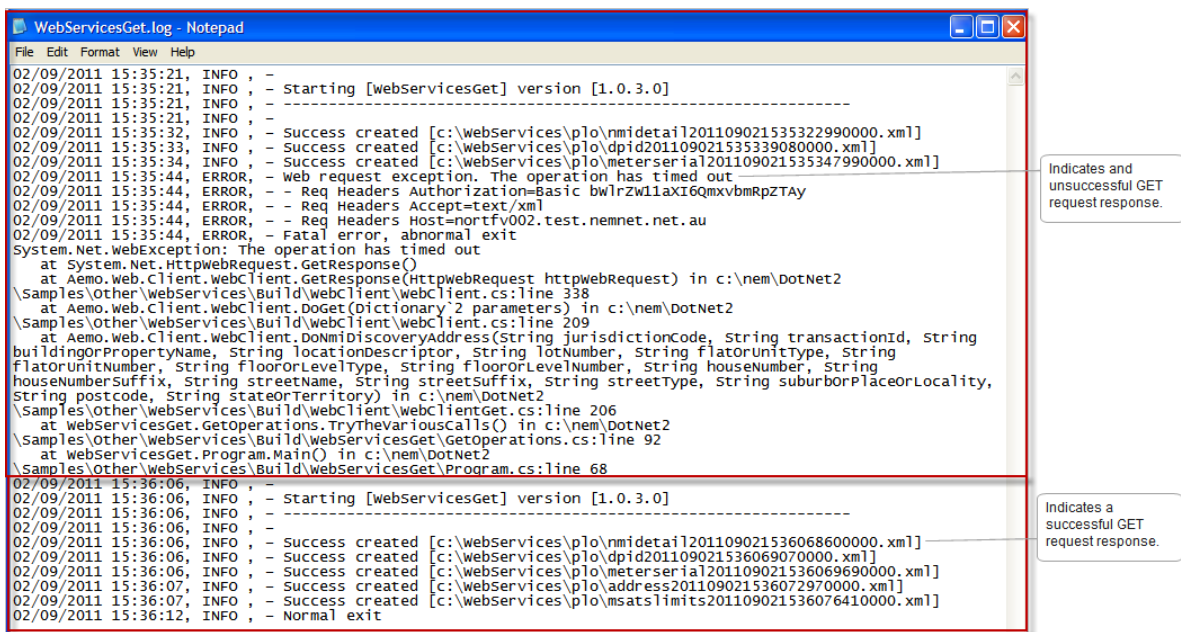
```

C:\WebServices\WebServicesGet.exe
02/09/2011 15:36:06 Created file [c:\WebServices\plo\nmidetail201109021536068600
000.xml]
02/09/2011 15:36:06 Created file [c:\WebServices\plo\dpid201109021536069070000.x
ml]
02/09/2011 15:36:06 Created file [c:\WebServices\plo\meterserial2011090215360696
90000.xml]
02/09/2011 15:36:07 Created file [c:\WebServices\plo\address20110902153607297000
0.xml]
02/09/2011 15:36:07 Created file [c:\WebServices\plo\msatslimits2011090215360764
10000.xml]
Normal exit
  
```

3. When the file is processed, a successful request is indicated by a response created in the participant outbox directory (`plo`).



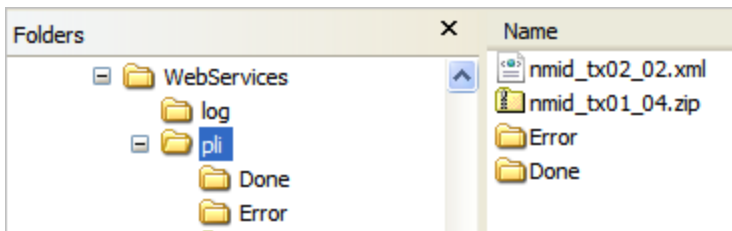
4. Otherwise, check the WebServicesGet.log for any errors. Fix the errors and run the WebServicesGet.exe again until you have a working web services Get sample, see "Response Codes" on page 51 and see "Needing help?" on page 56.



### 4.5.7 Running the WebServicesPost.exe

To run the sample WebServicesPost.exe:

1. Place a NMI Discovery web services request .XML or .ZIP file into the pli directory.

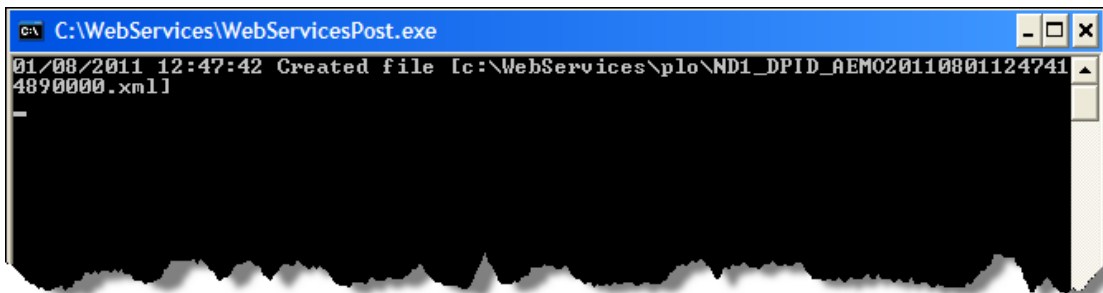


2. From the root directory or the command line, run the WebServicesPost.exe.

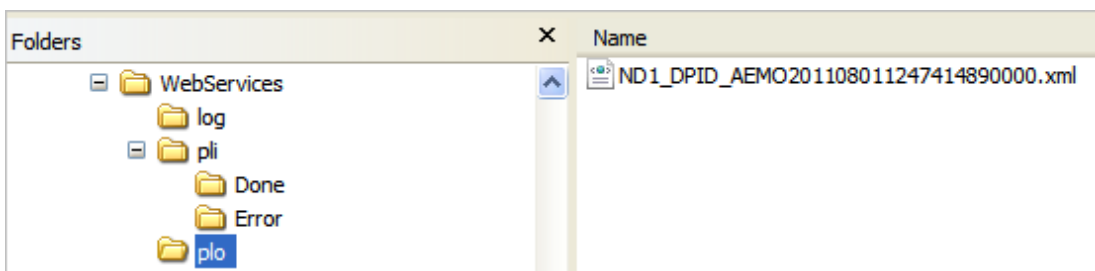
Be sure to edit the webServicesPost.exe.config file first, see § 4.5.5 "Editing the WebServicesPost.exe.config file".

Name	Size	Type
log		File Folder
pli		File Folder
plo		File Folder
src		File Folder
log4net.dll	264 KB	Application Extension
log4net.xml	1,334 KB	XML Document
WebClient.dll	17 KB	Application Extension
WebClient.pdb	28 KB	PDB File
WebClient.xml	13 KB	XML Document
WebServices.chm	150 KB	Compiled HTML Help file
WebServicesGet.exe	15 KB	Application
WebServicesGet.exe.config	4 KB	CONFIG File
WebServicesGet.pdb	24 KB	PDB File
WebServicesGet.xml	5 KB	XML Document
WebServicesPost.exe	16 KB	Application
WebServicesPost.exe.config	3 KB	CONFIG File
WebServicesPost.pdb	26 KB	PDB File
WebServicesPost.xml	6 KB	XML Document

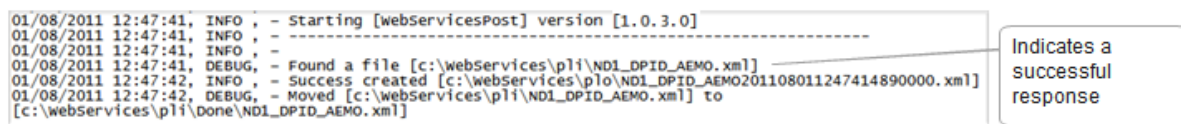
3. The web service script runs, watch for any errors.
4. While the `WebServicesPost.exe` is running, it checks periodically for a file.



5. When the script completes, a successful request is indicated by a response created in the participant outbox directory (`plo`).



6. Otherwise, check the `WebServicesPost.log` for any errors. Fix the errors and run the `WebServicesPost.exe` again until you have a working web services Get sample, see "Response Codes"<sup>51</sup> and see "Needing help?"<sup>56</sup>.



#### 4.5.8 Creating .NET Framework-based application instances

To make different web service request using the .NET Framework-based application, participants create multiple instances of the application in different locations (for example, one instance for NMI Discovery, and one for MSATS Limits). The .NET Framework-based application only processes one type of web service request within one instance.

Each instance must have corresponding `.exe.config` and `web_services.exe` files in the application's root directory. Each instance is configured differently and can only connect to one AEMO server with one User ID.

To convert the working sample to another instance:

1. Copy the `webServices` directory to another location on your participant systems.
2. Configure the new `.exe.config` files (see § 4.5.4 "Editing the Web-ServicesGet.exe.config file" and see § 4.5.5 "Editing the WebServicesPost.exe.config file").

## 4.6 Maintenance

---

The following housekeeping tasks are needed to keep your `web_services_client` software running smoothly:

1. Purge log files older than a specified date.
2. AEMO passwords have a 90-day expiry, so you need to have a password change process in place to ensure your password does not expire. In the event that your password expires or becomes locked out, please contact your Participant Administrator (who may need to contact the AEMO's Information and Support Hub, see "References" on page 59).

# 5 Library Functions

In this chapter:

---

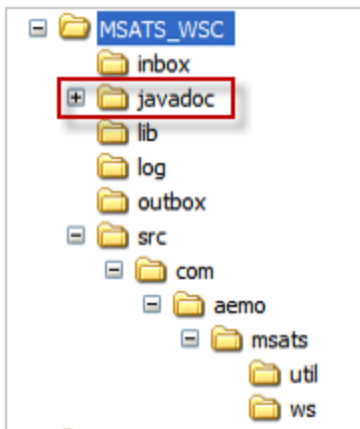
<b>5.1 Reference documentation</b> .....	<b>39</b>
<b>5.2 C4 NMI Master report library functions</b> .....	<b>40</b>
<b>5.3 MSATS Limits library functions</b> .....	<b>42</b>
<b>5.4 NMI Detail library functions</b> .....	<b>42</b>
<b>5.5 NMI Discovery library functions</b> .....	<b>44</b>
<b>5.6 NMI Discovery 3 library functions</b> .....	<b>48</b>
<b>5.7 Participant System Status library functions</b> .....	<b>49</b>

## 5.1 Reference documentation

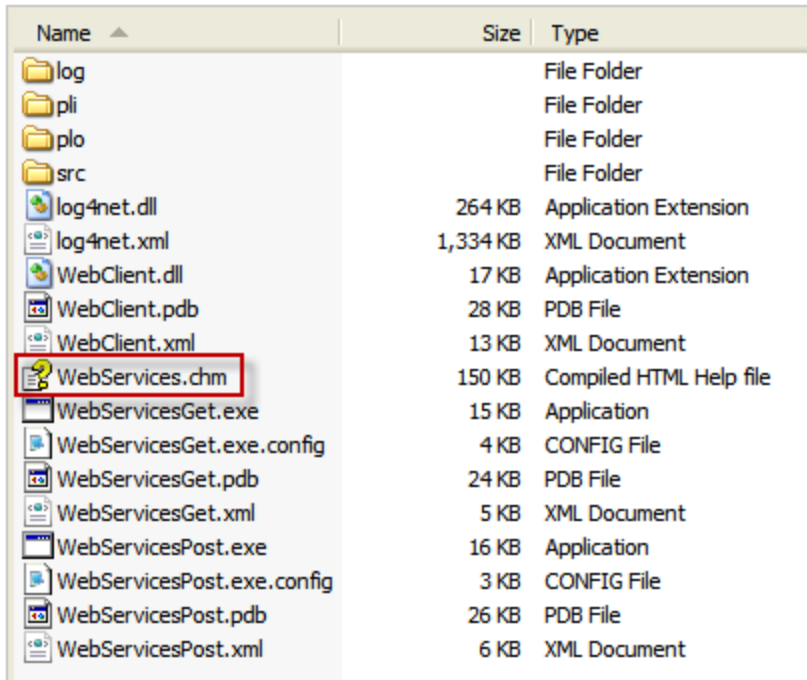
---

For further help with library functions, see the reference documentation in the relevant web services software directory:

- Java Platform, see the javadoc folder. To open, click `index.html`.



- .NET Framework application, see the `webServices.chm` file (this file cannot be read from a network drive).



Name	Size	Type
log		File Folder
pli		File Folder
plo		File Folder
src		File Folder
log4net.dll	264 KB	Application Extension
log4net.xml	1,334 KB	XML Document
WebClient.dll	17 KB	Application Extension
WebClient.pdb	28 KB	PDB File
WebClient.xml	13 KB	XML Document
<b>WebServices.chm</b>	150 KB	Compiled HTML Help file
WebServicesGet.exe	15 KB	Application
WebServicesGet.exe.config	4 KB	CONFIG File
WebServicesGet.pdb	24 KB	PDB File
WebServicesGet.xml	5 KB	XML Document
WebServicesPost.exe	16 KB	Application
WebServicesPost.exe.config	3 KB	CONFIG File
WebServicesPost.pdb	26 KB	PDB File
WebServicesPost.xml	6 KB	XML Document

## 5.2 C4 NMI Master report library functions

---

### 5.2.1 Contents

- 5.2.2 Get C4 NMI Master Report
- 5.2.3 Post C4 NMI Master Report

### 5.2.2 Get C4 NMI Master Report

Make a Get request to the C4 – NMI Master Report web service.

	Java	.NET Framework
Class	com.aemo.msats.ws.C4	No equivalent library functions.
Method	doC4(parameters)	
Parameters	<p>transactionId: identifies the requestor's transaction. Make unique to enable tracking of request.</p> <p>NMI: single NMI only.</p> <p>fromDate: start date of change.</p> <p>toDate: start date of change.</p> <p>asatDate: view standing data from this date.</p> <p>participantId: optional parameter, must be the same as the participant in the requesting URL and the "From" participant. Required for schema validation and consistency in output only.</p> <p>roleId: optional parameter, valid CATS role, e.g. FRMP. Required for schema validation only—if not supplied, then the first valid role for the participant is used.</p> <p>initTransId: optional parameter containing the transaction ID of the initiating participant.</p>	
Return value	The aseXML web service response.	

### 5.2.3 Post C4 NMI Master Report

Make a Post request to the C4 – NMI Master Report web service by posting an aseXML file from the inbox. The file should contain a single, valid ReportRequest transaction. If the web service request is successful, the response file is written to the local outbox.

	Java	.NET Framework
Class	com.aemo.msats.ws.C4	No equivalent library functions.
Method	doC4(parameters)	
Parameters	<p>dataFilename: the file to post from the inbox.</p> <p>requestContentType: defines the file type as XML or ZIP.</p>	
Return value	The aseXML web service response written to the local outbox.	

## 5.3 MSATS Limits library functions

### 5.3.1 Contents

- 5.3.2 Get MSATS Limits
- 5.3.3 Post MSATS Limits

### 5.3.2 Get MSATS Limits

Make a Get request to the MSATS Limits web service.

	Java	.NET Framework
Class	com.aemo.msats.ws.MSATSLimits	Aemo.Web.Client.WebClient
Method	doMSATSLimitsTransid("transactionId")	DoMsatsLimits(string transactionId)
Parameters	transactionId: Identifies the requestor's transaction. Make unique to enable tracking of the request.	transactionId: Identifies the requestor's transaction. Make unique to enable tracking of the request.
Return value	The aseXML web service response.	The aseXML stream dependent on the accept content requested.

### 5.3.3 Post MSATS Limits

Make a request to the MSATS Limits web service by posting an aseXML file from the inbox. The file should contain a single, valid ReportRequest transaction. If the web service request is successful, the response file is written to the local outbox.

	Java	.NET Framework
Class	com.aemo.msats.ws.MSATSLimits	No equivalent library functions.
Method	doMSATSLimits ("dataFilename", "requestContentType")	
Parameters	dataFilename: the file to post from the inbox. requestContentType: defines the file type as XML or ZIP.	
Return value	The aseXML web service response written to the local outbox.	

## 5.4 NMI Detail library functions

### 5.4.1 Contents

- 5.4.2 Get NMI Detail
- 5.4.3 Post NMI Detail



### 5.4.2 Get NMI Detail

Make a Get request to the NMI Detail web service to fetch details for a specific NMI.

	Java	.NET Framework
Class	com.aemo.msats.ws.NMIDetail	Aemo.Web.Client.WebClient
Method	doNmiDetail("transactionId", "nmi", "checksum")	DoNmiDetail(string transactionId, string nmi, string checksum)
Parameters	<p>transactionId: identifies the requestor's transaction. Make unique to enable tracking of request.</p> <p>nmi: identifies which national metering identifier (NMI) to fetch details for.</p> <p>checksum: single character string to check the NMI.</p>	<p>transactionId: identifies the requestor's transaction. Make unique to enable tracking of request.</p> <p>nmi: identifies which national metering identifier (NMI) to fetch details for.</p> <p>checksum: single character string to check the NMI.</p>
Return value	The aseXML web service response.	The aseXML stream dependent on the accept content requested.

### 5.4.3 Post NMI Detail

Make a request to the NMI Detail web service by posting an aseXML file from the inbox. The file should contain a single, valid NMISstandingDataRequest transaction. If the web service request is successful, the response file is written to the local outbox.

	Java	.NET Framework
Class	com.aemo.msats.ws.NMIDetail	Aemo.Web.Client.WebClient
Method	doNmiDetail("filename.xml", "text/xml")	DoNmiDetail(string dataFileName, string requestContentType)
Parameters	<p>dataFileName (Java), dataFileName (.NET): the aseXML file to post from the inbox.</p> <p>requestContentType: defines the file type as XML (text/xml) or ZIP (application/zip). Can include the web service version and schema version.</p>	<p>dataFileName (Java), dataFileName (.NET): the aseXML file to post from the inbox.</p> <p>requestContentType: defines the file type as XML (text/xml) or ZIP (application/zip). Can include the web service version and schema version.</p>
Return value	The aseXML web service response written to the local outbox.	The aseXML stream dependent on the accept content requested.

## 5.5 NMI Discovery library functions

### 5.5.1 Contents

- 5.5.2 Get NMI Discovery by DPID
- 5.5.3 Post NMI Discovery by DPID
- 5.5.4 Get NMI Discovery by meter serial
- 5.5.5 Post NMI Discovery by meter serial
- 5.5.6 Get NMI Discovery by address
- 5.5.7 Post NMI Discovery by address

### 5.5.2 Get NMI Discovery by DPID

Make a Get request to the NMI Discovery web service to search for NMIs by delivery point identifier.

	Java	.NET Framework
Class	<code>com.aemo.msats.ws.NMIDiscovery</code>	<code>Aemo.Web.Client.WebClient</code>
Method	<code>doNmiDiscoveryDpid("jurisdictionCode", "transactionId", "deliveryPointIdentifier")</code>	Method: <code>DoNmiDiscoveryDpid(string jurisdictionCode, string transactionId, string deliveryPointIdentifier)</code>
Parameters	<p><code>jurisdictionCode</code>: nominates a jurisdiction to search for e.g. NSW.</p> <p><code>transactionId</code>: identifies the requestor's transaction. Make unique to enable tracking of request.</p> <p><code>deliveryPointIdentifier</code>: the postal delivery point to search for.</p>	<p><code>jurisdictionCode</code>: nominates a jurisdiction to search for e.g. NSW.</p> <p><code>transactionId</code>: identifies the requestor's transaction. Make unique to enable tracking of request.</p> <p><code>deliveryPointIdentifier</code>: the postal delivery point to search for.</p>
Return value	The aseXML web service response.	The aseXML stream dependent on the accept content requested.

### 5.5.3 Post NMI Discovery by DPID

Make a request to the NMI Discovery web service by posting an aseXML file from the inbox. The file should contain a single, valid `NMIDiscoveryRequest` transaction. If the web service request is successful, the response file is written to the outbox.

	Java	.NET Framework
Class	<code>com.aemo.msats.ws.NMIDiscovery</code>	<code>Aemo.Web.Client.WebClient</code>
Method	<code>doNmiDiscovery("dataFilename", "requestContentType")</code>	<code>DoNmiDiscovery(string dataFileName, string requestContentType)</code>

	Java	.NET Framework
Parameters	<p><b>dataFilename:</b> the aseXML file to post from the inbox.</p> <p><b>requestContentType:</b> defines the file type as XML (text/xml) or ZIP (application/zip). Can include web service and schema version.</p>	<p><b>dataFileName:</b> the aseXML file to post from the inbox.</p> <p><b>requestContentType:</b> defines the file type as XML (text/xml) or ZIP (application/zip). Can include web service and schema version.</p>
Return value	The aseXML web service response written to the local outbox.	The aseXML stream dependent on the accept content requested.

#### 5.5.4 Get NMI Discovery by meter serial

Make a Get request to the NMI Discovery web service to search for NMIs by meter serial number.

	Java	.NET Framework
Class	<code>com.aemo.msats.ws.NMIDiscovery</code>	<code>Aemo.Web.Client.WebClient</code>
Method	<code>doNmiDiscoveryMeterSerial("jurisdictionCode", "transactionId", "meterSerialNumber")</code>	<code>DoNmiDiscoveryMeterSerial(string jurisdictionCode, string transactionId, string meterSerialNumber)</code>
Parameters	<p><b>jurisdictionCode:</b> nominates a jurisdiction to search for e.g. NSW.</p> <p><b>transactionId:</b> identifies the requestor's transaction. Make unique to enable tracking of request.</p> <p><b>meterSerialNumber:</b> the meter serial number to search for.</p>	<p><b>jurisdictionCode:</b> nominates a jurisdiction to search for e.g. NSW.</p> <p><b>transactionId:</b> identifies the requestor's transaction. Make unique to enable tracking of request.</p> <p><b>meterSerialNumber:</b> the meter serial number to search for.</p>
Return Value	The aseXML web service response dependent on the content requested.	The aseXML stream dependent on the accept content requested.

#### 5.5.5 Post NMI Discovery by meter serial

Make a request to the NMI Discovery web service by Posting an aseXML file from the inbox. The file should contain a single, valid `NMIDiscoveryRequest` transaction. If the web service request is successful, the response file is written to the outbox.

	Java	.NET Framework
Class	<code>com.aemo.msats.ws.NMIDiscovery</code>	No equivalent library functions.
Method	<code>doNmiDiscovery("dataFilename", "requestContentType")</code>	
Parameters	<code>dataFilename</code> : the file to post from the inbox.  <code>requestContentType</code> : defines the file type as XML or ZIP.	
Return value	The aseXML web service response written to the outbox, dependent on the content requested.	

### 5.5.6 Get NMI Discovery by address

Make a Get request to the NMI Discovery web service to search for NMIs by address. Either a `suburbOrPlaceOrLocality`, a `postcode`, or both values must be provided. A `stateOrTerritory` value must be provided. Other address parameters can be null.

	Java	.NET Framework
Class	<code>com.aemo.msats.ws.NMIDiscovery</code>	<code>Aemo.Web.Client.WebClient</code>
Method	<code>doNmiDiscoveryAddress("jurisdictionCode", "transactionId", "buildingOrPropertyName", "locationDescriptor", "lotNumber", "flatOrUnitType", "flatOrUnitNumber", "floorOrLevelType", "floorOrLevelNumber", "houseNumber", "houseNumberSuffix", "streetName", "streetSuffix", "streetType", "suburbOrPlaceOrLocality", "postcode", "stateOrTerritory")</code>	<code>DoNmiDiscoveryAddress(string jurisdictionCode, string transactionId, string buildingOrPropertyName, string locationDescriptor, string lotNumber, string flatOrUnitType, string flatOrUnitNumber, string floorOrLevelType, string floorOrLevelNumber, string houseNumber, string houseNumberSuffix, string streetName, string streetSuffix, string streetType, string suburbOrPlaceOrLocality, string postcode, string stateOrTerritory)</code>

	Java	.NET Framework
Parameters	<p><b>jurisdictionCode:</b> nominates a jurisdiction to search for e.g. NSW.</p> <p><b>transactionId:</b> identifies the requestor's transaction. Make unique to enable tracking of request.</p> <p><b>buildingOrPropertyName:</b> building or property name as per Australian Standard AS4590.</p> <p><b>LocationDescriptor:</b> location descriptor as per Australian Standard AS4590.</p> <p><b>LotNumber:</b> lot number as per Australian Standard AS4590.</p> <p><b>flatOrUnitType:</b> flat or unit types as per Australian Standard AS4590:2006.</p> <p><b>flatOrUnitNumber:</b> flat or unit number as per Australian Standard AS4590.</p> <p><b>floorOrLevelType:</b> floor or level types as per Australian Standard AS4590:2006.</p> <p><b>floorOrLevelNumber:</b> floor or level number as per Australian Standard AS4590.</p> <p><b>houseNumber:</b> house number as per Australian Standard AS4590.</p> <p><b>houseNumberSuffix:</b> house number suffix as per Australian Standard AS4590.</p> <p><b>streetName:</b> street name as per Australian Standard AS4590.</p> <p><b>streetSuffix:</b> street suffixes as per Australian Standard AS4590:2006.</p> <p><b>streetType:</b> street types as per Australian Standard AS4590:2006.</p> <p><b>suburbOrPlaceOrLocality:</b> suburb or locality as per Australian Standard AS4590.</p> <p><b>postcode:</b> Australian postcode as per Australian Standard AS4590.</p>	<p><b>jurisdictionCode:</b> nominates a jurisdiction to search for e.g. NSW.</p> <p><b>transactionId:</b> identifies the requestor's transaction. Make unique to enable tracking of request.</p> <p><b>buildingOrPropertyName:</b> building or property name as per Australian Standard AS4590.</p> <p><b>LocationDescriptor:</b> location descriptor as per Australian Standard AS4590.</p> <p><b>LotNumber:</b> lot number as per Australian Standard AS4590.</p> <p><b>flatOrUnitType:</b> flat or unit types as per Australian Standard AS4590:2006.</p> <p><b>flatOrUnitNumber:</b> flat or unit number as per Australian Standard AS4590.</p> <p><b>floorOrLevelType:</b> floor or level types as per Australian Standard AS4590:2006.</p> <p><b>floorOrLevelNumber:</b> floor or level number as per Australian Standard AS4590.</p> <p><b>houseNumber:</b> house number as per Australian Standard AS4590.</p> <p><b>houseNumberSuffix:</b> house number suffix as per Australian Standard AS4590.</p> <p><b>streetName:</b> street name as per Australian Standard AS4590.</p> <p><b>streetSuffix:</b> street suffixes as per Australian Standard AS4590:2006.</p> <p><b>streetType:</b> street types as per Australian Standard AS4590:2006.</p> <p><b>suburbOrPlaceOrLocality:</b> suburb or locality as per Australian Standard AS4590.</p> <p><b>postcode:</b> Australian postcode as per Australian Standard AS4590.</p> <p><b>stateOrTerritory:</b> Australian states and territories as per Australian Standard AS4590.</p>

	Java	.NET Framework
	stateOrTerritory: Australian states and territories as per Australian Standard AS4590.	
Return value	The aseXML web service response dependent on the content requested.	The aseXML stream dependent on the accept content requested.

### 5.5.7 Post NMI Discovery by address

	Java	.NET Framework
Class	com.aemo.msats.ws.NMIDiscovery	No equivalent library functions.
Method	doNmiDiscovery("dataFilename", "requestContentType")	
Parameters	dataFilename: the file to post from the inbox. requestContentType: defines the file type as XML or ZIP.	
Return value	The response file is written to the local outbox.	

## 5.6 NMI Discovery 3 library functions

### 5.6.1 Contents

- 5.6.2 Get NMI Discovery Type 3
- 5.6.3 Post NMI Discovery Type 3

### 5.6.2 Get NMI Discovery Type 3

Make a Get request to the NMI Discovery Type 3 web service.

	Java	.NET Framework
Class	com.aemo.ws.msats.NMIDetail.	No equivalent library functions.
Method	doNMIDetail ("transactionId", "nmi", "checksum", "type", "reason").	
Parameters	nmi, checksum, type, reason, transactionId: identifies the requestor's transaction; make unique to enable tracking of request.	
Return value	The aseXML web service response.	

### 5.6.3 Post NMI Discovery Type 3

Make a request to the NMI Discovery Type 3 web service by posting a valid aseXML file, containing a NMI Detail transaction request, from the local inbox. If the web service request is successful, the response file is written to the local outbox.

	Java	.NET Framework
Class	com.aemo.ws.msats.NMIDetail.	No equivalent library functions.
Method	doNMIDetail ("transactionId", "nmi", "checksum", "type", "reason").	
Parameters	dataFilename: aseXML file to post from the inbox. requestContentType: defines the file type as XML (text/xml) or ZIP (application/zip). Can include the web service version and schema version.	
Return value	The aseXML web service response written to the local outbox.	

## 5.7 Participant System Status library functions

### 5.7.1 Contents

- 5.7.2 Get Participant System Status
- 5.7.3 Post Participant System Status

### 5.7.2 Get Participant System Status

Make a Get request to the Participant System Status web service.

	Java	.NET Framework
Class	com.aemo.msats.ws.ParticipantSystemStatusBean	No equivalent library functions.
Method	doParticipantSystemStatusTransid("transactionId")	
Parameters	transactionId: identifies the requestor's transaction. Make unique to enable tracking of request.	
Return value	The aseXML web service response.	

### 5.7.3 Post Participant System Status

Make a request to the Participant System Status web service by posting a valid aseXML file, containing a report transaction request, from the local inbox. If the web service request is successful, the response file is written to the local outbox.

	Java	.NET Framework
Class	<code>com.aemo.ws.msats.ParticipantSystemStatusBean</code>	No equivalent library functions.
Method	<code>dodoParticipantSystemStatus("dataFilename", "requestContentType")</code>	
Parameters	<p><code>dataFilename</code>: aseXML file to post from the inbox.</p> <p><code>requestContentType</code>: defines the file type as XML (<code>text/xml</code>) or ZIP (<code>application/zip</code>). Can include the web service version and schema version.</p>	
Return value	The aseXML web service response written to the local outbox.	



## 6 Response Codes

In this chapter:

---

<b>6.1 Contents</b> .....	<b>51</b>
<b>6.2 Console and log file responses</b> .....	<b>51</b>
<b>6.3 Java software response codes</b> .....	<b>52</b>
<b>6.4 Microsoft .NET Framework-based web services client software response codes</b> ...	<b>53</b>

### 6.1 Contents

---

- 6.2 Console and log file responses
- 6.3 Java software response codes
- 6.4 Microsoft .NET Framework-based web services client software response codes

### 6.2 Console and log file responses

---

All response codes are listed in the log file and on the console, for example:

*Figure 6-1: console response example*

```

C:\Windows\system32\cmd.exe

C:\Develop\MSATS_WSC>nmid_fail.cmd
MSATS Web Service API

Parameter List:
    nmid_fail.properties
    NMIDetail
    file=nmid_fail.xml
    type=text/xml

Feb 13, 2013 10:37:14 AM com.aemo.msats.util.Constants setConstants
INFO: Constants#setConstants(): LogManager configured.
Testing doNmiDetail() ...
Response Code: 400 - Bad Request
Response Detail: text/html; UTF-8; null
<HTML><HEAD><TITLE>400 Bad Request</TITLE></HEAD>
<BODY><H1>400 Bad Request</H1>
NMI not found<BR/>
</BODY></HTML>

C:\Develop\MSATS_WSC>
    
```

Figure 6-2: log file response example

```

16/08/2011 16:14:20, ERROR, – Web request exception. The remote server returned an
error: (403) Forbidden.
16/08/2011 16:14:20, ERROR, – - Req Headers Authorization=Basic
bWlrZW11aXI6QmxvbmRpZTAy
16/08/2011 16:14:20, ERROR, – - Req Headers Accept=text/xml
16/08/2011 16:14:20, ERROR, – - Req Headers Host=002.test.nemnet.NET.au
16/08/2011 16:14:20, ERROR, – - Req Headers Connection=Keep-Alive
16/08/2011 16:14:20, ERROR, – - Response <HTML><HEAD><TITLE>403
Forbidden</TITLE></HEAD>
<BODY><H1>403 Forbidden</H1>Invalid Authorization header<BR/> </BODY></HTML>
16/08/2011 16:14:20, ERROR, – Fatal error, abnormal exit
System.NET.WebException: The remote server returned an error: (403) Forbidden.
  at System.NET.HttpWebRequest.GetResponse()
  at Aemo.Web.Client.WebClient.GetResponse(HttpWebRequest httpWebRequest) in
C:\nem\DotNet2\Samples\Other\WebServices\WebClient\WebClient.cs:line 338
  at Aemo.Web.Client.WebClient.DoGet(Dictionary`2 parameters) in
C:\nem\DotNet2\Samples\Other\WebServices\WebClient\WebClient.cs:line 209
  at Aemo.Web.Client.WebClient.DoNmiDetail(String transactionId, String nmi, String
checksum) in
C:\nem\DotNet2\Samples\Other\WebServices\WebClient\WebClientGet.cs:line 75
  at WebServicesGet.GetOperations.TryTheVariousCalls() in
C:\nem\DotNet2\Samples\Other\WebServices\WebServicesGet\GetOperations.cs:line 74
  at WebServicesGet.Program.Main() in
C:\nem\DotNet2\Samples\Other\WebServices\WebServicesGet\Program.cs:line 68

```

### 6.3 Java software response codes

This table lists some of the response codes displayed in the log file or on the console.

Table 6-1: Java software response codes

Code	Description
200	Indicates a successful request.
400	No <participantId> provided in the request.
401	The WWW-Authenticate: Basic realm="<applicationId>" header is not provided.
403	The user credentials cannot be decoded and authenticated.
404	The webpage cannot be found.
405	The requested method is not supported.
406	The version specified in the header is not supported.

Code	Description
413	The posted file has exceeded its limit of 1 MB.
500	AEMO Internal Server Error
503	The configured limit for concurrent requests or request rate is exceeded.

## 6.4 Microsoft .NET Framework-based web services client software response codes

This table lists some of the response codes in the WebClient.dll library. The response codes are found in the log file or displayed on the console. For further help with parameter values, see "Library Functions" on page 39.

Table 6-2: MS .NET Framework-based software response codes

Problem	Error
Action routine	Fatal error, abnormal exit System.ArgumentNullException: WebClient errorAction routine cannot be null Parameter name: errorAction
Content type	Fatal error, abnormal exit System.ArgumentNullException: DoPost with null or empty contentType Parameter name: contentType
Host key value is missing.	Fatal error, abnormal exit System.ArgumentNullException: WebClient.Host is null or empty Parameter name: Host
MSATS Limits request transactionId missing.	Fatal error, abnormal exit System.ArgumentNullException: DoMsatsLimits transactionId cannot be null or empty Parameter name: transactionId
NMI Detail request Checksum missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDetail Checksum cannot be null or empty Parameter name: checksum
NMI Detail request ContentType missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDetail requestContentType cannot be null or empty Parameter name: requestContentType
NMI Detail request dataFilename missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDetail dataFilename cannot be null or empty Parameter name: dataFileName
NMI Detail request Nmi missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDetail Nmi cannot be null or empty Parameter name: nmi
NMI Detail request TransactionId missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDetail TransactionId cannot be null or empty Parameter name: transactionId
NMI Discovery address search jurisdictionCode missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscoveryAddress jurisdictionCode cannot be null or empty Parameter name: jurisdictionCode
NMI Discovery address search transactionId missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscoveryAddress transactionId cannot be null or empty Parameter name: transactionId

Problem	Error
NMI Discovery DPID search <code>deliveryPointIdentifier</code> missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscoveryDpid <code>deliveryPointIdentifier</code> cannot be null or empty Parameter name: <code>deliveryPointIdentifier</code>
NMI Discovery DPID search <code>jurisdictionCode</code> missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscoveryDpid <code>jurisdictionCode</code> cannot be null or empty Parameter name: <code>jurisdictionCode</code>
NMI Discovery DPID search <code>transactionId</code> missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscoveryDpid <code>transactionId</code> cannot be null or empty Parameter name: <code>transactionId</code>
NMI Discovery meter serial search <code>jurisdictionCode</code> missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscoveryMeterSerial <code>jurisdictionCode</code> cannot be null or empty Parameter name: <code>jurisdictionCode</code>
NMI Discovery meter serial search <code>meterSerialNumber</code> missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscoveryMeterSerial <code>meterSerialNumber</code> cannot be null or empty Parameter name: <code>meterSerialNumber</code>
NMI Discovery meter serial search <code>transactionId</code> missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscoveryMeterSerial <code>transactionId</code> cannot be null or empty Parameter name: <code>transactionId</code>
NMI Discovery request <code>dataFileName</code> missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscovery <code>dataFileName</code> cannot be null or empty Parameter name: <code>dataFileName</code>
NMI Discovery request <code>requestContentType</code> missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDiscovery <code>requestContentType</code> cannot be null or empty Parameter name: <code>requestContentType</code>
<code>Participant</code> key value is missing.	Fatal error, abnormal exit System.ArgumentNullException: WebClient.Participant cannot be null or empty Parameter name: <code>participant</code>
<code>Password</code> key value is missing.	Fatal error, abnormal exit System.ArgumentNullException: WebClient.Password cannot be null or empty Parameter name: <code>password</code>
<code>Protocol</code> key value must be http or https.	Fatal error, abnormal exit System.ArgumentException: WebClient.Protocol incorrect must be http or https
<code>ProxyPort</code> key value is incorrect.	Fatal error, abnormal exit System.ArgumentException: WebClient.ProxyPort is not a +ve integer or is > 65535
Remote server	Web request exception. The remote server returned an error: xxxxxxxxxx.
<code>Resource</code> key value is incorrect.	Fatal error, abnormal exit System.ArgumentException: WebClient.Resource must start and end with /
<code>Resource</code> key value is missing.	Fatal error, abnormal exit System.ArgumentNullException: WebClientResource string cannot be null or empty Parameter name: <code>resource</code>
Null Post Data	Fatal error, abnormal exit System.ArgumentNullException: DoPost with null post data stream Parameter name: <code>streamPostData</code>

Problem	Error
Timeout key value is incorrect.	Fatal error, abnormal exit System.ArgumentException: Timeout value in milliseconds cannot be zero or -ve
UserName key value is missing.	Fatal error, abnormal exit System.ArgumentNullException: WebClient.Username cannot be null or empty Parameter name: username

This table lists some of the `webServicesGet` and `webServicesPost` response codes. All configuration parameters in the `.exe.config` files are checked and must be either, not null and non-blank, otherwise the following errors are produced.

Problem	Error
Config item is blank	Fatal error, abnormal exit System.ArgumentNullException: Value cannot be null. Parameter name: Config item blank [xxxx]
Config item is null	Fatal error, abnormal exit System.ArgumentNullException: Value cannot be null. Parameter name: Config item null [xxxx]
NMI Detail Checksum is missing.	Fatal error, abnormal exit System.ArgumentNullException: DoNmiDetail Checksum cannot be null or empty Parameter name: xxxxx
Protocol incorrect, must be http or https.	Fatal error, abnormal exit System.FormatException: Protocol incorrect must be http or https

## 6 Needing help?

In this chapter:

---

<b>6.5 Authentication errors</b> .....	<b>56</b>
<b>6.6 .NET Framework-based web services client software</b> .....	<b>56</b>
<b>6.7 Other errors</b> .....	<b>57</b>
<b>6.8 AEMO's Information and Support Hub</b> .....	<b>57</b>

### 6.5 Authentication errors

---

- Check credentials (passwords do expire).

Java Platform web services client software

- Is your web services software sample installed on your C:\ drive and not on a network drive?
- Do the settings in the `setenv.cmd` and the `.properties` file match the location of your Java installation?
- Is your Java developer version 7 or above? To check, use `java -version` in the command line.
- Have you edited `sample.properties` file, especially `http.username`, `http.password`, `participant`?
- Do the `inbox`, `outbox`, `truststore` and `keystore` properties have the correct paths?
- Do the path locations in your `.properties` file have forward slashes and not backslashes?
- Are the `sample.properties` and `sample.CMD` files in the application root directory?

### 6.6 .NET Framework-based web services client software

---

- Is your web services software sample installed on your C:\ drive and not on a network drive?
- Is your .NET Framework version 4?
- Have you edited the `webServicesGet.exe.config` file, especially `File`, `Participant`, `Username`, `Password`, `Host`, and selected a `Get` request?
- Have you edited the `webServicesPost.exe.config` file, especially `File`, `Participant`, `Username`, `Password`, and `Host`?
- Do the `p1i` and `p1o` key values have the correct paths?

- If your application is stopping or exiting with an error, check:
  - All required property values exist in the `.exe.config` file.
  - The submitting `.XML` file in the `pli` directory is correct.
- Are the `WebServicesPost.exe.config` and `WebServicesPost.exe` files in the application root directory?
- For new instances, do your `.exe.config` and `.exe` files have the same name, for example, `WebServicesPost.exe.config` and `WebServicesPost.exe`?

## 6.7 Other errors

---

- Do you have the latest release of AEMO's web services client software for your participant environment?

## 6.8 AEMO's Information and Support Hub

---

### 6.8.1 Contacting AEMO's Information and Support Hub

IT assistance is requested through AEMO's Information and Support Hub using one of the following methods:

- Phone: 1300 AEMO 00 (1300 226 600) and follow the prompts.

For non-urgent issues, normal coverage is 8:00 AM to 6:00 PM on weekdays, Eastern Standard Time (EST).

- Email: [supporthub@aemo.com.au](mailto:supporthub@aemo.com.au)
- The Customer Portal, <http://helpdesk.preprod.nemnet.net.au/nemhelplite/> allows you to log your own requests for assistance. For access credentials, see your company's IT security contact or PA.

---

Please note that AEMO recommends participants call AEMO's Information and Support Hub for all urgent issues, whether or not you have logged a call in the Customer Portal.

---

### 6.8.2 What can I check before requesting IT assistance from AEMO?

- Check the status of your organisation's IT systems with your internal IT department.
- Check the status of AEMO's production market systems. Call 1300 AEMO 00 (1300 236 600) and following the prompts to listen to the recording.
- Check the "Needing Help" section of AEMO's user guides. To obtain guides, see [Using Energy Market Information Systems](#).
- Check the [IT Assistance](#) topics on AEMO's website.

### **6.8.3 Information to provide AEMO**

Please provide the following information when requesting IT assistance from AEMO:

- Your name
- Organisation name
- Participant ID
- System or application name
- Environment: production or pre-production
- Problem description
- Steps that caused the problem
- Screenshots

For AEMO software-related issues please also provide:

- Version of software
- Logs of abnormal behaviour
- Properties file
- Diagram of your organisation's IT architecture
- Can you reproduce the problem?



## 6 References

The resources listed in this section contain additional related information that may assist you.

In this chapter:

---

<b>6.9 Rules, Law, and Government Bodies</b> .....	<b>59</b>
<b>6.10 Oracle</b> .....	<b>59</b>
<b>6.11 AEMO's website</b> .....	<b>59</b>
<b>6.12 Feedback</b> .....	<b>61</b>

### 6.9 Rules, Law, and Government Bodies

---

- "Australian Energy Market Commission" (AEMC), electricity and gas rules  
<http://www.aemc.gov.au/index.html>. Viewed 06 February 2013.
- "Australian Energy Regulator (AER)", [www.aer.gov.au](http://www.aer.gov.au). Viewed 06 February 2013.

### 6.10 Oracle

---

- "Oracle Downloads", SE 7 JRE and JDK  
downloads:<http://www.oracle.com/technetwork/java/javase/downloads/index.html>.  
Viewed 21 March 2013.

### 6.11 AEMO's website

---

You can find the following resources on AEMO's website:

- "aseXML Standards", help with aseXML, including guidelines, schemas, change process, sample files and white papers, <http://www.aemo.com.au/About-the-Industry/Information-Systems/aseXML-Standards> (Home>About the Industry>Information Systems>aseXML Standards). Viewed 30 January 2013.
- *Guide to User Rights Management*, <http://www.aemo.com.au/About-the-Industry/Information-Systems/Using-Energy-Market-Information-Systems> (Home > About the Industry > Information Systems). Viewed 15 February 2013.
- *Guide to Web Services*, [http://aemo.com.au/About-the-Industry/Information-Systems/Using-Energy-Market-Information-Systems#Web\\_Services](http://aemo.com.au/About-the-Industry/Information-Systems/Using-Energy-Market-Information-Systems#Web_Services) (Home > About the Industry > Information Systems > Using Energy Market Information Systems). Viewed 14 March 2013.

- "IT Assistance", information to assist participants with IT related issues:  
<http://www.aemo.com.au/About-the-Industry/Information-Systems/IT-Assistance>  
 (Home > About the Industry > Information Systems > IT Assistance). Viewed 15 February 2013.
- "MSATS B2B User Interface Guide", <http://www.aemo.com.au/Electricity/Policies-and-Procedures/Market-Settlement-and-Transfer-Solutions/MSATS-Participant-User-Interface-Guides> (Home > Electricity > Policies & Procedures > MSATS > MSATS Participant User Interface Guides). Viewed 11 February 2013.
- *MSATS NMI Discovery Questions and Answers*, contains answers compiled in response to questions from participants asking how to get the best results from the MSATS NMI Discovery features. Its purpose is to assist participants' staff who are actively involved in discovering NMIs through MSATS. See  
<http://www.aemo.com.au/Electricity/Policies-and-Procedures/Market-Settlement-and-Transfer-Solutions/NMI-Discovery-Questions-and-Answers> (Home>Electricity>Policies & Procedures>MSATS>NMI Discovery Questions and Answers). Viewed 15 February 2013.
- "MSATS Participant User Interface Guides", <http://www.aemo.com.au/Electricity/Policies-and-Procedures/Market-Settlement-and-Transfer-Solutions/MSATS-Participant-User-Interface-Guides> (Home > Electricity > Policies & Procedures > MSATS > MSATS Participant User Interface Guides). Viewed 21 February 2013.
- "RESTful Web Services: *The basics*",  
<http://www.ibm.com/developerworks/webservices/library/ws-restful/>. Viewed 06 February 2013.
- *Technical Guide to Electricity IT Systems*, <http://aemo.com.au/About-the-Industry/Information-Systems/Using-Energy-Market-Information-Systems>  
 (<http://aemo.com.au/About-the-Industry/Information-Systems/Using-Energy-Market-Information-Systems>). Viewed 15 February 2013.
- "Guide to Transition of aseXML", <http://www.aemo.com.au/About-the-Industry/Information-Systems/Using-Energy-Market-Information-Systems>(Home > About the Industry > Information Systems > Using Energy Market Information Systems). Viewed 04 February 2013.

Note: AEMO can change the version of aseXML for output, but the timing does not always suit data recipients. To assist data recipients with making the transition, AEMO supports the delivery of both the new and immediately superseded versions of aseXML data. Each participant receives data conforming to one of the versions at any one time.

- "Using Energy Market Information Systems", IT systems documentation and software:  
<http://www.aemo.com.au/About-the-Industry/Information-Systems/Using-Energy-Market-Information-Systems> (Home > About the Industry > Information Systems > Using Energy Market Information Systems). Viewed 15 February 2013.

## 6.12 Feedback

---

To suggest corrections to this document, please contact [Information and Support Hub](#).

## 6 Index

.

.NET Framework-based web services client software quick start guide 25

.NET Framework root directory 27

.NET Framework SRC directory and subdirectories 28

**A**

Accept 31, 34

aseXML 7

aseXML version 7

**C**

C4 NMI Master report library functions 40

Configuring the Java .properties file 18

Console 31, 34

Creating .NET Framework-based application instances 38

Creating Java web services client software instances 25

CycleInSec 34

**D**

DoMsatsLimits 31

DoNmiDetail 31

DoNmiDiscoveryAddress 31

DoNmiDiscoveryDpid 31

DoNmiDiscoveryMeterSerial 31

Downloading the .NET Framework-based web services client software 26

Downloading the Java web services client software 16

**E**

Editing the WebServicesGet.exe.config file 28

Editing the WebServicesPost.exe.config file 31

Extracting the .NET Framework-based distribution file 26

Extracting the Java distribution file 16

## F

Figures iv

File size limits 9

## G

GET NMI Discovery by address 46

GET NMI Discovery by meter serial 45

Glossary v

## H

Headers 9

Host 30, 33

HTTP Processing 19

http.accept 20

http.host 20

http.password 20

http.port 20

http.protocol 19

http.timeout 20

http.username 20

HTTPS requests 8

HTTPS responses 11

## I

inbox 21

Information and Support Hub 57

## J

Java Platform 14

Java root directory 17

Java software response codes 52

Java SRC directory and subdirectories 17

Java web services client software quick start guide 15

Java web services client software response codes 52

javax.NET.debug 19

## K

keystore 20

keystore.password 21

keystore.type 20

## L

Library Functions 39

Logging configuration 21

## M

Maintenance 38

MaskXml 34

MaskZip 34

Microsoft .NET Framework-based application 15

Microsoft .NET Framework-based web services client software response codes 53

MS .NET Framework-based software response codes 53

MSATS Limits library functions 42

## N

NMI Detail library functions 42

NMI Discovery 3 library functions 48

NMI Discovery library functions 44

## O

outbox 21

## P

participant 20

Participant 30, 34

Participant System Status library functions 49

Password 30, 34

Performing a GET request from Internet Explorer 10

Pli 34

Plo 31, 34

POST NMI Discovery by address library functions 48

POST NMI Discovery by meter serial library functions 45

Protocol 30, 33

proxy.host 20

proxy.port 20

ProxyAddress 30, 33

ProxyPort 30, 33

## R

requesting IT assistance 57

Resource 34

Response Codes 51

RESTful architecture 3

Running the Java web services client software 22

Running the WebServicesGet.exe 35

Running the WebServicesPost.exe 36

## S

Security and authentication 8

System requirements 5

## T

Tables v

Timeout 30, 33

trustore 21

truststore.password 21

truststore.type 21

## U

URL 8

User access 8

UserName 30, 34

---

**W**

Web Services Client Software 13

WebServicesGet.exe.config appSettings properties 30

WebServicesGet.exe.config log4net parameters 29

WebServicesPost.exe.config appSettings properties 33

WebServicesPost.exe.config log4net parameters 32