ANZLIC
the Spatial Information Council

National Address Management Framework

Web Services Specification

Endorsed by the Ministerial Online and Communications Council on 12 December 2008

Document Control

Author: John Gottschalk

CTG Consulting

Document No: 01 **Date of Issue:** 12 December 2008
Version: 1.0

Document Release Information and Distribution

This is a controlled document. The Quality Document Register is maintained within CTG to manage the distribution of controlled documents and the issue of any revisions. Printed versions are uncontrolled.

Date	Version	Released To
8th August 2007	0.4	Bruce Thompson, Karina Donaire, Liz Marchant
20th August 2007	0.7	NAMF Project Team
4th Sept 2007	0.8	NAMF Project Team
11th Sept 2007	1.1	NAMF Project Team
August 2008	1.1	Approved by the NAMF Program Control Board
November 2008	1.1	Endorsed by the Cross Jurisdictional CIO Committee
12 December 2008	1.0	Official Version – endorsed by the Ministerial Online and Communications Council
May 2009	1.0	For publication on ANZLIC web site
July 2010	1.3.2	Changes to document to reflect first deployment of the system, and migration of development documentation to the Web Services Developers Guide.

Table of Contents

1 Glossary of Terms

Address	An address is a textual representation of a physical or virtual point for the purposes of either identifying a physical parcel of land on the land surface of the earth or as a communication channel to a recipient. An address may refer to items other than a physical point such as an email address. A single point may have many textual addresses associated with it. These can be different ways of representing the official address or, in the case of Private Estates, a way of addressing the internal structure of the Estate. Many attributes may also be associated with an address. For example, Australian Bureau of Statistics (ABS) Mesh Block/ Collector District, Business or Private use, Electoral district(s), Latitude and Longitude of are also attributes to an address.
AMAS	The Address Matching Approval System (AMAS®) is a certification program that has been developed by Australia Post to improve the accuracy of postal addresses. The software prepares addresses for barcode creation, ensures quality addressing, and enables mailers to qualify for postal discounts for Pre-Sort letter lodgements. More information can be found on the Australia Post website (www.auspost.com.au).
ANZLIC	ANZLIC — the Spatial Information Council — is the peak inter-governmental council for the coordination of spatial information policy and strategic issues in Australia and New Zealand. See www.anzlic.org.au for more information.
API	Application Programming Interface. An interface to a software application for access by other computer systems.
Call	A Call is the name of the Web Service function that is invoked by a client requesting the service. It consists of a name and a set of mandatory or optional arguments. The set of Calls and their attendant arguments comprise the web service interface. An example of a call is "*validateAddress"*
Data Custodian	The role of a Data Custodian, in relation to address data, is to ensure that an address data is up-to-date, accurate and as complete as possible. PSMA Australia is a Data Custodian of location addresses. Australia Post is a Data Custodian of postal addresses.
DPID	Delivery Point Identifier (DPID) is an eight-digit number which uniquely identifies a postal delivery point in Australia.
G-NAF	The G-NAF is a collection of addresses primarily collected from Australia's government mapping agencies and land registries and ultimately generated from all Local Government Authorities. PSMA Australia Limited has the responsibility for building G-NAF. The Geo-coded National Address File (GNAF) is an index of physical Australian addresses, each with a geographic coordinate.
HTTP	HTTP - The language and protocol overlying TCP/IP and used to connect coordinate and orchestrate interactions between Web systems. The HTTP variant HTTPS is equivalent but adds dynamic encryption and decryption to the data stream so that intermediate stages cannot

	intercept and view the data. HTTP supports a number of request types, the most widely used of which are GET and POST.
JSON	JavaScript Object Notation. A text-based structured data format that is easily used by browser-based web applications that use the JavaScript language
NWS	NAMF Web Services
PAF	Postal Address File. In terms of this report the PAF is used generically to mean Australia Post maintained address data, although the actual address data resource provided by Australia Post may be a specific resource appropriately specified to meet NAMF requirements, rather than the existing PAF product.
PSMA Australia	PSMA Australia Limited is an unlisted public company wholly owned by the State, Territory and Australian Governments. The core business of PSMA Australia is facilitating the creation of and access to seamless national spatial data sets for government, industry and community use. See www.psma.com.au for more information.
SOAP	Simple Object Access Protocol. An XML-based standard for encapsulating requests and responses to and from clients and servers. SOAP provides an "Envelope" into which requests and responses can be placed. Servers can then "read" the "envelope" and ensure the request is directed to the correct target.
TCP/IP	Transmission Control Protocol/Internet Protocol. The Packet transport network underlying the Internet.
Web Service	A Web Service is a software system that adheres to a set of conventions to support machine to machine interaction over a network. Further, the set of conventions is typically provided in a machine-interpretable form. In its most usual implementation, a Web Service uses TCP/IP and HTTP as the underlying transfer technology, with XML as the content language, SOAP as the XML encapsulation technology and WSDL as the machine-interpretable language used to describe the interface.
WSDL	Web Services Description Language is an XML-based language that provides a model for describing Web Services.
XML	eXtensible Markup Language. XML is a standards controlled framework for the interchange of data. Its primary purpose is to facilitate the sharing of data across different information systems, particularly via the Internet.
XML Schema	A way to define the structure, content and, to some extent, the semantics of XML documents.

2 Introduction

2.1 Purpose of this document

This document provides a recommendation for a standard Web Services interface for the National Address Management Framework (NAMF) and discusses the issues involved in designing and implementing such an interface. Both postal and location address are required to support the business operations of government, the private sector and the community. The recommended approach means both postal and location address capability will be available in a simple, efficient arrangement, to support the full range of required business activities.

The recommendations in this report build upon the recommendations of the NAMF Address Data Set and the NAMF Address Data Interchange Format reports.

2.2 Current Situation

Address data, in all its forms, is vital for the efficient operation of all public and many private organisations. However it is a particularly difficult form of data to validate, store and use for purposes such as address validation. These difficulties derive from the unstructured and frequently ad-hoc approach to address capture, storage and dissemination.

Many organisations find that access to high quality address data for address validation, parsing and address capture is difficult and costly. No single source of address truth exists and no formal processes exist to enable a feedback of addressing issues in order to increase address accuracy.

Many organisations purchase a licence to use one of the commercial address management products, which are typically provided by vendors along with an address data set. These products have proprietary Web Services interfaces, and so software systems that require access to address services must be implemented according to a specific vendor's proprietary interfaces. This is an impediment to the NAMF goal of enabling an organisation's address data and address management services to be accessed by other organisations. Therefore there is a need for a Web Services interface to be adopted as a standard for address management systems.

At the time of writing this report there was no nationally or internationally-recognised standard definition of Web Services for address management. Therefore one had to be developed for the NAMF.

2.3 Scope

Figure 1 shows the supply chain for address data. The creation of addresses and address quality improvement initiatives, that is, the supply-side shown in the upper part of Figure 1, is *not in scope* of the NAMF.

The processes to make address data available to users such as Government and emergency services organisations, that is, the demand-side shown in the lower half of Figure 1, is *in scope* of the NAMF.

The NAMF is a demand, or user, side strategy. That is, the NAMF will provide a framework to make address management and address operations more efficient and effective, principally through the provision of:

- A single authoritative address data set;
- A national standard for address data storage and interchange; and
- National standards for web services that support common address-related functions (such as address parsing and validation).

These three aspects of the NAMF are shown in Figure 1. The subject of this report is the NAMF Web Services specification.

The NAMF is not a supply side strategy. That is, the NAMF will have no direct impact on existing address creation and maintenance processes[1]. There is a range of ongoing address quality (currency, timeliness, completeness, accuracy) improvement initiatives underway in States and Territories, in Australia Post and in many other organisations.

[1] While the NAMF will not directly alter existing maintenance processes, its implementation will provide the means to significantly improve address quality. The single most important factor will be the automatic, real-time notification (from the NAMF address user to address providers) of address parsing, validation and geo-code failures. Lack of early advice of address errors or omissions is one of the fundamental deficiencies of existing address creation and maintenance processes.

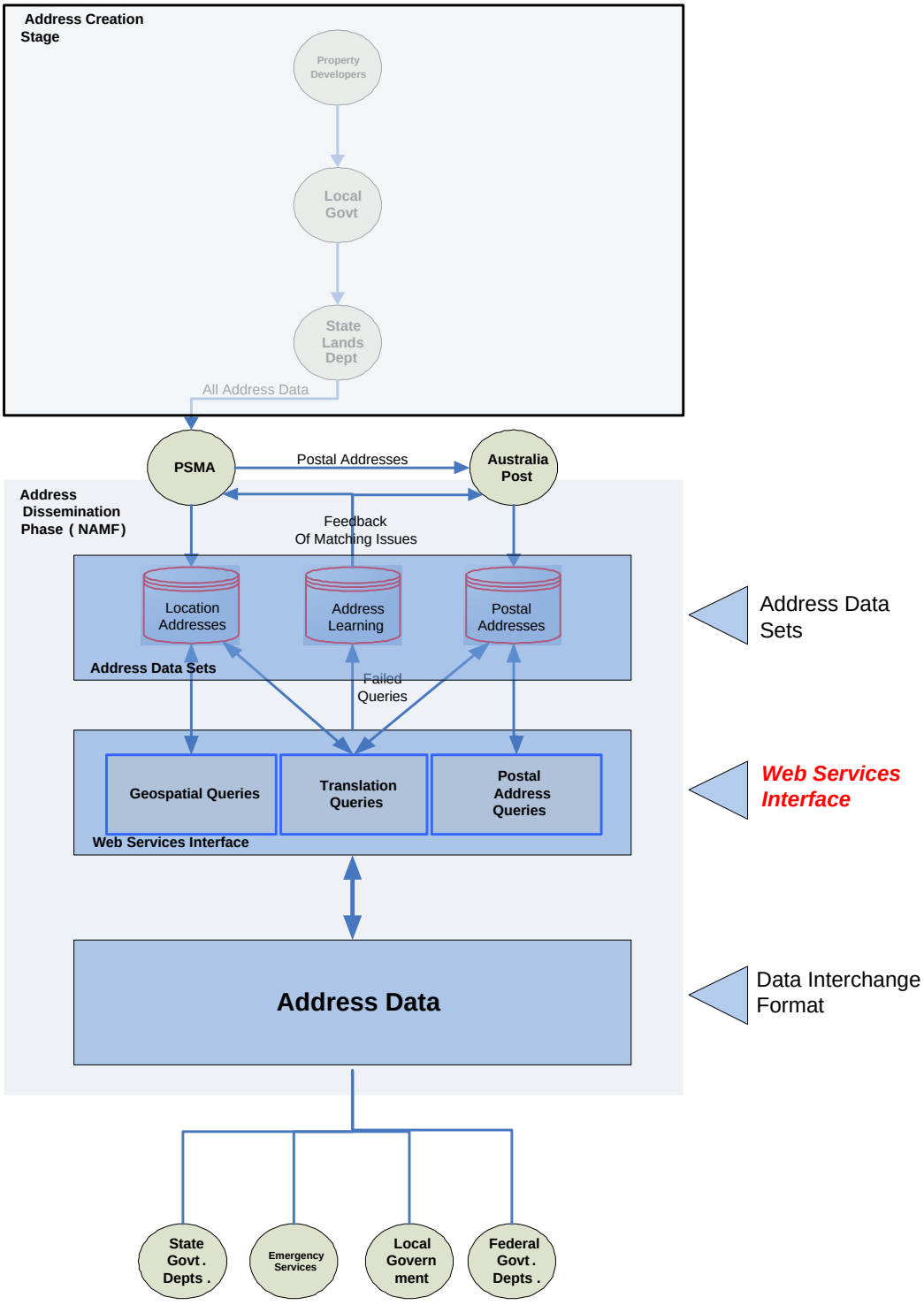


Figure 1: Address Data Supply Chain

2.4 Core Address Management Services

This report specifies a set of Web Services to support the use of address information within the National Address Management Framework.

This report discusses the many technical and usage issues inherent in the design of a Web Service for address management, and develops an overarching framework within which detailed Web Services specifications can be placed.

This report recommends the development of a set of Web Services with the following calls:

validateAddress

This call covers a wide range of address query, search and validation functions. The Web Service user calls validateAddress with one or more Addresses (managed as structured objects) and indicates, using *feature* objects, what the call is to do with them. The call supports address parsing, address validation, geocoding, reverse geocoding and batch operations simply by varying the *feature* arguments presented

listAddress

The *listAddress* call allows users to request lists of address elements by partially populating an address structure. The *listAddress* call can return lists of street numbers, street names, postcodes, suburbs etc. The *listAddress* call is designed to be invoked for "real-time" address validation, such as when a user is entering an address in an application or a web page. For example, the user may type in the first three letters of a street name. A call to listAddress with those three initial street name characters would return all matching streets. These can then be managed on the client, such that when the user enters a fourth character, the list of matching streets is correspondingly reduced, to the eventual point where the list is small enough for the user to pick the correct street name from the list. The same principle could then provide a list of postcodes given a suburb, a list of street numbers given a street name etc.

The listAddress call can be used in any situation where a list of matching choices for partial entry is required. While not restricted to use from an address entry client, it is in this particular use that the call can be most effective. In future, it would also be possible to return a structured data form, such as JSON which would provide a lighter-weight data alternative for browser-based web clients.

notifyAddress

The notifyAddress call is specifically designed to allow feedback on address data. It allows users to send useful information about a particular address which would then be forwarded to the data custodian for evaluation and possible address remediation.

getNotifications

The getNotifications call is specifically designed to retrieve notifications submitted via the notifyAddress call.

getInformation

The *getInformation* call will report information about the specific instance of NAMF Web Services supported by the server. This includes information on the various levels and versions of calls supported the default values for features and notes when they are omitted from a request or response and any other information that could prove helpful in using the service properly.

getFieldNames

The getFieldNames call is specifically designed to retrieve the list of domains supported by the NAMF Server.

getFieldValues

The getFieldValues call is specifically designed to retrieve the list of values for each domain supported by the NAMF Server.

3 Approach

3.1 Investigation Approach

The NAMF Address Data Set report recommended that the G-NAF data set from PSMA Australia, and postal delivery data set available through AMAS accredited providers from Australia Post, be used for the NAMF. These data sets are available from a number of third-party vendors. Some of these vendors also sell address management software that may be applicable to the implementation of NAMF-compliant systems.

The PSMA Australia website lists resellers of their G-NAF data set. A number of these vendors also sell software for address data management. See http://www.psma.com.au/resellers. The Australia Post website has a list of vendors of AMAS certified software (see http://www.auspost.com.au).

Information on the Web Services interfaces provided by a number of commercial address management products in the PSMA Australia and Australia Post list were evaluated. Key personnel in these organisations were contacted and interviewed.

Two state-based whole of government address management frameworks were also examined. These are the Victorian government's VMAS system, and the Queensland government's IQ Address Service.[2]

3.2 Findings

The overall finding is that while some initial work had been undertaken, in comparison with the NAMF Address Data Set and NAMF Address Data Interchange format reports, the use of Web Services for address management is relatively new; substantial implementations are both uncommon and, in all cases, specific to a relatively narrow set of customer requirements. A web search of overseas systems identified a similar lack of comprehensive, standardised and/or government sanctioned Web Services. Most vendors of address management software appear to have adopted Web Services as an additional access point into their existing products.

The existing Web Services interfaces provided by address management systems have mostly been developed to provide a better service to the vendor's own customer base with little or no consideration of a wider user base. Typically, the Web Services provide a front-end into the vendor's existing address management functions and the interface is largely built around feeding the correct arguments to these functions and returning the results.

In designing and implementing their own Web Services, address management software vendors have essentially addressed some of the same issues required of a NAMF-based solution. Examination of these existing proprietary interfaces is therefore a first step in designing the NAMF Web Services specification.

[2] When this report was written the IQ Address Service had not been implemented. Instead, the investigation examined the functional specification of the IQ Address Service.

Further, as these same organisations will be the likely implementers of the NAMF Web Services specification, concordance, where possible, between the NAMF and their proprietary systems is desirable.

There are also some overseas address management experiences that may be useful in designing a set of Web Services.

3.3 Recommendations

Based on these findings, a small but flexible set of Web Services for address management is proposed. The web services are referred to as *NAMF Web Services* (NWS) throughout the balance of this document.

The main call *validateAddress* is designed for use in a wide range of queries involving Address data. The concept of address includes both Postal and Spatial attributes, allowing the same call to be used for accessing a Delivery Point ID as well as Geocoding and Reverse Geocoding. The same call also supports the batch validation of large numbers of addresses.

In addition, the *listAddress* function supports interactive address entry (validating the partial address as it is entered and supplying lists of matching addresses and/or address fragments such that subsequent validation is unnecessary) and a *notifyAddress* call which allows users to provide feedback to the service (and eventually to address data custodians).

The essential difference between the validateAddress and listAddress calls is that the former is designed to work without human interaction, while the latter provides lists of address alternatives which a human would typically then choose from.

Because the NWS will develop in response to user feedback and requirements, an administrative framework has been proposed, consisting of Levels of Web Services (with higher levels providing more functionality) as well as a Version attribute on each call allowing services to provide more than one version of a particular call to their user base.

The NWS needs to utilise both the postal address and G-NAF data sets seamlessly. It needs to support a wide set of address attributes including both spatial and postal delivery functions. In addition, it should support other address related activities such as mail creation, validation, and feedback. The NWS specification should enable future extension to support additional address management services.

In order to control the scope of the report, those discussion topics that are not of immediate utility to the NWS users are provided in a separate section on Future Extensions.

3.4 Address usage

A Web Service consists of a specific request - such as "*validateAddress*" sent from a requesting system to a system providing the service. The request contains information (such as address details a user may have entered) that is used by the service to perform the function. In turn, the server responds with other information (such as a validated address) to the original requestor.

The set of possible Web Service requests should follow the various ways in which people use address information. These include:

Address validation: confirming that an address exists, and has a defined location.

Postal address validation: confirming that a postal article is likely to be successfully delivered to the intended address.

Completing an address - perhaps by adding the correct street ending and post code - again improving the likelihood of the address succeeding.

Assisting in address entry - for example by providing a list of streets for a particular area or for those starting with the letters a user has already entered. This is effectively a way of validating the address in real time - ensuring the address is valid as soon as it is entered.

Locating an address – (geocode) a geographic coordinate can be returned indicating where on the earth's surface the address is to be found. This then enables addresses to be displayed on a map.

Finding an address – (reverse geocode) in a reverse of the address use above, a geographic coordinate can be used to find and return an address.

In addition, large-scale address users will want the ability to submit many requests at once and receive the results back when they have been processed and are available.

Finally, while every attempt is made to keep address information accurate and current, it is inevitable that errors and omissions will occur. By providing a mechanism allowing users to provide notification of address errors, the process of correcting the error by the original address custodian becomes quicker, easier and more likely to be adopted and used.

The general address usage case involves a requestor supplying a small amount of address information and the service using it to find and return further address information. There is a lot of commonality in the address functions that deal with the sending and receiving of addresses. Consequently, a single validation call could be used for a wide variety of address services:

- Supply an unstructured address line for parsing and to return a validated address
- Supply a streetNumber, streetName, locality and postcode and request both a validated address and the geocoded coordinates of the address;
- Supply an address and request an attribute related to that address;

- Supply a set of geographic coordinates and request the address closest to that location;
- Supply a set of 1000 unstructured addresses and request back 1000 validated addresses.

It is with these major address use categories in mind that the NWS command set has been developed.

3.5 Technical Overview

In designing NWS, technical decisions have, as far as possible, sought to include and incorporate existing technologies. For example, the predominant underlying Web Service technologies incorporate XML messages (optionally encapsulated with SOAP) communicating using HTTP. The recommendation is that HTTP be used as underlying transport mechanism with XML as the default structured data language. In addition, support for both a legacy data format (CSV) and an emerging structured data format (JSON) are recommended.

Web Services are designed to support computers talking to each other. Usually, this consists of a server making a request of another server. More recently, however, the initiating computer could be a user's own desktop or laptop computer running a web browser. As modern browsers support XML natively, it is entirely possible for a web page to initiate NWS calls and receive NWS responses directly.

NWS make use of the XML Schemas for AS4590, 2006 version (see the NAMF Address Data Interchange Format report). These schemas, in turn, have been based on the Australian Standard AS4590 or the Interchange of Client Information. Only the Address, Complex and Geocode schemas have been used as other schemas are outside the scope of the NAMF.

4 NWS Design Criteria

There are a number of criteria to which NWS needs to adhere. These have been further divided into issues concerning the implementation ***technologies*** and the ***use*** of the Web Service itself.

4.1 Technical Criteria

4.1.1 Standards-based

NWS should be based on the recognised and de-facto standards that are currently used in existing Web Services. Specifically, these include TCP/IP, HTTP, XML and SOAP. The advantage of SOAP is that it provides a structured way to characterise an applications interface. Using a Web Services Description Language (WSDL) document, a SOAP Client can understand what calls the server supports, the structure the request should be provided in, and the structure of the result. While a set of SOAP wrappers is included in the recommendation, it is not mandated as there are many instances where the extra complexities and costs involved in supporting the SOAP environment may not necessary or justifiable.

4.1.2 Security

As with any other Web Service, the NWS will need to support a security model so that Services can understand who the client is representing and, therefore, which services and information can or cannot be provided. The three key elements in the provision of security include authentication, authorisation and encryption, described in the following sections.

4.1.2.1 Authentication

There are a number of existing technologies for verifying the identity of a particular Web Service requestor. These technologies typically involve a challenge - such as a request for a username or email address and password, and a response - either accepting (and then tracking) the requestor or rejecting them. In the Web Services case, a username and password can be incorporated into each request message.

4.1.2.2 Authorisation

Once authenticated, the NWS should support a mechanism for determining which users are allowed to request which services. Typically, this involves categorising the authenticated clients into groups and allocating privileges for all group members.

4.1.2.3 Encryption

NWS should also support the optional protection of data as it moves across the Web using encryption. The most widely used technology for encryption is HTTPS which automatically encrypts all traffic between network nodes so that the content is not accessible at intermediate transmission points.

In practise, these security issues are both universal enough (in that they are an issue with ALL Web Services) and specific enough (in that different NWS implementers will probably have their own security systems) that they are outside the scope of the NWS itself. The important point is that the NWS, while not mandating particular security mechanisms, should not preclude the incorporation of security information (usernames, passwords, tokens etc) for developers implementing NWS-compliant systems.

In keeping with support for Authentication, the NWS Requests object will support an authentication structure with *username* and *password* fields.

4.2 Usage Criteria

In designing a Web Services standard comprehensive enough to cover the majority of Australian addressing requirements, it is important that flexibility is built into the design. It is impossible to predict every scenario or use case in which it will be required to perform so NWS must be extensible to cope with future addressing demands.

At the same time, NWS needs to be simple enough to be readily incorporated into existing and future address management systems. These two requirements indicate that a range of both implementation levels and versions should be developed.

Some of the design criteria that need to be addressed are described in the following subsections.

4.2.1 Simple Entry Point

While NWS must be able to support complicated addressing requirements, it is even more important that it also supports the very simple address services, such as validation of an address, which will answer the majority of users' requirements.

4.2.2 Data Independent

NWS should be designed such that the data elements apparent to users of the Web Services are not dependent on the underlying data sets. Part of the design should be the creation of a virtual data set that reflects as many aspects of address data as possible and practical. This would then be mapped (by NWS implementers) to both the postal address data set and G-NAF and, potentially, to other address data sets that might be useful in certain specialised situations. For example, an NWS implementer may wish to map some NWS data elements to their own supplementary address data which might contain other address-related attributes. The advantage of a defined, documented, published and enforced standard for web services is that Web Service users can confidently develop applications that adhere to the published interface. It is the Web Service implementers that will map the requests to access the appropriate (and possibly changing) realities of the underlying data sources and structures.

4.2.3 Feedback

An important factor in determining how well a Web Service is received and utilised is the trust users have in the quality of the data. The very nature of address data is that both its creation and the detailed knowledge of the address are widely distributed. Addresses are typically created by developers and/or local government staff, geographically close to where they are used. Knowledge about changes to the address is also local.

One of the potential uses of NWS is in supporting a Web Service that allows suitable authenticated and authorised users to provide feedback to the address data custodians. The support might be in the form of "notes" that can be attached to an address, viewed by the data maintainers, with the ability to reply to the notes etc. It might also be useful to support maps of the address area so that spatial and contextual issues can be highlighted.

The constant activity associated with the live Australian addresses data sets would benefit from a mechanism that supported and encouraged feedback from informed users of errors and omissions, automatically ensuring that the feedback was directed to the custodians of the data in question for evaluation and possible incorporation into the data set.

4.2.4 Lists

NWS should be able to provide lists of enumerated types (such as Street names, localities, cities etc) based on contextual search criteria. For example, one list request might return all streets in a given administrative area (suburb, LGA etc). Another might return all street numbers for a given street.

4.2.5 Integrate Location and Postal addressing environments

It is evident in evaluating the various available commercial Web Service products that they have come from one of two very different starting points. One is that of the Postal Address, Australia Post and the PAF where the address is a means by which mail is delivered. The other is the G-NAF /mapping/spatial starting point where location and spatial context (as in a map) are more important than how one might access the location in order to deliver something. Both are valid aspects of addressing and NWS should provide a seamless interface to both postal and location address data with possible future extension to support other address data sets.

While the postal and location address data sets will remain under separate management and access arrangements, NWS users should not need to aware that there are two data sets - or, in fact, that still other data sets may become involved in servicing a particular request. The NAMF Web Services user should "see" a single, consistent set of attributes for each address, be it a PO Box or a location on the ground.

5 Use Cases

5.1 Overview

The following pages provide a set of Use Cases for the NWS Application Programming Interface (API). The use cases are not intended to be exhaustive but should, together, illustrate how the API can be used. At the same time, the use cases will identify the fitness of the design and indicate areas for improvement.

The simplest case of address validation is of a single address. NWS should provide for the efficient servicing of a single address query, providing the answer as quickly as possible. In practise, the address validation requests are often originating from sophisticated software applications that are the managing thousands of addresses. In this latter case, it is important the Web Service can adapt to manage these large datasets efficiently. This can be accomplished with a number of design features outlined below.

5.2 Multiple requests per request

The design of the messages allows each request to include a number of individual requests - so all requests are designed to handle multiple addresses. The trivial case then becomes a request set with a single request. The responses are returned as a corresponding set of individual responses, one per original request. Note that there is no requirement on the service to return the responses in the same order as the requests. This then allows the service to use parallel execution in order to improve speed, adding responses to the eventual response message as they become available.

To support this multiple request / response communication, it is essential that each request is uniquely identified so that its response can also be identified (as it carries the requests original unique id). The Web Service also supports the ability to customise (through the use of "features") the overall requests message as well as each individual request, the former applying to all requests while the latter can be used to override the global values.

5.3 Synchronous and Asynchronous Communication

Two ways of managing the request-response pair are supported. The first or Synchronous (meaning "Same Time") communication mechanism is that used by most web browsers. A request is sent and the sender then waits for a response. Typically is no response is received with a set time (say 300 seconds) then a "Timeout" error occurs. Synchronous communication is the best option for relatively small requests that can be serviced in a short time. It is also appropriate when the requesting system can't continue until it has received a response.

While this is fine for small requests, it is not appropriate for a system wanting to validate, say, 25,000 addresses. In this case, the web service supports the ability to send a set of requests and then disconnect. At some time later, calling system will

intermittently poll the web service to see whether the request has been processed, and retrieve the response.

5.4 Communication Formats

A number of communication formats are supported by the web service:

5.4.1 HTTP

This is the format used by most web browsing activity and allows text messages to be transmitted reliably across a network. Both the GET and POST encoding types are supported. GET is used for relatively short messages as will be seen in the following user cases. POST is used for all XML-based and Multiple requests.

5.4.2 HTTPS

This is a variant of HTTP in which all data is encrypted before being sent across the network where it is decrypted at the other end. This ensures any eavesdropping on the data as it traverses the network is not decipherable. Users of Internet banking will be familiar with the closed padlock symbol that denotes a session is running over HTTPS. HTTPS uses a technology called SSL to perform the encryption and decryption.

5.5 Data formats

The NWS is expected to support a number of data formats.

5.5.1 Current Support

5.5.1.1 XML

The most common format for structured human-readable data is XML. All service calls will support the reading and writing of XML messages.

5.5.2 Future Support

5.5.2.1 JSON

Client systems (those running directly on a user's browser) are now powerful and sophisticated enough to communicate directly with an address server using HTTP. While modern, javascript-enabled web browsers support XML through a Document Object Model API (DOM). The composing, sending, receiving and parsing of large DOM documents can impose memory and performance constraints on the browser, leading to slow response. An alternative structured data format, JSON, can be converted into JavaScript native structures with a single statement, leading to better performance and responsiveness.

5.5.3 HTTP GET

Small amounts of request information can be placed directly on the URL string that is used to connect to the Address Server. Note that while simple, this format can only support a short string of name=value pairs. As the GET format cannot be encrypted, username/password pairs should not be provided for authorisation. Instead, an encrypted key which includes the domain name of the requesting system should be used.

5.6 Request / Response Modifiers

The NWS API consists of seven calls. To provide a greater granularity of control over how calls are handled, the request can include command modifiers called *features* both at the global <requests> level as well as at the level of each individual <request>.

In order to allow address verification vendors to provide metadata specific to their web services back to caller systems. This metadata can be passed back as *attributes* which are name and value pairs, applicable both at the global <requests> level as well as at the level of each individual <request>.

A similar capability is available in the response where a *notes* section can include multiple *note* elements, either at the global responses level or at the level of each individual response. The notes can be used to provide further relevant information to the requestor.

For example, if a particular address was only matched to an approximate level during a geocode operation, the accuracy of the geocode can be provided through the matchCertaintyMessage, and vendor specific attributes such as flags could be passed back as extra attributes.

5.7 Use Case Elements

5.7.1 Actors

As these use cases involve a set of services, the Actors involved are both humans and IT systems:

Actor	Icon	Description
User		Represents the user of the Address Application.
Browser		Represents an application running in the user's browser
Application Server		Represents the local server running the application
NWS Server		Represents the server that runs the NWS.
File Server		A Source and Destination system for file transfers
Custodian		Represents an organization that creates, collates and manages address data.
Address Dataset		Represents a set of addresses either to be processed or after processing

5.7.2 Data Formats

Data Format	Icon	Description
XML	XML	XML Structured text format
JSON	JSON	JavaScript Object Notation - used for applications with JavaScript.
HTTP GET	GET	HTTP GET String
PDF	PDF	Adobe Portable Document Format.

5.7.3 Message formats

The various messages appearing in the use cases are indicated using coloured boxes:

```
XML request/response format
```

```
JSON request/response format (not currently supported)
```

5.7.4 Transport Protocols

Protocol	Icon	Description
HTTP / HTTPS		HyperText Transport Protocol / Secure HyperText Transport Protocol
Undefined		Could be email, phone etc

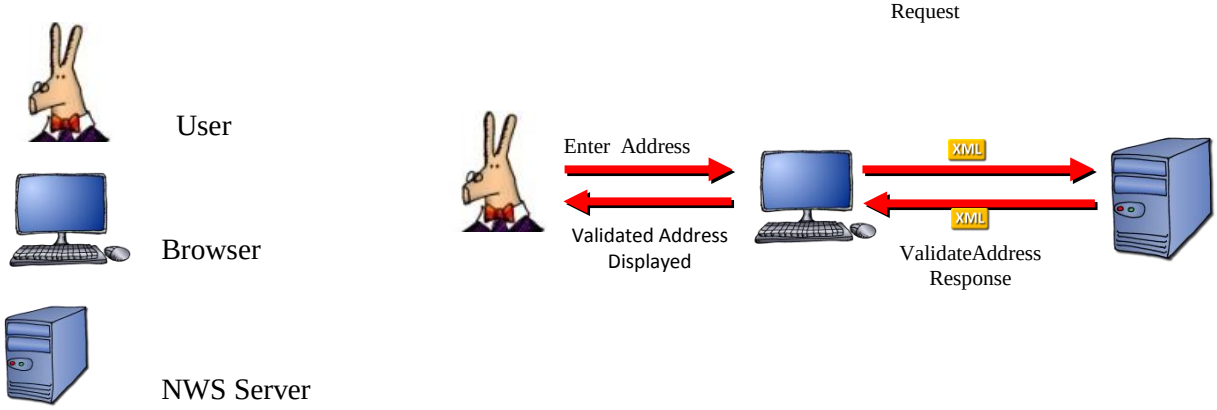
6 Use Cases - Details

6.1 Validate a single address from a browser.

6.1.1 Description

Address user wishes to validate a single address to ensure it is correct. The user is running a web page that communicates directly with an Address Server.

6.1.2 Actors



6.1.3 Scenario

1. *User* is logged on to an application that wishes to confirm that an address entered into the application's web page is valid.
2. *User* enters the address. For example, "*21 King George St Lavender Bay*"
3. The *Browser* displaying the web page takes the entered address details and uses the JavaScript language's Document Object Model (DOM) functions to create an XML *validateAddress* call.

```
<requests id="165744" version="1.0">
  <authentication>
    <username>referenceImplementation</username>
    <password>doTryThisAtHome</password>
  </authentication>
  <features/>
  <request id="165744.1" name="validateAddress">
    <features>
      <feature name="validationType"><featureValue>unstructuredAddress</featureValue>
    </feature>
    <address>
      <unstructuredAddressLine1>21 King George St Lavender Bay</unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
</requests>
```

4. The XML request is sent as an HTTP POST message to the URL of the Address Server.

5. The Address Server receives the incoming request and authenticates the message using the authentication object's username and password.
6. If the authentication fails, an XML Error Object is returned to the Browser.

```
<responses id="165744">
  <result hasErrorsInResponseElements="false" completed="true" status="ERROR"/>
  <error code="2">Invalid Username / Password</error>
</responses><responses id="165744">
  <result hasErrorsInResponseElements="false" completed="true" status="ERROR"/>
  <error code="2">Invalid Username / Password</error>
</responses>
```

7. If the authentication succeeds, the address line is extracted and the address validated.
8. A response object is created with details on the address and returned in a responses object.

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <responseResult>
      <address>
        <addressIdentifier>GANSW705185328</addressIdentifier>
        <status>OFF</status>
        <cadastralIdentifier>7//520996</cadastralIdentifier>
        <streetNumber1>21</streetNumber1>
        <streetName>KING GEORGE</streetName>
        <streetType>ST</streetType>
        <localityName>LAVENDER BAY</localityName>
        <stateTerritory>NSW</stateTerritory>
      </address>
      <attributes>
        <attribute name="matchCertainty"><attributeValue>Full</attributeValue></attribute>
      </attributes>
    </responseResult>
    <status>OK</status>
  </response>
</responses>
```

9. The validated address can then be parsed for address fields (both structured and unstructured) that can be used to populate the address fields of the application.
10. If the address is not found, an appropriate message in a note is returned

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <notes>
      <note priority="INFO" name="addressFound">false</note>
    </notes>
    <status>OK</status>
  </response>
</responses>
```

6.2 Validate a single address from a server.

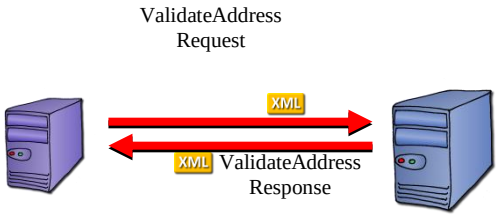
6.2.1 Description

Address user wishes to validate a single address to ensure it is correct. The user is running a web page that communicates with an application Server. The Server communicates with the Address Server and formats the returned address information.

6.2.2 Actors

Application Server

NWS Server



6.2.3 Scenario

1. Communications between the user, the browser and the Server are outside the bounds of this use case.
2. The Server creates an XML request document for a *validateAddress* request.
3. In this case, the server uses structured address fields rather than a single unstructured field. This will allow the Address Server to indicate which field was in error if necessary.

```
<requests id="165744" version="1.0">
  <authentication>
    <username>referenceImplementation</username>
    <password>doTryThisAtHome</password>
  </authentication>
  <features/>
  <request id="165744.1" name="validateAddress">
    <address>
      <streetNumber1>24</streetNumber1>
      <streetName>King George</streetName>
      <localityName>Lavender Bay</localityName>
      <stateTerritory>NSW</stateTerritory>
    </address>
    <attributes/>
  </request>
</requests>
```

4. The XML request is sent as an HTTP POST message to the URL of the Address Server.
5. The Address Server receives the incoming request and authenticates the message using the authentication object's username and password.
6. If the authentication succeeds, the address fields are extracted and the address validated.
7. If no match is found, no result is returned and *optionally* an addressProblem note is provided.

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <notes>
```

```
      <note priority="INFO" name="addressFound">false</note>
      <note priority="INFO" name="addressProblem">streetNumber1 is unknown</note>
    </notes>
    <status>OK</status>
  </response>
</responses>
```

8. If the validation succeeds, the address details are returned in an XML Responses Object

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <responseResult>
      <address>
        <addressIdentifier>GANSW705185329</addressIdentifier>
        <status>OFF</status>
        <cadastralIdentifier>7//520997</cadastralIdentifier>
        <streetNumber1>24</streetNumber1>
        <streetName>KING GEORGE</streetName>
        <streetType>ST</streetType>
        <localityName>LAVENDER BAY</localityName>
        <stateTerritory>NSW</stateTerritory>
      </address>
      <attributes>
        <attribute name="matchCertainty"><attributeValue>Full</attributeValue></attribute>
      </attributes>
    </responseResult>
    <status>OK</status>
  </response>
</responses>
```

6.3 Validate multiple addresses immediately.

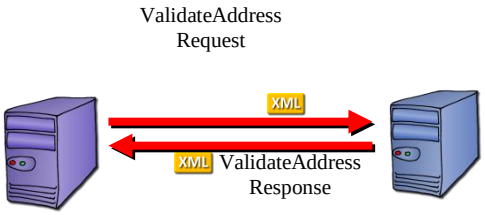
6.3.1 Description

An application Server wishes to validate a set of addresses to ensure they are all correct. The Server communicates with the Address Server and formats the returned address information.

6.3.2 Actors

Application Server

NWS Server



6.3.3 Scenario

1. The Server creates an XML request document for a *validateAddress* request.
2. The request-response pair is synchronous - that is, the request waits until the response is available. If the response takes too long to return, a timeout error can occur. Consequently, this form of the validateAddress request should not include an excessive number of request objects.
3. In this case, the server supplies 5 unstructured addresses.
4. Some of the addresses may contain errors such as:
   - Misspelt street names - *Kign* instead of *King*
   - Street names with the wrong Street Type *King Rd.* instead of *King St.*
   - Addresses with the wrong Postcode (*2061* instead of *2060*)... etc

Because there is no human intervention in the process, the Application Server wants to receive back an address wherever possible, including a "Best Match" address which would be found by applying matching rules that would correct the types of errors described above. In order to activate this option, a feature is included to enable address correction (*minMatchingAccuracy* of partial).

```
<requests id="165744" version="1.0">
  <authentication>
    <username>referenceImplementation</username>
    <password>doTryThisAtHome</password>
  </authentication>
  <features>
    <feature name="validationType">unstructuredAddress</feature>
    <feature name=" minMatchingAccuracy">partial</feature>
  </features>
  <request id="165744.1" name="validateAddress">
    <address>
      <unstructuredAddressLine1>12 King George St Lavender Bay NSW 2060</unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
  <request id="165744.2" name="validateAddress">
    <address>
```

```
      <unstructuredAddressLine1>200 George St Sydney NSW 2000</unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
  <request id="165744.3" name="validateAddress">
    <address>
      <unstructuredAddressLine1>47a Rockladns St Crows Nest NSW 2065</unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
  <request id="165744.4" name="validateAddress">
    <address>
      <unstructuredAddressLine1>30 Collins St </unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
  <request id="165744.1" name="validateAddress">
    <address>
      <unstructuredAddressLine1> Level 1 / 47 Burswood Road Burswood WA 6100
</unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
</requests>
```

5. The XML request is sent as an HTTP POST message to the URL of the Address Server.
6. The Address Server receives and authenticates the incoming request and then processes the individual requests.
7. Where an address is not found, the Server invokes address matching functions that look for misspelt and erroneous addresses. If it finds one or more matches using these rules that also have a matching accuracy of two or less, the one deemed closest to the likely correct address (as determined by the matching algorithms) is returned. A note element is added to this response to indicate the level of accuracy achieved
8. The results for each request are assembled into a responses object and returned to the sender.

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <responseResult>
      <address>
        <addressIdentifier>GANSW56798765</addressIdentifier>
        <streetNumber1>12</streetNumber1>
        <streetName>King George</streetName>
        <streetType>St</streetType>
        <localityName>Lavender Bay</localityName>
        <stateTerritory>NSW</stateTerritory>
        <postcode>2060</postcode>
      </address>
    </responseResult>
    <status>OK</status>
  </response>
  <response id="165744.2">
    <responseResult>
      <address>
        <addressIdentifier>GANSW56798765</addressIdentifier>
        <streetNumber1>200</streetNumber1>
        <streetName>George</streetName>
        <streetType>St</streetType>
        <localityName>Sydney</localityName>
        <stateTerritory>NSW</stateTerritory>
```

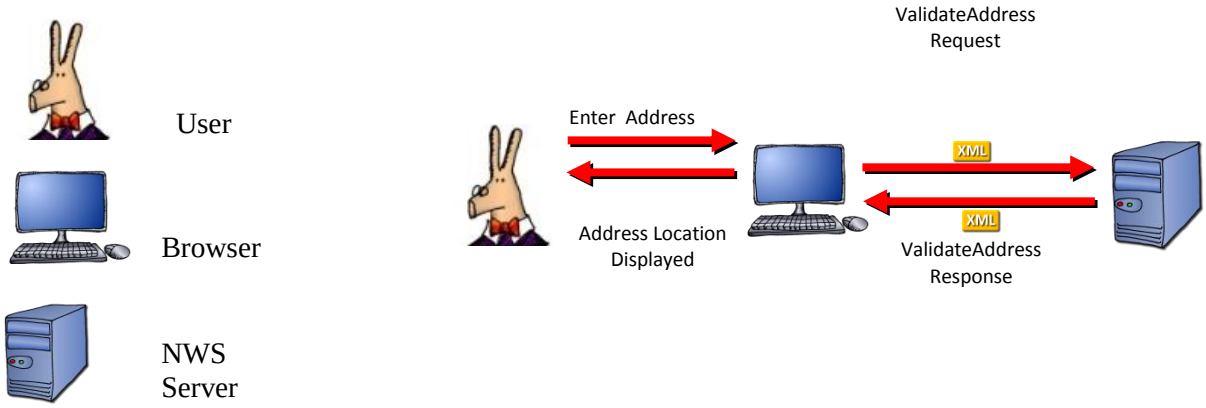
```
        <postcode>2000</postcode>
    </address>
    </responseResult>
    <status>OK</status>
  </response>
  <response id="165744.3">
    <responseResult>
      <address>
      <addressIdentifier>GANSW67987654</addressIdentifier>
        <streetNumber1>47</streetNumber1>
        <streetName>Rocklands</streetName>
        <streetType>Rd</streetType>
        <localityName>Crows Nest</localityName>
        <stateTerritory>NSW</stateTerritory>
        <postcode>2065</postcode>
      </address>
    </responseResult>
    <status>OK</status>
  </response>
  <response id="165744.4">
    <responseResult>
      <address>
        <addressIdentifier>GANSW56798766</addressIdentifier>
        <streetNumber1>31</streetNumber1>
        <streetName>Collins</streetName>
        <streetType>St</streetType>
        <localityName>Sydney</localityName>
        <stateTerritory>NSW</stateTerritory>
      </address>
    </responseResult>
    <responseResult>
      <address>
        <addressIdentifier>GATAS56798766</addressIdentifier>
        <streetNumber1>31</streetNumber1>
        <streetName>Collins</streetName>
        <streetType>St</streetType>
        <localityName>Hobart</localityName>
        <stateTerritory>NSW</stateTerritory>
      </address>
    </responseResult>
    <status>OK</status>
  </response>
  <response id="165744.5">
    <notes>
      <note priority="INFO" name="addressFound">false</note>
      <note priority="INFO" name="addressProblem">streetNumber1 is unknown</note>
    </notes>
    <status>OK</status>
  </response>
</responses>

   : (etc)
</responses>
```

6.4 Geocode a single address from a browser.

6.4.1 Description

Address user wishes to find the geographic location a single address. The user is running a web page that can communicate directly with the NWS.

6.4.2 Actors



6.4.3 Scenario

1. *User* is logged on to an application that wishes to obtain the geographic location of an address entered into the application's web page.
2. *User* enters the address. For example, "*21 King George St Lavender Bay*"
3. The *Browser* displaying the web page takes the entered address details and uses the JavaScript language's Document Object Model (DOM) functions to create an XML *validateAddress* call.

```
<requests id="165744" version="1.0">
  <authentication>
    <username>referenceImplementation</username>
    <password>doTryThisAtHome</password>
  </authentication>
  <features>
    <feature name="geocode">true</feature>
  </features>
  <request id="165744.1" name="validateAddress">
    <address>
      <unstructuredAddressLine1>21 King George St Lavender Bay</unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
</requests>
```

4. The presence of the "geocode" feature indicates that the geocode should be returned.
5. The XML request is sent as an HTTP POST message to the URL of the Address Server.
6. The Address Server receives the incoming request and authenticates the message using the authentication object's username and password.

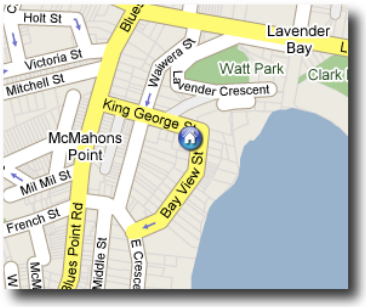
7. If the authentication fails, an XML Error Object is returned to the Browser.

```
<responses id="165744">
  <result hasErrorsInResponseElements="false" completed="true" status="ERROR"/>
  <error code="2">Invalid Username / Password</error>
</responses>
```

8. If the authentication succeeds, the address line is extracted and the address validated.
9. A response object is created with details on the address location and returned in a responses object.

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <responseResult>
      <address>
        <addressIdentifier>GANSW705185328</addressIdentifier>
        <status>OFF</status>
        <cadastralIdentifier>7//520996</cadastralIdentifier>
        <streetNumber1>21</streetNumber1>
        <streetName>KING GEORGE</streetName>
        <streetType>ST</streetType>
        <localityName>LAVENDER BAY</localityName>
        <stateTerritory>NSW</stateTerritory>
        <geoFeature>Property centroid</geoFeature>
        <geoDatumCode>EPSG:4326</geoDatumCode>
        <geoNorthSouthCoordinate>-33.844528</geoNorthSouthCoordinate>
        <geoEastWestCoordinate>151.20609</geoEastWestCoordinate>      </address>
      <attributes>
        <attribute name="matchCertainty"><attributeValue>Full</attributeValue></attribute>
      </attributes>
    </responseResult>
    <status>OK</status>
  </response>
</responses>

    </address>
  </response>
</responses>
```

10. The coordinates can then be used to show the location of the address on a map



11. If the address is not found, an appropriate note is returned instead

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <notes>
```

```
      <note priority="INFO" name="addressFound">false</note>
    </notes>
    <status>OK</status>
  </response>
</responses>
```

6.5 Geocode a single address from a server.

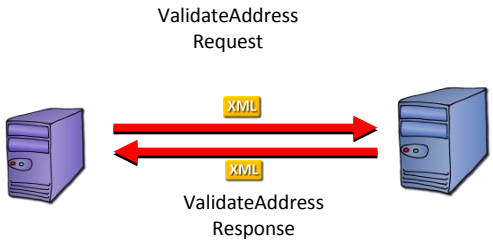
6.5.1 Description

Address user wishes to find the geographic location for a single address. The user is running a web page that communicates with an Application Server. The Server communicates with the Address Server and formats the returned address information.

6.5.2 Actors

Application Server

Address Server



6.5.3 Scenario

1. Communications between the user, the browser and the Server are outside the bounds of this use case.
2. The Server creates an XML request document for a *validateAddress* request.
3. In this case, the server uses structured address fields rather than a single unstructured field. This will allow the Address Server to indicate which field was in error if necessary.

```
<requests id="165744" version="1.0">
  <authentication>
    <username>referenceImplementation</username>
    <password>doTryThisAtHome</password>
  </authentication>
  <features>
    <feature name="geocode">true</feature>
    <feature name="omitAddress">true</feature>
    <feature name="validationType">unstructuredAddress</feature>
  </features>
  <request id="165744.1" name="validateAddress">
    <address>
      <unstructuredAddressLine1>21 King George St Lavender Bay</unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
</requests>
```

4. The XML request is sent as an HTTP POST message to the URL of the Address Server.
5. The Address Server receives the incoming request and authenticates the message using the authentication object's username and password.
6. If the authentication succeeds, the address fields are extracted and the address validated.

7. If no match is found no results are returned

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <notes>
      <note priority="INFO" name="addressFound">false</note>
    </notes>
    <status>OK</status>
  </response>
</responses>
```

8. If the validation succeeds, the location details are returned in an XML Responses Object. As the user includes the omitAddress feature, the textual address fields do not appear in the response.

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <responseResult>
      <address>
        <geoFeature>Property centroid</geoFeature>
        <geoDatumCode>EPSG:4326</geoDatumCode>
        <geoNorthSouthCoordinate>-33.844528</geoNorthSouthCoordinate>
        <geoEastWestCoordinate>151.20609</geoEastWestCoordinate>
      </address>
      <attributes>
        <attribute name="matchCertainty"><attributeValue>Full</attributeValue></attribute>
      </attributes>
    </responseResult>
    <status>OK</status>
  </response>
</responses>
```

6.6 Geocode multiple addresses

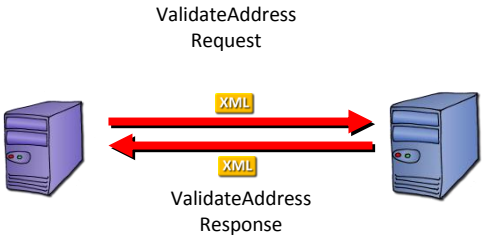
6.6.1 Description

An application Server wishes to obtain geographic locations for a set of five addresses. It already has the unique Address Identifiers from a previous request for three of the addresses and so only wants the geographic information for those. It only has the unstructured address line for the other two addresses and so wants both address and geographic information for those addresses.

6.6.2 Actors

Application Server

NWS Server



6.6.3 Scenario

1. The Server creates an XML request document for a *validateAddress* request.
2. The request-response pair is synchronous - that is, the request waits until the response is available. If the response takes too long to return, a timeout error can occur. Consequently, this form of the validateAddress request should not include an excessive number of request objects.
3. In this case, the server supplies 3 addresses as simply the addressIdentifier while the other two addresses are supplied as unstructured lines. In order to indicate what the NWS is to do, it includes features at both the global (*requests*) level as well as features at the individual request level. Feature at the individual *request* level override features with the same name at the global or *requests* level.

```
<requests id="165744" version="1.0">
  <authentication>
    <username>referenceImplementation</username>
    <password>doTryThisAtHome</password>
  </authentication>
  <features>
    <feature name="geocode">true</feature>
    <feature name="omitAddress">true</feature>
  </features>
  <request id="165744.1" name="validateAddress">
    <features>
      <feature name="validationType">addressIdentifier</feature>
    </features>
    <address>
      <addressIdentifer>1234567</addressIdentifer>
    </address>
    <attributes/>
  </request>
  <request id="165744.2" name="validateAddress">
```

```
    <features>
      <feature name="validationType">addressIdentifier</feature>
    </features>
    <address>
      <addressIdentifer>2345678</addressIdentifer>
    </address>
    <attributes/>
  </request>
  <request id="165744.3" name="validateAddress">
    <features>
      <feature name="validationType">addressIdentifier</feature>
    </features>
    <address>
      <addressIdentifer>3456789</addressIdentifer>
    </address>
    <attributes/>
  </request>
  <request id="165744.4" name="validateAddress">
    <features>
      <feature name="omitAddress">false</feature>
      <feature name="validationType">unstructuredAddress</feature>
    </features>
    <address>
      <unstructuredAddressLine1>30 Collins St</unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
  <request id="165744.5" name="validateAddress">
    <features>
      <feature name="omitAddress">false</feature>
      <feature name="validationType">unstructuredAddress</feature>
    </features>
    <address>
      <unstructuredAddressLine1>47 Burswood Road Burswood WA 6100</</unstructuredAddressLine1>
    </address>
    <attributes/>
  </request>
</requests>
```

4. The XML request is sent as an HTTP POST message to the URL of the Address Server.
5. The Address Server receives and authenticates the incoming request and then processes the individual requests.
6. The results for each request are assembled into a responses object and returned to the sender.

```
<responses id="165744">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="165744.1">
    <responseResult>
      <address>
        <geoFeature>Property centroid</geoFeature>
        <geoDatumCode>EPSG:4326</geoDatumCode>
        <geoNorthSouthCoordinate>-33.844528</geoNorthSouthCoordinate>
        <geoEastWestCoordinate>151.20609</geoEastWestCoordinate>
      </address>
      <attributes>
        <attribute name="matchCertainty"><attributeValue>Full</attributeValue></attribute>
      </attributes>
    </responseResult>
    <status>OK</status>
  </response>
  <response id="165744.2">
    <responseResult>
      <address>
        <geoFeature>Property centroid</geoFeature>
```

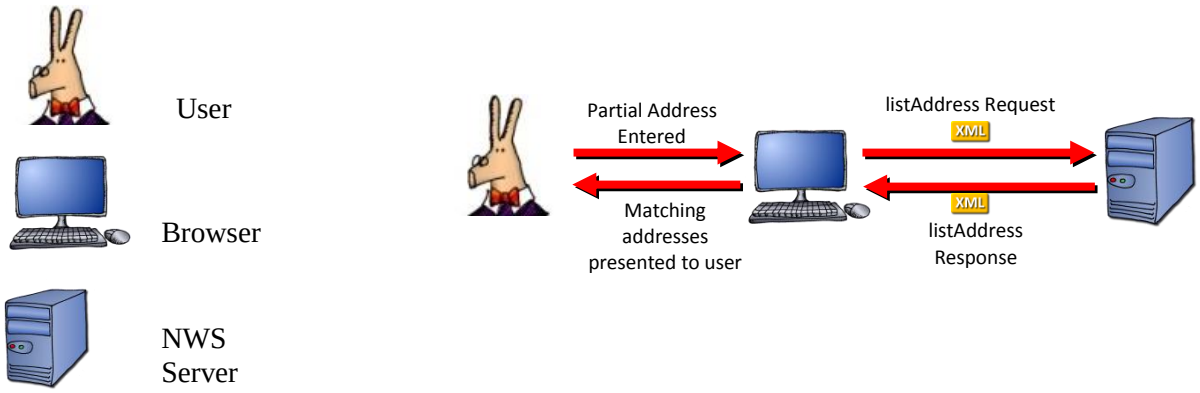
```
        <geoDatumCode>EPSG:4326</geoDatumCode>
        <geoNorthSouthCoordinate>-33.844528</geoNorthSouthCoordinate>
        <geoEastWestCoordinate>151.20609</geoEastWestCoordinate>
      </address>
      <attributes>
        <attribute name="matchCertainty"><attributeValue>Full</attributeValue></attribute>
      </attributes>
    </responseResult>
    <status>OK</status>
  </response>
  <response id="165744.3">
    <notes>
      <note priority="INFO" name="addressFound">false</note>
    </notes>
    <status>OK</status>
  </response>
  <response id="165744.4">
    <responseResult>
      <address>
        <addressIdentifier>43567987</addressIdentifier>
        <streetNumber1>21</streetNumber1>
        <streetName>King George</streetName>
        <streetType>St</streetType>
        <localityName>Lavender Bay</localityName>
        <stateTerritory>NSW</stateTerritory>
        <postcode>2060</postcode>
        <geoFeature>Property centroid</geoFeature>
        <geoDatumCode>EPSG:4326</geoDatumCode>
        <geoNorthSouthCoordinate>-33.844528</geoNorthSouthCoordinate>
        <geoEastWestCoordinate>151.20609</geoEastWestCoordinate>
      </address>
      <attributes>
        <attribute name="matchCertainty"><attributeValue>Full</attributeValue></attribute>
      </attributes>
    </responseResult>
    <status>OK</status>
  </response>
  <response id="165744.5">
    <responseResult>
      <address>
        <addressIdentifier>354687089</addressIdentifier>
        <streetNumber1>21</streetNumber1>
        <streetName>King George</streetName>
        <streetType>St</streetType>
        <localityName>Lavender Bay</localityName>
        <stateTerritory>NSW</stateTerritory>
        <postcode>2060</postcode>
        <geoFeature>Property centroid</geoFeature>
        <geoDatumCode>EPSG:4326</geoDatumCode>
        <geoNorthSouthCoordinate>-33.844528</geoNorthSouthCoordinate>
        <geoEastWestCoordinate>151.20609</geoEastWestCoordinate>
      </address>
      <attributes>
        <attribute name="matchCertainty"><attributeValue>Full</attributeValue></attribute>
      </attributes>
    </responseResult>
    <status>OK</status>
  </response>
</responses>

      <geoFeature>Property centroid</geoFeature>
      <geoDatumCode>EPSG:4326</geoDatumCode>
      <geoNorthSouthCoordinate>-33.844528</geoNorthSouthCoordinate>
      <geoEastWestCoordinate>151.20609</geoEastWestCoordinate>
    </address>
</responses>
```

6.7 List Addresses matching a single template.

6.7.1 Description

An address user is entering an address into fields on a web browser form. The browser application would like to prompt the user with possible correct answers as she is entering the information. The application waits until sufficient characters have been added and then formats a request to the NWS. The list of resulting matches is presented to the user, giving her an opportunity to select one to complete the field and, possibly, other subsequent fields.

6.7.2 Actors



6.7.3 Scenario

1. *User* is filling out a street name field in a form on a browser-based application. The application asks the NWS for a list of matching street names once the user has entered at least three characters. To ensure only a reasonable number of streets are returned, either a localityName or stateTerritory must also be present in the request.
2. The *Browser* displaying the web page takes the entered address details and uses the JavaScript language's Document Object Model (DOM) functions to create an XML *validateAddress* call.

```
<requests id="73934" version="1.0">
   <authentication>
      <username>referenceImplementation</username>
      <password>doTryThisAtHome</password>
   </authentication>
   <features />
   <request id="73934.1" name="listAddress">
      <features>
         <feature name="minMatchingAccuracy">
            <featureValue>full</featureValue>
         </feature>
      </features>
      <address>
         <streetName>MO*</streetName>
         <localityName>PARKES</localityName>
      </address>
```

```
      <attributes>
         <attribute name="listField">
            <attributeValue>streetName</attributeValue>
         </attribute>
      </attributes>
   </request>
</requests>
```

12. The XML request is sent as an HTTP POST message to the URL of the Address Server.
13. The Address Server receives the incoming request and authenticates the message using the authentication object's username and password.
14. If the authentication fails, an XML Error Object is returned to the Browser.

```
<responses id="73934">
  <result hasErrorsInResponseElements="false" completed="true" status="ERROR"/>
  <error code="2">Invalid Username / Password</error>
</responses>
```

3. If the authentication succeeds, the fields supplied in the address object are used to look up matching street names. In this case, the search will be for all addresses in the localities of Parkes where streetName starts with "*MO*".
4. A response object is created with a collection of attributes that contains the matching street names.

```
   <responses id="73934">
      <result completed="true" status="OK" hasErrorsInResponseElements="false" />
      <response id="73934.1">
         <responseResult>
            <attributes>
               <attribute name="MOLONG" />
               <attribute name="MONASTRY" />
               <attribute name="MONICA" />
               <attribute name="MOON" />
               <attribute name="MOOR" />
               <attribute name="MOSSGIEL" />
               <attribute name="MOULDEN" />
            </attributes>
         </responseResult>
         <status>OK</status>
      </response>
   </responses>
```

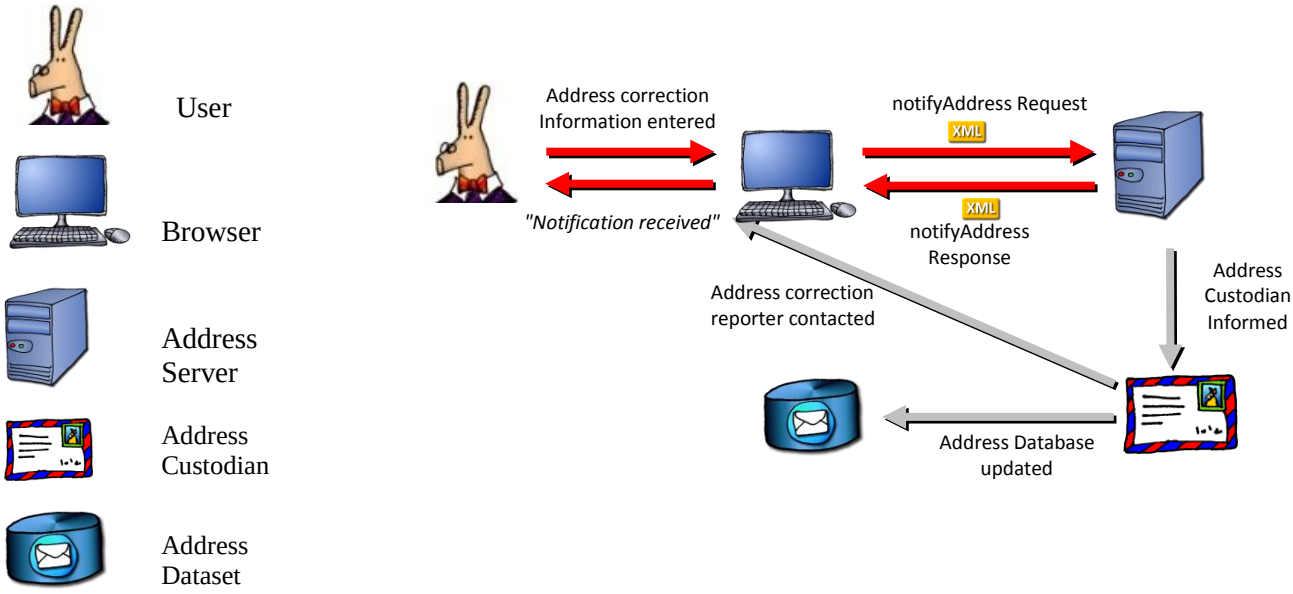
5. On the client, the XML response is evaluated and the array of street names might be transformed into a JavaScript array. As the user enters more characters, the list of street names can be reduced, to the eventual point where either one address remains (in which case, the street name field can be populated with that street name) or no matches are found (in which case an error message can be presented to the user). Alternatively, the user can select from a displayed list of street name alternatives.

6.8 Send information about a single address.

6.8.1 Description

An address user has found what they believe to be an error in an address. They access a browser-based application maintained by the NWS provider. If they are looking to correct an existing enter enough information to uniquely identify an existing address. If they wish to highlight a missing address, they enter enough information to help the address custodians identify the problem. In order to help resolution of the problem, the user may optionally include contact information so custodial staff can seek further clarification.

6.8.2 Actors



6.8.3 Scenario

1. *User* notices an address has an incorrect street name. In the database it is denoted as "Manoff Rd" when it is, in fact, "Manor Place"
2. User accesses the web page for a NWS provider that includes a form for notification of address errors. He enters the unique addressIdentifier and indicates that the Street Name and Street Type are incorrect. He also fills in a textual description of the error.
3. The web page creates a call to the NWS and provides the entered information in an XML structure.

```
<requests id="137263" version="1.0" >
  <authentication>
    <username>referenceImplementation</username>
    <password>doTryThisAtHome</password>
  </authentication>
  <features/>
  <request id="137263.1" name="notifyAddress">
```

```
    <address>
      <streetNumber1>595</streetNumber1>
      <streetName>NAMF TEST</streetName>
      <streetType>RD</streetType>
      <localityName>BRISBANE</localityName>
      <stateTerritory>QLD</stateTerritory>
      <postcode>4000</postcode>
    </address>
    <attributes>
      <attribute name="notificationType"><attributeValue>X_COMMENT</attributeValue></attribute>
      <attribute name="certainty"><attributeValue>full</attributeValue></attribute>
      <attribute name="comments"><attributeValue> At 115.8034,-31.79134 you have a street (off
Waterview Drive) as "Manoff Rd" when it is, in fact, "Manor Place"

Context is the suburb of Woodvale, north of Perth.

Regards

Peter Bayley
OpenEarth </attributeValue></attribute>
    </attributes>
  </request>
</requests>
```

4. The XML request is sent as an HTTP POST message to the Address Server.
5. The Address Server receives the incoming request and authenticates the message by decoding the key checking that it is valid and that the source domain name (nws.openearth.com.au in the example above) matches that of the requesting server.
6. If the authentication fails, an XML Error Object is returned to the Browser.

```
<responses id="137263">
  <result hasErrorsInResponseElements="true" completed="true" status="OK"/>
  <response id="137263.1">
    <status>ERROR</status>
    <error code="103">This server does not support the validateAddress request</error>
  </response>
</responses>
```

7. If the authentication succeeds, a message is returned to the requester acknowledging receipt of the message.

```
<responses id="137263">
  <result completed="true" status="OK" hasErrorsInResponseElements="false"/>
  <response id="137263.1">
    <responseResult>
      <attributes>
        <attribute name="notificationID"><attributeValue>678766</attributeValue></attribute>
        <attribute name="effectiveDate"><attributeValue>21/06/2010</attributeValue></attribute>
      </attributes>
    </responseResult>
    <status>OK</status>
  </response>
</responses>
</responses>
```

8. The information can then be sent to the custodian of the data although these communications are outside the scope of the NAMF Web Service. For example, the NWS server might look

up the authoritative custodian of that particular address (identified by the addressIdentifier value) and send them an email or a fax message.

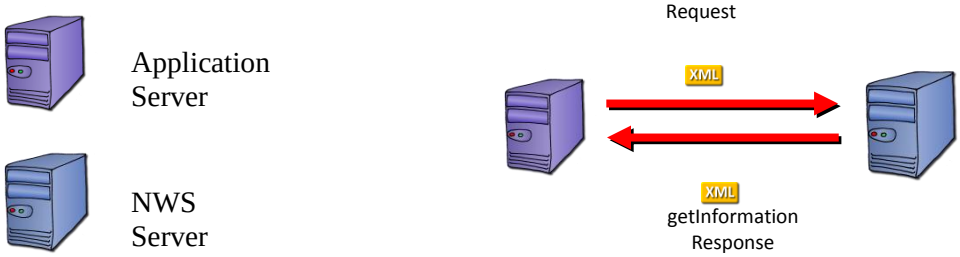
9. The Custodians can use the contact information included to contact the original notifier. Again, this communication falls outside the scope of the NAMF Web Service.

6.9 Request information about the address service.

6.9.1 Description

An Application Server wishes to make an address service available for its users but initially wants to find out about the service. What versions of the various calls are supported, what default values are used when they are not present in a message, etc. It forms a URL to the NWS with GET parameters requesting information.

6.9.2 Actors



6.9.3 Scenario

1. *User* requests information on a NWS Service. This might be through a web service invocation tool, such as soapui.
2. The following request might be made:

```
<requests id="12345" version="1.0">
   <authentication>
      <username>referenceImplementation</username>
      <password>doTryThisAtHome</password>
   </authentication>
   <features />
   <request id="12345.1" name="getInformation">
      <features />
   </request>
</requests>
```

3. The NWS responds with a standard XML document that describes the services. The Web Service Developers Guide provides a full example of what this response might look like.

7 NAMF Web Services Developers Guide

The NAMF Web Services Developers Guide provides a comprehensive description of how to use the NAMF Web Services, and what they should look like. It is intended for use by developers of NAMF Web Services and client applications, and covers the use of all functions described in this document.

8 Future Extensions

This section discusses a set of technical and usage issues which are out of scope for this report, but are recorded here for future reference.

8.1 Technical Extensions

8.1.1 Emerging Standards

The following are emerging technologies which, while not yet standardised, are present in the more-innovative web applications and are likely to experience a strong growth in popularity and application. It is intended that future releases of the NAMF Web Services Specification will support the use of JSON.

- **JSON -** JavaScript Object Notation provides a lightweight, efficient means of structuring and transferring data. JSON can be used as a replacement for XML, especially when interacting directly with Javascript-driven browser-based applications. Native support for JSON within JavaScript is scheduled for inclusion in the next release of ECMAScript - the standard programming language more commonly known as JavaScript.
- **AJAX** - Asynchronous JavaScript and XML is a set of technologies which together provide a more-interactive and intuitive user experience for web-based applications. Traditionally, web pages have completely reloaded from the server for each major user interaction, especially where more data needs to be fetched from the server. Using AJAX, the client can transparently access further data and then format and display it when it arrives.
- **REST** - REpresentational State Transfer is a set of conventions and architectural constraints that formalise how the web and its interactions should be structured. REST principles should be applied in developing the NAMF Web Services. An introduction to REST is available at: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Web Technologies Example

By way of illustration of the emerging standards described above, readers directed to the *"Google Suggest"* demonstrator at:

http://www.google.com/webhp?complete=1&hl=en

This application allows a user to type into an input field. As the user types, the application dynamically displays matching topics in a list below the input field. This application demonstrates:

- **AJAX** using **HTTP GET** server request (containing the letters you type). The request for matching strings is sent without the need to interrupt the user or change the web page
- **JSON** - the return string is a set of matching strings in a JavaScript array
- **REST** - the application is REST compliant

Obviously, a similar application could provide real-time address matching and lookup for users entering an address in a web page.

8.2 Usage Extensions

8.2.1 Federation

An extension to the NAMF Web Services that may be useful in future is for one NAMF-compliant address management system to support the ability to call other NAMF-compliant address management systems in search of further address information. This could be useful as different NAMF-compliant systems may contain data for different jurisdictions (e.g. a NAMF-compliant system in one state may contain the postal and location address data for just that state). From the original requestor's point of view, they are unaware that multiple NAMF-compliant address management systems have worked co-operatively in order to satisfy the query.

8.2.2 Presentation

The NAMF Web Services may, in future, support the ability to present the results of address requests in presentation formats that best suit the eventual use of the data. For example, if the ultimate use of the data is in the printing of address labels, the request could contain information on the size of labels being used and the service could return rendered text (in HTML/CSS or RTF or even Image format) which could be used to print labels directly. If an image is returned, it could include DPID-based barcodes and other customer information, thus eliminating the extra effort required to create the bar codes from returned text. The Image presentation option could also be used to format International addresses in different character sets.

Another presentation option might be an image which shows a map either centred on the address or in the case of multiple addresses, scaled to fit all addresses at once.

Yet another presentation option could be an audio file that speaks the validated address for disabled users or users without English language skills.

8.2.3 Editing

The NAMF Web Services could also provide a set of calls that would allow suitably authenticated and authorised users to make changes either to the current "snap-shot" of an address data set or to the source (authoritative) data sets themselves.

8.2.4 Address Creation

Much of the benefit of a National Address Management Framework will be found in better management of the quality of the address data sets. One of the most efficient ways of ensuring this quality is by controlling the address at the time of its creation. Currently this activity occurs in the geographically-dispersed periphery of the address system, undertaken by property developers and/or local government planning staff. The NAMF Web Services could provide a set of Web Services to support this address

creation activity, checking adjoining localities, for example, for similar street names, checking address numbering conforms to prevailing standards, etc. The Web Service could be implemented as a simple Web page.

8.2.5 Support International address functions

While the majority of the NAMF Web Services activity will concern Australian users accessing Australian addresses, a truly national approach may also address how Australian addresses might cooperate internationally, as described in the following subsections.

8.2.5.1 Australian user validating an International address

It may be useful for a NAMF-compliant system to pass an address to a foreign address validation service. A simple example would be the ability to validate a New Zealand address for an international mail item.

8.2.5.2 International user validating an Australian address

It may be useful to make the NAMF-compliant systems accessible to overseas users, for example to validate an Australian address. For example, a New Zealand addressing service should be able to access an Australian NAMF-compliant system to validate an Australian address. In this future extension it would be important to support and international standard for address data interchange, such as the OASIS xAL.