

**Real Estate Data Interchange Standard:
Real Estate Transaction Specification
Version 2.0
Release Candidate 6**

RETS2 Web Service

July 27th, 2006

Dedication

We dedicate this work to Bruce Toback (1957 – 2005), the first RETS work group technical chairman, whose leadership, energy and passion for excellence nurtured and guided the growth of this vital standard.

Authors

This document was authored by Bruce Toback (OPT Inc.), Paula O'Brien (Falcon Technologies Corp., formerly of Avantia Inc.), Jeff Brush (Falcon Technologies Corp., formerly of Avantia Inc.), Paul Stusiak (Falcon Technologies Corp.)

Table of Contents

| | |
|--|------------|
| 1 Purpose | 1-1 |
| 2 Notational Conventions | 2-1 |
| 2.1 Terminology Conventions | 2-1 |
| 2.2 Typographic Conventions | 2-1 |
| 2.3 Rules | 2-2 |
| 3 Web Services in the Real Estate Context - an Introduction | 3-1 |
| 3.1 Context | 3-1 |
| 3.2 Advantages of Web Services | 3-1 |
| 3.3 Disadvantages of Web Services | 3-2 |
| 3.4 SOAP | 3-2 |
| 3.5 Document Model | 3-2 |
| 3.6 The Resource, Vocabulary & Payload Model | 3-3 |
| 4 Web Services Specifications | 4-1 |
| 4.1 Basic Web Services Components | 4-1 |
| 4.2 SOAP Message Transmission Optimization Method | 4-1 |
| 4.3 Web Services Security in RETS2 | 4-1 |
| 5 Description and Discovery | 5-1 |
| 5.1 Service Registry | 5-1 |
| 5.2 Web Services Description Language (WSDL) | 5-1 |
| 5.2.1 RETS2 Service WSDL | 5-1 |
| 5.2.2 RETS2 Service Extensions WSDL | 5-1 |
| 6 Service Vocabularies and Responses | 6-1 |
| 6.1 Soap Headers | 6-1 |
| 6.1.1 Request Headers | 6-1 |
| 6.1.2 Response Header | 6-1 |
| 6.2 XML Formats and Versioning | 6-1 |
| 6.2.1 XML Based | 6-1 |
| 6.2.2 XML Request Formats | 6-1 |
| 6.2.3 XML Response Formats | 6-1 |
| 6.2.4 XML Response Document Versioning | 6-1 |
| 6.2.5 Unknown Elements and Attributes | 6-2 |
| 6.3 RETS Manifest | 6-2 |
| 6.3.1 ManifestType Type Schema | 6-3 |
| 6.4 Resource Discovery - Metadata | 6-4 |
| 6.4.1 ResourceList | 6-5 |
| 6.4.2 Vocabulary | 6-12 |
| 6.4.3 DataDictionary | 6-21 |
| 6.4.4 LookupList | 6-24 |
| 6.4.5 Formats and Extensions to Metadata | 6-25 |
| 6.5 The UserInfo Resource and UserInformation Response | 6-26 |
| 6.6 Message of the Day | 6-28 |
| 7 Metadata Actions | 7-1 |
| 7.1 GetResourceList - Request Document | 7-1 |
| 7.1.1 Examples | 7-1 |
| 7.2 GetLookupList - Request Document | 7-2 |
| 7.2.1 Examples | 7-2 |
| 7.3 GetMetadata - Request Document | 7-3 |
| 7.3.1 Examples | 7-4 |

| | |
|--|-------------|
| 7.4 ObjectResources and ObjectReferenceLists | 7-5 |
| 8 Update Action | 8-1 |
| 8.1 Update Request | 8-1 |
| 8.2 UpdateResponse | 8-2 |
| 9 Search Action | 9-1 |
| 9.1 OutputObject Types | 9-1 |
| 9.1.1 Common General Properties of Delimited and Encoded-Delimited OutputObjects | 9-1 |
| 9.1.2 Delimited OutputObject | 9-1 |
| 9.1.3 Encoded-Delimited OutputObject | 9-2 |
| 9.2 Search Request Document | 9-2 |
| 9.3 Supported Query Types | 9-12 |
| 10 Error Handling | 10-1 |
| 10.1 Supported Faults | 10-1 |
| 10.1.1 Authentication and Authorization Faults | 10-2 |
| 10.1.2 SearchFaults | 10-2 |
| 10.1.3 UpdateFaults | 10-4 |
| 10.1.4 MetadataFaults | 10-5 |
| 11 Message Transports | 11-1 |
| 11.1 Soap over Http – Request and Response | 11-1 |
| 11.1.1 Data Compression in Rets Transactions over Http | 11-1 |
| 11.1.2 Secure HTTP | 11-1 |
| 12 Security | 12-1 |
| 13 Service Compliance | 13-1 |
| 14 RQL Language | 14-1 |
| 15 Acknowledgments | 15-1 |
| 16 Bibliography | 16-1 |
| 17 Footnotes | 17-1 |
| 18 Glossary | 18-1 |
| 19 Appendix A – Supporting Documents, Schema and WSDL Files | 19-1 |
| 20 Appendix B - Requirements Summary | 20-1 |
| 21 Appendix C – RETS 2 WSDL | 21-1 |

List of Figures

| | | |
|-------------|--|------|
| Figure 4.1 | Username Token Fragment Using Digest | 4-2 |
| Figure 6.1 | ManifestType Fragment. | 6-3 |
| Figure 6.2 | Sample ResourceList Metadata Document | 6-6 |
| Figure 6.3 | ResourceList ResourceName Fragment | 6-8 |
| Figure 6.4 | ResourceList Resource Fragment | 6-8 |
| Figure 6.5 | ResourceList InformationURL Fragment | 6-8 |
| Figure 6.6 | ResourceList VocabularyName Fragment | 6-9 |
| Figure 6.7 | ResourceList OutputFormats Fragment | 6-9 |
| Figure 6.8 | ResourceList OutputFormat Fragment. | 6-10 |
| Figure 6.9 | ResourceList InputFormats Fragment | 6-11 |
| Figure 6.10 | ResourceList MetadataFormats Fragment | 6-11 |
| Figure 6.11 | Sample Vocabulary Metadata | 6-14 |
| Figure 6.12 | Vocabulary Timestamp Change Fragment. | 6-16 |
| Figure 6.13 | Figure 19 - Vocabulary Field Fragment. | 6-17 |
| Figure 6.14 | Vocabulary Field LookupName Fragment | 6-17 |
| Figure 6.15 | Vocabulary RequiredGroups Fragment | 6-18 |
| Figure 6.16 | Vocabulary RequiredGroup Fragment | 6-19 |
| Figure 6.17 | Vocabulary DisplayGroups Fragment | 6-19 |
| Figure 6.18 | Vocabulary DisplayGroup Fragment | 6-20 |
| Figure 6.19 | Sample DataDictionary Metadata | 6-23 |
| Figure 6.20 | Sample LookupList | 6-25 |
| Figure 6.21 | Schema Extension Point Fragment. | 6-25 |
| Figure 6.22 | Sample ResourceList with UserInfo Resource | 6-26 |
| Figure 6.23 | Sample UserInformation Fragment. | 6-27 |
| Figure 6.24 | Sample ResourceList with MessageOfDay Resource | 6-28 |
| Figure 6.25 | Sample MessageOfDay Fragment | 6-29 |
| Figure 7.1 | ResourceListRequest Fragment. | 7-1 |
| Figure 7.3 | Example ResourceListRequest document for a specific Resource | 7-1 |
| Figure 7.4 | Example ResourceListRequest document for a Timestamp | 7-2 |
| Figure 7.5 | LookupListRequest Fragment | 7-2 |
| Figure 7.6 | LookupListRequest for all LookupLists | 7-2 |
| Figure 7.7 | LookupListRequest for a specific LookupName | 7-3 |
| Figure 7.8 | LookupListRequest for a Timestamp | 7-3 |
| Figure 7.9 | MetadataRequest Fragment | 7-3 |
| Figure 7.10 | MetadataRequest for a Vocabulary Document | 7-4 |
| Figure 7.11 | MetadataRequest for a DataDictionary Document | 7-4 |
| Figure 7.12 | MetadataRequest for a UserInformation Document | 7-4 |
| Figure 7.13 | MetadataRequest for a Custom/Local Metadata Document | 7-4 |
| Figure 7.14 | Metadata ResourceList with Object Resource | 7-5 |
| Figure 7.15 | Metadata Object Resource OutputFormat Fragment | 7-6 |
| Figure 7.16 | Sample of ObjectReferenceList Document. | 7-6 |

| | | |
|-------------|--|------|
| Figure 7.17 | ObjectResource Fragment | 7-10 |
| Figure 7.18 | ObjectResource referenceld Fragment. | 7-11 |
| Figure 7.19 | Multiple Object ObjectType Fragment. | 7-11 |
| Figure 7.20 | Object Photo Fragment | 7-12 |
| Figure 8.1 | Update Request Fragment | 8-2 |
| Figure 8.2 | Example of a Successful UpdateResponse Document with No Warnings | 8-3 |
| Figure 8.3 | Example of a Successful UpdateResponse Document with Warnings | 8-3 |
| Figure 8.4 | Example of a failed UpdateResponse Document with Fatal Faults | 8-4 |
| Figure 8.5 | Example of an unsuccessful Update transaction returning a SOAP Fault | 8-4 |
| Figure 8.6 | Update Response Fragment | 8-5 |
| Figure 8.7 | RetsFaultList Fragment | 8-6 |
| Figure 9.1 | SearchRequest complexType Schema Fragment | 9-6 |
| Figure 9.2 | Example SearchTransaction Request | 9-8 |
| Figure 9.3 | Example Search Request - Delimited | 9-8 |
| Figure 9.4 | Example Search Response - Delimited Payload with Escape Encoding | 9-8 |
| Figure 9.5 | Example Search Request - Delimited , Select Output | 9-8 |
| Figure 9.6 | Example Search Response - Delimited payload with Local Names | 9-9 |
| Figure 9.7 | Example Search Request - ObjectReferenceList | 9-9 |
| Figure 9.8 | Example Search Response - ObjectReferenceList | 9-9 |
| Figure 9.9 | Example Search Request - All Thumbnail Objects | 9-11 |
| Figure 9.10 | Example SearchTransaction Request, Count Only. | 9-12 |
| Figure 9.11 | Example Search Request - Message of the Day | 9-12 |
| Figure 10.1 | Example of a SOAP Fault | 10-1 |

List of Tables

| | | |
|------------|--|------|
| Table 3.1 | Differences Between Message-Centric and RPC Invocations | 3-3 |
| Table 1: | ManifestType Type | 6-3 |
| Table 6.1 | RETS2 Metadata Documents | 6-5 |
| Table 6.2 | The ResourceList Resource Element | 6-5 |
| Table 6.3 | The Vocabulary Elements and Attributes | 6-13 |
| Table 6.4 | The Vocabulary Field Element | 6-16 |
| Table 7: | The Vocabulary RequiredGroup element | 6-18 |
| Table 8: | The VocabularyDisplayGroup element | 6-20 |
| Table 6.1 | DataDictionary | 6-21 |
| Table 6.2 | LookupList and Map Elements | 6-24 |
| Table 6.3 | UserInformation Elements and Attributes | 6-27 |
| Table 6.4 | MessageOfDay Elements and Attributes | 6-29 |
| Table 7.1 | GetResourceList - Request Docment Elements and Attributes | 7-1 |
| Table 7.2 | GetLookupList - Request Document | 7-2 |
| Table 7.3 | GetLookupList - Request Document Elements. | 7-3 |
| Table 7.4 | Well Known Object Resource Type Names. | 7-5 |
| Table 7.5 | The ObjectType Elements and Attributes | 7-12 |
| Table 8.1 | The Update Response Elements and Attributes | 8-2 |
| Table 8.2 | The RETSFaultList Fault Elements and Attributes | 8-6 |
| Table 9.1 | Delimited OutputFormat Field Character Encoding Delimiters | 9-1 |
| Table 9.2 | The SearchRequestType Elements and Attributes. | 9-2 |
| Table 10.1 | Soap Authorization Fault Codes | 10-2 |
| Table 10.2 | Soap Search Fault Codes | 10-2 |
| Table 10.3 | Soap Update Fault Codes | 10-4 |
| Table 10.4 | Soap Metadata Fault Codes. | 10-5 |

1 Purpose

The Real Estate Transaction Standard 2 (RETS2) is a specification for communicating real estate information between two or more end-points. It defines a series of services for use by applications such as agent desktop software, Internet Data Exchange (IDX) systems, data aggregation systems, transaction (workflow) systems, broker back-office systems, mortgage systems and other third party participants in a real estate property transaction.

This specification builds upon other specifications. These specifications, referenced in this document and listed in the Bibliography, combined with this document and the companion XML schema and WSDL files, listed in Appendix A constitutes the specification for the standard.

RETS2 has been created over several years to address concerns identified in the Real Estate Transaction Standard 1 family (RETS1) of specifications that could not be addressed by changes to RETS1. As well, new technologies and business opportunities suggested new thinking on the meaning of the standard.

2 Notational Conventions

2.1 Terminology Conventions

Key terms and acronyms are defined in the Glossary. The first use of an acronym is expanded to the full form followed by the acronym. An example is Hyper-Text Transport Protocol (HTTP). Subsequent use is by acronym only.

2.2 Typographic Conventions

Examples, fragments and artifact abstracts are set in a mono-spaced font:

```
<xmlfragment attribute="quoted value">
  element data
</xmlfragment>
```

No information is implied by whitespace within examples, fragments and artifact abstracts. The figures have been re-formatted to fit the document. The normative reference is the artifact and not the abstract in this document. Please refer to the reference location for the artifact, defined in either Appendix A or in the Bibliography.

Specific bibliographic references are denoted in the document text as [\[BIBLIOGRAPHY\]](#) where the all-small-caps reference corresponds to a bibliography entry. This type of entry represents a reference specification.

Specific citation references are denoted in the document text as [\[1\]](#) where the numeric reference corresponds to a Footnotes entry. This type of entry represents a citation to included text.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#). Use of the key word in contexts other than all-caps is not interpreted as described in [\[RFC2119\]](#). For convenience, the key word definitions are reproduced in this document, but the normative definition is the reference document.

Normative statements of requirements in the Specification, those impacting compliance, as outlined in [\[COMPLIANCE\]](#) are presented in the following manner:

Rnnnn Statement text here.

The value "nnnn" is replaced by a number that is unique among the requirements in the Specification, thereby forming a unique requirement identifier.

Requirement identifiers are namespace qualified, in such a way as to be compatible with Qualified Name (QName) from [\[namespaces\]](#). If there is no explicit namespace prefix on a requirement identifier, it should be interpreted as being in the namespace identified by the compliance URI of the document section it occurs in. An example would be "R9999". If it is qualified, the prefix should be interpreted according to the namespace mappings in effect, as documented below. An example would be "RETS2:R9999".

Some requirements clarify the referenced specification, but do not place additional constraints upon implementations. For convenience, clarifications are annotated in the following manner:

Clarification

Some requirements are derived from ongoing standardization work on the referenced specification. For convenience, such forward-derived statements are annotated in the following manner: [WSDL20], where "WSDL20" is an identifier for the specification, "WSDL Version 2.0" in this example. **A Bibliographic citation is provided in the Bibliography for each referenced specification. Bibliographic citation is provided at the end of the document.** Where possible, a specific reference version is cited. In certain cases, standardization work is on-going and was not complete when this document was published, the specification that the requirement is derived from may change; this information is included only as a convenience to implementers.

Extensibility points in underlying specifications [COMPLIANCE] are presented in a similar manner:

Ennnn Extensibility Point Name - Description

The value "nnnn" is replaced by a number that is unique among the extensibility points in the Specification. As with requirement statements, extensibility statements can be considered namespace-qualified.

2.3 Rules

The following rules are used throughout this specification to describe parsing constructs. The US-ASCII coded character set is defined by ANSI X3.4-1986 [ANSI]. Parsed entities are constructed combinations of atoms or other entities as defined below. Atoms may be combined and repeated to form longer constructs. When there are constraints on the repetition of atoms, the constraints are expressed by a notation of the form described in the Augmented BNF [RFC2234], [RFC4234].

3 Web Services in the Real Estate Context - an Introduction

3.1 Context

Interchange of information between parties involved in Real Estate workflow processes require several pieces to be agreed upon for successful interchange. While workflow processes can be implemented without technology, this document is written from the perspective of automating as many pieces as possible.

Interchange parties must agree on who can request the information, what actions are required to receive the information, how the information is delivered, what the information looks like and when they can make those requests.

Using open standards can help to simplify the agreement between parties by accepting appropriate constraints embodied in the various standards. As well, open standards can allow RETS to benefit from the work in different industries that are applicable to the Real Estate information technology environment.

After initial engineering work and in consultation with the industry, the Web Services standards technology suite **were was** identified as the target standards to build RETS2 around.

3.2 Advantages of Web Services

Quoting from the W3C Web Services Activity statement:

Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable (sic) descriptions thanks to the use of XML. They can be combined in a loosely coupled way in order to achieve complex operations. Programs providing simple services can interact with each other in order to deliver sophisticated added-value services.[1]

They use open standards and protocols. Both protocols and data formats are text-based where possible, making it easy for developers to comprehend. Protocols include enveloping, security and transport and use mature standards that have been tested in industry. A language for describing structured data widely used today, Extensible Markup Language (XML), provides the descriptions for the services available, enveloping, and security. These protocols can work through many common firewall security measures without requiring changes to the firewall filtering rules. Data formats include XML formats, allowing greater standardization of data names and types.

Widespread adoption of Web Services and XML technologies has caused the creation of many tools and techniques to effectively develop Service Provider and Service Requestor applications.

3.3 Disadvantages of Web Services

Web services do not count among the design goals conciseness or wire-line efficiency in the base description. Further, the increased capabilities of Web Services compared with older technologies in the Real Estate context comes at a cost of potentially increased complexity.

While these disadvantages are real, careful design of the standard and careful implementation of the solution should address each of the disadvantages.

3.4 SOAP

SOAP, previously known as the Simple Object Access Protocol [SOAP2], provides a simple and lightweight mechanism for exchanging structured and typed information between end-points in a distributed environment using XML.

Quoting from the SOAP Primer:

SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information. SOAP is silent on the semantics of any application-specific data it conveys, as it is on issues such as the routing of SOAP messages, reliable data transfer, firewall traversal, etc. However, SOAP provides the framework by which application-specific information may be conveyed in an extensible manner. Also, SOAP provides a full description of the required actions taken by a SOAP node on receiving a SOAP message.[2]

SOAP defines a simple mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of systems ranging from messaging systems to Remote Procedure Call (RPC) systems.

Familiarity with SOAP is essential to understanding RETS2. Please refer to the bibliography for additional background material on SOAP.

SOAP messages can be described as an envelope. The envelope consists of a header and the message body. An analogy can be drawn between SOAP messages and a postal letter. The letter has an envelope (envelope) that encloses some information about the end-point and possibly the message itself in the address and physical envelope (the header) and the contents of the letter (the body).

3.5 Document Model

SOAP can use one of two message styles: Document/literal or Remote Procedure Call (RPC). The document/literal, or message-centric, model is frequently used for data-centric programming. Data is exchanged in a prescribed message format that both the sender and receiver can understand. The emphasis is on data delivery using communication protocols and programming support to send and

receive data. The strengths of the document model lie in the implementation of the messaging system, such as support for a wide variety of platforms and languages, assured delivery, persistence, support for asynchronous invocations, support for open standards, and performance. The RPC approach emphasizes actions on the data. **RPC or remote method invocation is better suited to fine-grained access to specific attributes.**

Given these two models, the RPC model was deemed inappropriate since real estate standards have been focused on exchanging data not application parameters. Document/Literal model messages are also the most interoperable.

R0001 All RETS2 Messages **MUST** use the SOAP Document/Literal Model.

Table 3.1 Differences Between Message-Centric and RPC Invocations

| MESSAGE-CENTRIC INVOCATION | RPC INVOCATION |
|--|--|
| Data centric: <ul style="list-style-type: none"> Aligns well when there is an exchange of data between applications. | Function centric: <ul style="list-style-type: none"> Aligns well when service invocation is thought of as a procedure call. |
| Data need not have an object representation. Data can be sent as a payload. | Works well for data and objects that can be serialized easily. |
| Asynchronous invocation: <ul style="list-style-type: none"> Application makes a request and doesn't wait for a response, it relies on event notification. | Synchronous invocation: <ul style="list-style-type: none"> Application makes a request and waits for a response. |
| Strong notion of message semantics like assured deliver. | |

3.6 The Resource, Vocabulary & Payload Model

The RETS2 Service utilizes a Resource, Vocabulary and Payload model. Specific Resources like Agent, Office or Property have document request actions, using specific vocabularies, resulting in specific payloads returned as the result of the action. These actions along with the vocabularies and payloads can be generally described as services.

More specifically, the RETS2 Service implements a Data Services model. It is designed to expose a common data model across a wide variety of real estate applications. This specification also allows for the creation of localized data resources, vocabularies and payload formats, customized for the unique needs of the local parties.

Resources in RETS2 have been generalized from RETS1 to assist implementation. A more general Resource model was chosen because it accelerates application development.

RETS2 allows for the creation of payloads with unique data semantics for any real estate domain. Standardized data payloads are typed and exist independently of the services that act upon it.

4 Web Services Specifications

Unless specified elsewhere in this document, the specifications referenced in the bibliography below provide the message and transport standards for RETS2.

4.1 Basic Web Services Components

The Web Services components used by RETS2 are SOAP 1.2 [SOAP1], [SOAP2], [SOAP3], WSDL 1.1 [WSDL1], XML 1.0 [XML1], [XML2], and XML Schema [XML3], [XML4], [XML5], [XML6].

R0002 Services, Messages and Payloads **MUST** conform to the specifications defined in all of SOAP 1.2, WSDL 1.1, XML 1.0 and XML Schema.

The referenced specifications cover:

- Messaging: the exchange of Web service protocol elements, usually over a network
- Description: the enumeration of the messages associated with a Web service, along with implementation details
- Discovery: metadata that enables the advertisement of a Web service's capabilities
- Security: mechanisms that provide integrity, privacy, authentication and authorization.

4.2 SOAP Message Transmission Optimization Method

RETS2 uses the Message Transmission Optimization Method (MTOM) [MTOM1], [MTOM2] to allow for the streaming of payload data. MTOM defines an abstract mechanism for optimizing the transmission and/or wire format of SOAP messages by selectively encoding portions of the message using different encoding systems. MTOM is used instead of other methods, such as SOAP with Attachments, because MTOM has interoperability support between Microsoft and other vendors which is not present in SOAP with Attachments.

R0003 Providers **MAY** use the SOAP Message Transmission Optimization Method when sending payloads.

R0004 Requestors **MUST** accept the SOAP Message Transmission Optimization Method when receiving payloads.

Several industry usage scenarios make the use of MTOM critical to the RETS2 specification. MTOM allows streaming of response documents to the Requestor. Otherwise, Requestor processing of the payload is delayed until the entire payload is delivered. This could result in performance and memory problems for the Requestor.

4.3 Web Services Security in RETS2

RETS2 uses Web Services Security (WS-SECURITY) to address transport security, messaging security, and all other security considerations. RETS2 includes the following OASIS Web Services Security profiles: SOAP Message Security [SEC1], Username Token Profile [SEC2], X.509 Certificate Token Profile [SEC3] and SAML Token Profiles [SEC4].

R0005 Messages **MUST** conform to the SOAP Message Security specification.

The SOAP Message Security Specification describes enhancements to SOAP messaging to provide message integrity and single message authentication. The specified mechanisms can be used to accommodate a wide variety of security models and encryption technologies. Additionally, this specification describes how to encode binary security tokens, a framework for XML-based tokens, and how to include opaque encrypted keys. It also includes extensibility mechanisms.

R0006 Messages **SHOULD** use Username Token Profile.

E0001 Messages **MAY** support x.509 Token Profile or SAML Token Profile or additional profiles for authentication.

Username Token Profile describes how a Requestor can supply a Username Token from the WS-SECURITY specification as a means of identifying the Requestor by “username”, and optionally using a password, password equivalent or shared secret to authenticate that identity to the Provider.

Figure 4.1 Username Token Fragment Using Digest

```
<SOAP:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wss="
    http://www.docs.oasis-open.org/wss/2004/01/
    oasis-200401-wss-wssecuritysecext-1.0.xsd">

  <SOAP:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>RETSUser</wsse:Username>
        <wsse:Password Type="wsse:PasswordDigest">
          YI3n0dwejMNVksJUFV3t7rgHh3Rww=
        </wsse:Password>
        <wsse:Nonce>
          GRcqanjDTG9mQoBE02sAJP=
        </wsse:Nonce>
      </wsse:UsernameToken>
    </wsse:Security>
  </SOAP:Header>

</SOAP:Envelope>
```

R0007 Messages that use Username Token authentication **MUST** conform to the Web Services Security: Username Token Profile Document V1.0.

R0126 Messages that use Username Token authentication **SHOULD** use Password_Digest instead of Password_PlainText.

It should be noted that Digest by itself does ensure greater security. The Username Token profile uses the SHA-1 digest algorithm to create the hash. Refer to the reference documentation [SEC2] for more detail. Within the Username Token Profile, to minimize replay attacks, the optional wsse:Nonce and wsu:Created elements are recommended. To simplify interoperability within the RETS2 Service, the recommendations are elevated to must use elements.

R0127 A Provider that uses Username Token Profile authentication and implement Password_Digest **MUST** reject any Username token that does not use both Nonce and Created elements.

The document also provides recommendations with respect to the Provider implementation to improve security.

E0004 It is **RECOMMENDED** that a Provider that uses Username Token Profile authentication and implement Password_Digest reject any token that is older than a certain age. As a guideline, an age of five minutes for a valid Created element can be used to detect and reject replay attacks.[\[SEC2\]](#)

E0005 It is **RECOMMENDED** that a Provider that uses Username Token Profile authentication and implement Password_Digest cache Nonce values for a period at least as long as the age of the Created element above and that any token that is reuses a nonce be rejected.[\[SEC2\]](#)

As documented in the specification, [\[SEC2\]](#), while the typical use is that it represents an actual password, the wsse:Password element, password equivalents, including onetime password, can be used.

X.509 Certificate Token Profile describes how to use the x.509 authentication framework with the SOAP Message Security specification. An X.509 certificate specifies a binding between a public key and a set of attributes that must include a subject name, issuer name, serial number and validity interval. An X.509 certificate may be used to validate a public key that may be used to authenticate a WS-SECURITY enhanced message or to identify the public key with which a WS-SECURITY enhanced message has been encrypted.

E0002 Messages that use x.509 Token authentication **MUST** conform to the Web Services Security: x.509 Certificate Token Profile Document 200401.

SAML Token Profile describes how to use the SAML framework with the SOAP Message Security specification.

E0003 Messages that use XML Token authentication **MUST** conform to the Web Services Security: SAML Token Profile Document 200412.

5 Description and Discovery

5.1 Service Registry

Web Services defines an optional public registry to provide a general purpose service registration and description technique. This registry, the Universal Description, Discovery and Integration (UDDI) can allow external parties to dynamically discover the resources and services that a particular Provider offers. It was determined that UDDI will not be used as part of the RETS2 standard at this time. Such a public registry was not required for RETS2 because the services defined in the standard are published as part of the documents that make up RETS2 and there are typically contractual agreements in place between Providers and Requestors. Service Requestors who would use RETS must execute legal agreements and apply for access through the Multiple Listing Service. As part of the agreement between the parties, specific discovery information is provided, making dynamic discovery of services unnecessary.

5.2 Web Services Description Language (WSDL)

5.2.1 RETS2 Service WSDL

Web Services provides a description language to define data types, services and bindings that a specific instance supports. Instances of the description language will be referred to in the RETS2 documents as WSDL or RETS2 WSDL. The instances are contained in files that are used by various Web Service components to construct and validate the known services. These files will be referred to in the rets2 documents as wsdl files, since the files have the file extension of 'wsdl'.

- R0010 The RETS2 WSDL is defined in the document rets.wsdl.
- R0011 A Provider **MUST** support the SearchRequest Action.
- R0012 A Provider **MUST** support the GetResourceList Action.
- R0013 A Provider **MUST** support the GetLookupList Action.
- R0014 A Provider **MAY** support the UpdateRequest Action.
- R0015 A Provider **MAY** support the GetMetadataRequest Action.

For convenience, the RETS2 WSDL document is provided in Appendix C. The normative version is the wsdl file and not Appendix C.

5.2.2 RETS2 Service Extensions WSDL

To satisfy local needs, Providers may supply services not covered in the RETS2 specification documents. Such [service](#) extensions are supplied through separate wsdl files.

- R0016 A Provider who extends the services of RETS2 **MUST** use one or more separate wsdl files to describe those changes.
- R0145 Any extension wsdl file **MUST** conform to the Web Services Description Language 1.1 Specification.

6 Service Vocabularies and Responses

6.1 Soap Headers

6.1.1 Request Headers

At this time there are no SOAP Request Headers specific to RETS2.

6.1.2 Response Header

At this time there are no SOAP Response Headers specific to RETS2.

6.2 XML Formats and Versioning

6.2.1 XML Based

RETS2 requires all XML responses to be well-formed XML. In addition, this specification requires that Requestors ~~clients~~ parse RETS responses as XML, not as simple text streams. That is, servers are free to produce response XML in any format that complies with the W3C XML 1.0 recommendation. XML escaping of content is implied, as is XML processing of line endings and whitespace [xml1].

6.2.2 XML Request Formats

R0017 A Request Format **MUST** conform to the XML 1.0 Specification and the XML Schema Structures and Datatypes specifications.

6.2.3 XML Response Formats

R0018 An XML Response Document **MUST** conform to the XML 1.0 Specification and the XML Schema Structures and Datatypes specifications.

6.2.4 XML Response Document Versioning

It is expected that Response Documents will change over time. To ensure that Requestors can understand when such changes occur, Response Documents are versioned. This includes the schema defined by this standard as well as any custom schema implemented by Providers.

E0006 A Provider **MUST** have a versioned target namespace URI and a versionTimestamp for an XML OutputFormat schema.

Some changes to the document format will not impact systems built using previous document formats. In such cases, the new version of the documents is backward compatible with previous schema. For this standard, backward compatibility is defined as those documents, created by a specific version of the document schema, that are valid documents in earlier document schema versions.

As a guideline, where possible, all changes should be extensions to the schema. Following this guideline will assist in making newer schema documents backward compatible.

E0007 A Provider **SHOULD** use extensions to create new versions of existing document schema.

Examples of changes that make new documents incompatible with previous schema are:

- Changing the meaning or semantics of existing components
- Adding required elements or attributes
- Removing required elements or attributes
- Adding or removing a restriction on a component content model – changing a choice to a sequence is an example ~~one case~~

R0019 A Provider **MUST** change both the URI and the timestamp for an XML OutputFormat schema when a new version is not backward compatible.

Examples of changes that should make new documents compatible with previous schema are:

- Adding optional elements, attributes or a combination of elements and attributes
- Adding optional content to a component content model – adding an enumeration is one case

R0020 A Provider **MUST** change only the timestamp for an XML OutputFormat schema when a new version is backward compatible.

6.2.5 Unknown Elements and Attributes

Any output format or request that is an XML format should be a valid XML document by the corresponding XML schema. Unknown elements and attributes should be found in place of the <xs:any> elements in the schema. Unknown elements and attributes should be ignored by both Providers and Requestors.

R0021 An XML document that contains elements or attributes in place of the any or other schema elements and attributes **SHOULD** be processed by ignoring the unknown element or attribute.

R0022 A Provider **MAY** issue a fault when a document or request does not validate against the schema for that document.

6.3 RETS Manifest

RETS2 payloads may be XML documents, binary data such as .jpg or .gif images, or other MIME types supported by a Provider.

The payloads are packaged inside of the RETS Manifest. To preserve the logical structure of the document, RETS uses a manifest document to describe the structure, allowing the contents to be serialized as necessary.

6.3.1 ManifestType Type Schema

The ManifestType is made up of one or more *Reference* elements. Each *Reference* has three elements: *Description*, *Count* and *Content*. ~~Each Reference also has an optional *taskId* that uniquely identifies the task that relates to this Manifest.~~

Table 1: ManifestType Type

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|-------------------------------|--|
| ManifestType/Description | A human-readable text description |
| ManifestType/Description@lang | A token defining the language of the payload |
| ManifestType/Count | An optional element containing the number of records returned in the Payload. The element should be included in the response if the corresponding <i>SearchRequest/Count</i> element is set to true. The default value is false. The UpdateRequest Action does not use the <i>Count</i> element. |
| ManifestType/Content | The base64Binary object, either an XML document or RETS ObjectType (photo, thumbnail, etc.) |
| ManifestType/Content@key | an optional attribute that serves as the unique identifier for an Object, for example, photo |
| ManifestType/Content@type | an optional attribute to describe the <i>Content</i> MIME type, for example, text/xml |
| ManifestType@taskId | an optional attribute that uniquely identifies the task that relates to this Manifest |

~~The *Description* element contains a human-readable text description, with attributes of lang, to define the language and type, to define the type of content.~~

~~The *Count* element is optional. It may be used in a SearchResponse. If the corresponding *SearchRequest/Count* element is set to “true”, the Manifest *Count* element must contain the number of records returned in the Search payload. The UpdateRequest does not use the *Count* element.~~

~~The *Content* element contains the base64Binary object that is either an XML document or RETS ObjectType (photo, thumbnail, etc.). It has an optional *key* attribute that serves as the unique identifier for an Object, for example, photo and an optional *type* attribute to describe the *Content* MIME type, for example, text/xml.~~

Figure 6.1 ManifestType Fragment

```
<xsd:complexType name="ManifestType">
  <xsd:sequence>
    <xsd:element name="Reference" maxOccurs="unbounded">
      <xsd:complexType>
```

```

<xsd:sequence>
  <xsd:element name="Description">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:attribute name="lang" type="xsd:NMTOKEN"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Count"
    type="xsd:nonNegativeInteger" minOccurs="0"/>
  <xsd:element name="Content">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:base64Binary">
          <xsd:attribute name="key" type="xsd:string"/>
          <xsd:attribute name="type" type="xsd:string"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence>
  <xsd:attribute name="taskID" type="xsd:ID"/>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>

```

R0146 A Provider **MUST** return the number of records for the Search payload in the Manifest Count element when the SearchRequest Count is set to "true".

6.4 Resource Discovery - Metadata

Metadata documents are available to provide a Requestor with information that may be used to construct syntactically correct Search and Update actions, and process the responses. Several standard metadata documents are defined by RETS2. Those documents are ResourceList, LookupList, Vocabulary, and DataDictionary. Additional metadata documents may be added to the RETS standard Resources and Payloads in the future, as recommended by the RETS Metadata work group and adopted by the RETS Work Group

Table 6.1 RETS2 Metadata Documents

| DOCUMENT | DESCRIPTION |
|----------------|--|
| ResourceList | The ResourceList document provides a Requestor with basic information about all of the Resources supported by a RETS2 Provider. |
| Vocabulary | Vocabulary Metadata supplies all of the necessary information for a RETS Requestor to construct a Query for a RETS Search action. |
| LookupList | The LookupList contains Maps that can be used across Resources to define Lookup types. |
| DataDictionary | The DataDictionary Metadata document contains detailed information for all fields (searchable and non-searchable) within a Resource. |

6.4.1 ResourceList

The ResourceList document provides a Requestor with basic information about all of the Resources supported by a RETS2 Provider. As shown in the sample document, each Resource element in the ResourceList contains the following information:

Table 6.2 The ResourceList Resource Element

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|--------------------------|---|
| Resource/ResourceName | The name of the Resource and the key field to be used in performing Search, Update or Metadata actions against this Resource. |
| Resource@version | A URI that changes when the Resource is no longer backward compatible with the previous version |
| Resource@timestamp | A timestamp in ISO8601 format that changes whenever the Resource is modified. |
| Resource/InformationURL | A URL that links to a web page with human-readable content about the Resource. |
| Resource/VocabularyNames | A list of the vocabularies that may be used in a Search action to query the Resource. |
| Resource/OutputFormats | A list of the types of Search action responses, also known as payloads, supported for the Resource. |

Table 6.2 The ResourceList Resource Element

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|--------------------------|--|
| Resource/InputFormats | A list of the types of Update action request documents and objects supported for the Resource. |
| Resource/MetadataFormats | A list of the types of Metadata documents supported for the Resource. |

Figure 6.2 Sample ResourceList Metadata Document

```

<retsrsl:ResourceList
  versionTimestamp="2006-04-07T00:00:00Z"
  xmlns:retsrsl="http://www.rets.org/ns/ResourceList/200604"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.rets.org/ns/ResourceList/200604">
  <Resource timestamp="2006-03-01T09:30:47.0Z"
    version="http://www.example.com/02/01">
    <ResourceName>Residential</ResourceName>
    <InformationURL>
      http://www.example.com/info/Residential.html
    </InformationURL>
    <VocabularyNames>
      <VocabularyName>WellKnown</VocabularyName>
    </VocabularyNames>
    <OutputFormats>
      <OutputFormat>
        <FormatURL>
          http://www.rets.org/schema/ListingProperty.xsd
        </FormatURL>
        <Description>
          RETS standard ListingProperty payload
        </Description>
        <Displays>
          <Display name="Printable">
            http://www.example.com/listprop/PrettyPrint.xml
          </Display>
          <Display name="DisplayAsList">
            http://www.example.com/listprop/ItemList.xml
          </Display>
        </Displays>
      </OutputFormat>
      <OutputFormat>
        <FormatURL>
          http://www.rets.org/schema/Listing.xsd
        </FormatURL>
        <Description>
          RETS standard Listing payload
        </Description>
        <Displays>

```

```

    <Display name="Printable">
      http://www.example.com/listing/PrettyPrint.xml
    </Display>
    <Display name="DisplayAsList">
      http://www.example.com/listing/ItemList.xml
    </Display>
  </Displays>
</OutputFormat>
<OutputFormat>
  <FormatURL>
    http://www.rets.org/schema/ObjectReferenceList.xsd
  </FormatURL>
  <Description>
    RETS standard Object Reference List to provide details
    on all objects associated with this resource
  </Description>
</OutputFormat>
<OutputFormat>
  <ObjectType>DELIMITED</ObjectType>
  <Description>
    RETS DELIMITED format
  </Description>
</OutputFormat>
</OutputFormats>
<InputFormats>
  <InputFormat>
    <FormatURL>
      http://www.example.com/schema/ListingEntry.xsd
    </FormatURL>
    <Description>
      MLS Listing Entry Form
    </Description>
  </InputFormat>
</InputFormats>
<MetadataFormats>
  <MetadataFormat>
    <FormatURL>
      http://www.rets.org/schema/Vocabulary.xsd
    </FormatURL>
    <Description>
      RETS standard Vocabulary schema
    </Description>
  </MetadataFormat>
  <MetadataFormat>
    <FormatURL>
      http://www.rets.org/schema/DataDictionary.xsd
    </FormatURL>
    <Description>
      RETS standard DataDictionary schema
    </Description>
  </MetadataFormat>
</MetadataFormats>

```

```

    <FormatURL>
      http://www.example.com/schema/CustomMetadata.xsd
    </FormatURL>
    <Description>
      Custom User Interface Metadata definition
    </Description>
  </MetadataFormat>
</MetadataFormats>
</Resource>
</retsrl:ResourceList>

```

The ResourceName is the key field to be used in performing Search, Update or Metadata actions against this Resource. If the Resource corresponds to one of the RETS well-known Resources, the **Provider Endpoint** must use the well-known name as the ResourceName.

R0023 Where a Resource name corresponds to a RETS well-known name, the **Provider Endpoint** **MUST** use the well-known name. See the “RETS 2 Well-Known Names” document for the list.

Figure 6.3 ResourceList ResourceName Fragment

```

<retsmeta:ResourceName>Residential</retsmeta:ResourceName>

```

The version and timestamp attributes provide information regarding changes to the Resource. The version is a URI. It changes when the version of the Resource is not backward compatible to a previous version. The timestamp changes whenever the Resource has been modified.

Figure 6.4 ResourceList Resource Fragment

```

<retsmeta:Resource
  versionTimestamp="2006-01-02T09:40:04-05:00">
  ...
</retsmeta:Resource>

```

The InformationURL is an optional element that links to an html page where the Provider has described this Resource and its actions. This link may provide useful, human-readable content to a Requestor.

Figure 6.5 ResourceList InformationURL Fragment

```

<InformationURL>
  http://www.myserver.com/info/Residential.html
</InformationURL>

```

The VocabularyNames **element** lists the vocabularies that may be used in a Search action to query the Resource. The VocabularyName is a key field that can be used as a reference to retrieve the Vocabulary metadata document for a given Resource. There are two well-known Vocabulary names; WellKnown and Local.

The WellKnown Vocabulary provides the names listed in the “RETS Well-Known Names” document. The WellKnown vocabulary replaces the RETS1 StandardNames query concept.

The Local Vocabulary represent human-readable names specific to a Provider instance. The Local vocabulary replaces the RETS1 SystemNames query concept.

In addition to the Vocabulary names defined in this document, a Provider may define and support additional vocabularies. Those vocabulary names are added to the ResourceList VocabularyNames element node.

Figure 6.6 ResourceList VocabularyName Fragment

```
<retsmeta:VocabularyNames>
  <VocabularyName>Local</VocabularyName>
  <VocabularyName>WellKnown</VocabularyName>
</retsmeta:VocabularyNames>
```

OutputFormats list the types of Search responses supported for that Resource. An OutputFormat may be either an OutputObject or FormatURL. OutputObjects are predefined such as a Photo or the Delimited response type. A FormatURL will point to either a RETS standard schema, or a local/custom schema/dtd.

Figure 6.7 ResourceList OutputFormats Fragment

```
<OutputFormats>
  <OutputFormat>
    <FormatURL>
      http://www.rets.org/schema/ListingProperty.xsd
    </FormatURL>
    <Description>
      RETS standard ListingProperty payload
    </Description>
    <Displays>
      <Display name="Printable">
        http://www.myserver.com/listprop/PrettyPrint.xml
      </Display>
      <Display name="DisplayAsList">
        http://www.myserver.com/listprop/ItemList.xml
      </Display>
    </Displays>
  </OutputFormat>
  <OutputFormat>
    <FormatURL>
      http://www.rets.org/schema/Listing.xsd
    </FormatURL>
    <Description>
      RETS standard Listing payload
    </Description>
    <Displays>
      <Display name="Printable">
        http://www.myserver.com/listprop/PrettyPrint.xml
      </Display>
      <Display name="DisplayAsList">
        http://www.myserver.com/listprop/ItemList.xml
      </Display>
    </Displays>
  </OutputFormat>
</OutputFormats>
```

```

        </Display>
</Displays>
</OutputFormat>
<OutputFormat>
  <FormatURL>
    http://www.rets.org/schema/ObjectReferenceList.xsd
  </FormatURL>
  <Description>
    RETS standard Object Reference List to provide
    details on all objects associated with this resource
  </Description>
</OutputFormat>
<OutputFormat>
  <ObjectType>DELIMITED</ObjectType>
  <Description>
    RETS DELIMITED format
  </Description>
</OutputFormat>
</OutputFormats>

```

An OutputFormat may contain a Displays element. The Displays element contains one or more Display elements. A Display element provides a URL to an XSL transform created by a Provider. The transforms may be used by a Requestor to format the XML payload into specific user interface views such as printing or list displays. If display elements are present, a Requestor is not required to use the transform to display the data of the payload.

Figure 6.8 ResourceList OutputFormat Fragment

```

<OutputFormat>
  <FormatURL>
    http://www.rets.org/schema/Listing.xsd
  </FormatURL>
  <Description>
    RETS standard Listing payload
  </Description>
  <Displays>
    <Display name="Printable">
      http://www.myserver.com/listprop/PrettyPrint.xsl
    </Display>
    <Display name="DisplayAsList">
      http://www.myserver.com/listprop/ItemList.xsl
    </Display>
  </Displays>
</OutputFormat>

```

InputFormats list the types of Update request documents supported for that Resource. Like an OutputFormat, an InputFormat may be either An InputObject or a FormatURL. InputObjects are predefined types such as Photos. A FormatURL will point to either a RETS standard schema, or a local schema or a custom schema.

Figure 6.9 ResourceList InputFormats Fragment

```
<InputFormats>
  <InputFormat>
    <FormatURL>
      http://www.myserver.com/schema/ListingEntry.xsd
    </FormatURL>
    <Description>
      MLS Listing Entry Form
    </Description>
  </InputFormat>
  <InputFormat>
    <ObjectType>
      Photo
    </ObjectType>
    <Description>
      Jpg and Gif photos
    </Description>
  </InputFormat>
</InputFormats>
```

MetadataFormats list the types of Metadata documents supported for that Resource. A MetadataFormat will contain a URL pointing to either RETS Metadata document schema (Vocabulary, DataDictionary, or LookupList), or a local metadata schema or a custom metadata schema.

Figure 6.10 ResourceList MetadataFormats Fragment

```
<MetadataFormats>
  <MetadataFormat>
    <FormatURL>
      http://www.rets.org/schema/Vocabulary.xsd
    </FormatURL>
    <Description>
      RETS standard Vocabulary schema
    </Description>
  </MetadataFormat>
  <MetadataFormat>
    <FormatURL>
      http://www.rets.org/schema/DataDictionary.xsd
    </FormatURL>
    <Description>
      RETS standard DataDictionary schema
    </Description>
  </MetadataFormat>
  <MetadataFormat>
    <FormatURL>
      http://www.myserver.com/schema/CustomMetadata.xsd
    </FormatURL>
    <Description>
      Custom User Interface Metadata definition
    </Description>
  </MetadataFormat>
```

</MetadataFormats>

- R0024 A Provider **MUST** provide ResourceList Metadata.
- R0025 A Provider **MUST MAY** provide at least one ~~or more~~ Vocabulary for each Resource.
- R0026 A Resource **MUST** provide the WellKnown Vocabulary for that Resource.
- R0027 Wherever a well-known Resource can map a field to a WellKnown name, as defined in the Well-Known names document, it **MUST** provide a WellKnown Vocabulary containing that field using the WellKnown name.
- R0028 For any Resource, a Provider **MAY** provide only the WellKnown **Vocabulary names**.
- R0029 A Provider that operates as WellKnown names only Provider **MUST** provide a ResourceList Metadata document.
- R0030 A Provider that operates as a WellKnown names Provider is not required to provide any other standard Metadata documents including Vocabulary, DataDictionary or Lookup.
- R0031 A Search Action Resource **MUST** provide at least one OutputFormat.
- R0032 A Search Action Resource **MAY** provide multiple OutputFormats.
- R0033 A Provider **MAY** provide Display elements for an OutputFormat.
- R0034 A Requestor **MAY** use a Display element to format payloads for presentation.
- R0035 An Update Action Resource **MUST** provide at least one InputFormat.
- R0036 An Update Action Resource **MAY** provide multiple InputFormats.
- R0037 A ResourceList document **MUST** validate against the ResourceList.xsd schema.

6.4.2 Vocabulary

A Vocabulary in the context of RETS2 is a collection of fields for a specific Resource that can be used to form the criteria to request one or more documents that match the criteria of the Requestor. In addition, the Vocabulary provides the collection of fields that can be used to select the fields of the result set when the OutputFormat is Delimited [or Encoded-Delimited](#).

[The Vocabulary Metadata document is a structured document that realizes the Vocabulary in a specific system. Vocabulary and Vocabulary Metadata document are used interchangeably within this document. In certain cases, this term will be the abstract notion of the collection of fields while other](#)

cases will refer to the concrete document. Where this difference is important, the usage should be made clear by the context. In most cases, the reference is to the concrete document.

The WellKnown Vocabulary is a Vocabulary for a specific Resource that contains the WellKnown Names for that Resource available in the particular Provider system.

Some Providers may choose to implement a person-based or roles-base security limitation that causes the Vocabulary for a Resource to change based on a person or the role they are in. Some users, or class of users, may not be permitted to query on specific fields based on local business rules.

As a guideline, Requestors should be prepared for faults when using any Vocabulary in cases where a specific user is not permitted to form queries using one or more of the fields. Additionally, Requestors should be prepared for Vocabularies to change based on an individual.

Any Query that uses one or more Field Names that are not valid for a given user or system should result in a fault. Some Reasons for an invalid unknown field name fault may include using an unknown field that is not implemented unimplemented-name in a specific system, system security limitations, a mismatch between a Vocabulary and a Resource or other implementation specific causes.

R0148 A Provider **SHOULD** return a fault when an invalid Field Name is used in a Query by the Requestor.

The Vocabulary Metadata supplies all of the necessary information for a RETS Requestor to construct a Query for a Search action. Vocabularies are specific to a given Resource. One Resource may support multiple Vocabularies. The elements and attributes that compose a Vocabulary document are defined in the following table.

Table 6.3 The Vocabulary Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|------------------------------|--|
| Vocabulary@name | The name of the Vocabulary. A Requestor MUST use this name for the Vocabulary when referencing it in a Search action. RETS2 has two well-known vocabularies: WellKnown and Local. A Provider may support additional vocabularies. |
| Vocabulary@keyfield | The Field name in the Vocabulary that is the key field for this Resource. |
| Vocabulary@versionTimes-tamp | A timestamp, in ISO8601 format that changes when the Vocabulary schema has changed. |
| Vocabulary/Fields | A collection list of the searchable fields a Requestor MAY use in building a Query using this Vocabulary. |
| Vocabulary/RequiredGroups | An optional collection of lists of groups of field names that a Requestor MUST use when constructing a Query. Field names may appear in more than one list. List items are separated by whitespace. |

Table 6.3 The Vocabulary Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|--------------------------|--|
| Vocabulary/DisplayGroups | An optional collection of lists of field names that have some logical grouping. A Requestor MAY use the grouping to build query criteria input user screens. Field names may appear in more than one list. List items are separated by whitespace. |

Figure 6.11 Sample Vocabulary Metadata

```

<retsvoc:Vocabulary
  xmlns:retsvoc="http://www.rets.org/ns/Vocabulary/200604"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.rets.org/ns/Vocabulary/200604/Vocabulary.xsd"
  name="Local"
  keyField="ml_num"
  versionTimestamp="2006-04-07T00:00:00Z">
<Fields
  version="uri://tempuri.org/residential/vocabulary/local/200601"
  timestamp="2006-03-31T00:00:00Z">
  <Field searchable="true" selectable="true">
    <MetaEntryId>ml_num</MetaEntryId>
    <Name>MLSNumber</Name>
    <DataType>Character</DataType>
  </Field>
  <Field searchable="true" selectable="true">
    <MetaEntryId>status</MetaEntryId>
    <Name>ListingStatus</Name>
    <DataType>Character</DataType>
  </Field>
  <Field searchable="true" selectable="true">
    <MetaEntryId>price_list</MetaEntryId>
    <Name>ListingPrice</Name>
    <DataType>Integer</DataType>
  </Field>
  <Field searchable="true" selectable="true">
    <MetaEntryId>date_listed</MetaEntryId>
    <Name>ListingDate</Name>
    <DataType>Date</DataType>
  </Field>
  <Field searchable="true" selectable="true">
    <MetaEntryId>city1</MetaEntryId>
    <Name>City</Name>
    <DataType>Lookup</DataType>
    <LookupName>CityList</LookupName>
  </Field>
</Fields>
<RequiredGroups
  version="uri://tempuri.org/residential/vocabulary/local/reqd/200601"
  timestamp="2006-03-15T00:00:00Z">

```

```

<RequiredGroup>
  <Id>grp1</Id>
  <Name>cma</Name>
  <FieldList>
    ListingPrice ListingDate Area
  </FieldList>
</RequiredGroup>
<RequiredGroup>
  <Id>grp2</Id>
  <Name>listing_detail</Name>
  <FieldList>
    MLSNumber
  </FieldList>
</RequiredGroup>
<RequiredGroup>
  <Id>grp3</Id>
  <Name>listing_status</Name>
  <FieldList>
    ListingStatus ListingDate
  </FieldList>
</RequiredGroup>
</RequiredGroups>
<DisplayGroups
  version="uri://tempuri.org/residential/vocabulary/local/display/200601"
  timestamp="2006-03-01T00:00:00Z">
  <DisplayGroup>
    <Id>dgrp1</Id>
    <Name>Address</Name>
    <FieldList>
      StrNum StrName StrSuffix City State Zip
    </FieldList>
  </DisplayGroup>
  <DisplayGroup>
    <Id>dgrp2</Id>
    <Name>Listing_Info</Name>
    <FieldList>
      ListingStatus ListingPrice
    </FieldList>
  </DisplayGroup>
</DisplayGroups>
</retsvoc:Vocabulary>

```

The name attribute provides the identifier for this Vocabulary when a Requestor references it in a Search Action. The Vocabulary name may be used for the Vocabulary parameter in the Search Action, or as the Vocabulary prefix in a Search Action Query parameter.

A Provider may reuse the same Vocabulary name across Resources. Since the Vocabulary is associated with a Resource, the fields within that Vocabulary can vary by Resource. A Vocabulary document named WellKnown for Residential would have a different Fields content than the WellKnown Vocabulary document for LotsAndLand.

The keyField attribute identifies which Field in the Vocabulary acts as a Key Field for this Resource. A keyField is often used in a Query to retrieve an Object such as Photo.

The versionTimestamp attribute provide information regarding changes to the Vocabulary Schema. The versionTimestamp will change whenever the Vocabulary Schema has been modified.

Figure 6.12 Vocabulary Timestamp Change Fragment

```
<retsvoc:Vocabulary
  xmlns:retsvoc="http://www.rets.org/ns/Vocabulary/200604"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.rets.org/ns/Vocabulary/200604/Vocabulary.xsd"
  name="Local"
  keyfield="ml_number"
  versionTimestamp="2006-04-07T00:00:00Z">
  ...
</retsvoc:Vocabulary>
```

The Fields element contains Fields that may be used to construct a query. Based on roles or other business rules, a Field may be selectable and not searchable or searchable and not selectable. While providers SHOULD remove Fields from vocabularies that are neither searchable nor selectable, Requestors SHOULD evaluate Fields before using them.

Table 6.4 The Vocabulary Field Element

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|----------------------|--|
| Field/MetaEntryId | A persistent identifier that relates a Vocabulary field to a DataDictionary Field. |
| Field/Name | The name of the Field exactly as it SHOULD be used in the a Query. Although Name is a string, it cannot contain whitespace, otherwise it will fail to be correctly identified in DisplayGroups and RequiredGroups. |
| Field/DataType | An enumerated type that specifies the nature of the contents of the Field. |
| Field/LookupName | An identifier referencing a Map in the LookupList document for Fields with a DataType of "Lookup" or LookupMulti". For Fields with a DataType of Lookup or LookupMulti, an identifier referencing a Map in the LookupList document. |
| Field@isSearchable | An optional Boolean attribute that indicates that this Field MAY be used to construct the criteria. When missing, the default behavior is that the Field may be searched. This attribute defaults to true |

Table 6.4 The Vocabulary Field Element

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|----------------------|--|
| Field@isSelectable | An optional Boolean attribute that indicates that this Field may be used to construct the result payload when the Output-Type is Delimited or Encoded-Delimited. When missing, the default behavior is that the Field may be selected. This attribute defaults to true |

The MetaEntryId field is a persistent identifier that relates a Vocabulary field to a DataDictionary field as described in the DataDictionary document. The Name element of the Field is the name of the field exactly as it should be used in the a Query. While the Name element is currently defined as string in the reference Vocabulary.xsd document, implementation constraints and token list use force all names to be constructed without whitespace. The definition may be changed in a future version of this standard to enforce this constraint.

The DataType is an enumerated type that specifies the nature of the field. The enumeration values are:

- Base64Binary
- Boolean
- Character
- Currency
- Date
- DateTime
- Decimal
- HexBinary
- Integer
- Lookup
- LookupMulti
- UnsignedInteger

Figure 6.13 Figure 19 - Vocabulary Field Fragment

```
<Field isSearchable="true" isSelectable="true">
  <MetaEntryId>ml_num</MetaEntryId>
  <Name>MLSNumber</Name>
  <DataType>Character</DataType>
</Field>
```

If a Field has a DataType of Lookup, or LookupMulti, that Field must contain a LookupName. A LookupName is an id reference to a Map in the Provider supplied LookupList Metadata. A Requestor can use that LookupName in a GetMetadata Action to retrieve the map of keys and values.

Figure 6.14 Vocabulary Field LookupName Fragment

```
<Field isSearchable="true" isSelectable="true">
  <MetaEntryId>city1</MetaEntryId>
  <Name>City</Name>
```

```

    <DataType>Lookup</DataType>
    <LookupName>CityList</LookupName>
  </Field>

```

RequiredGroups provide information about groups of fields that are required in a Query. The RequiredGroup is an evolution of the required attribute of a field from RETS1.

In the Vocabulary RequiredGroups Fragment example, the RETS Provider lists three RequiredGroups. A RETS Requestor would be required to use one of these three RequiredGroups when performing a Query. If the Requestor chose the “listing_status” RequiredGroup, both ListingStatus and ListingDate must be included in the resultant Query.

Figure 6.15 Vocabulary RequiredGroups Fragment

```

<RequiredGroups version="uri://tempuri.org/residential/vocabulary/local/
reqd/200604"
  timestamp="2006-03-15T00:00:00Z">
  <RequiredGroup>
    <Id>grp1</Id>
    <Name>cma</Name>
    <FieldList>
      ListingPrice ListingDate Area
    </FieldList>
  </RequiredGroup>
  <RequiredGroup>
    <Id>grp2</Id>
    <Name>listing_detail</Name>
    <FieldList>
      MLSNumber
    </FieldList>
  </RequiredGroup>
  <RequiredGroup>
    <Id>grp3</Id>
    <Name>listing_status</Name>
    <FieldList>
      ListingStatus ListingDate
    </FieldList>
  </RequiredGroup>
</RequiredGroups>

```

The purpose of RequiredGroups is to inform the RETS Requestor that it must use **at least** one of the RequiredGroups when constructing a Query. All of the fields within **the selected a specific** RequiredGroup must be present in the Query.

Table 7: The Vocabulary RequiredGroup element

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|----------------------|-----------------------------------|
| RequiredGroup/Id | A unique identifier for the group |

Table 7: The Vocabulary RequiredGroup element

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|-------------------------|---|
| RequiredGroup/Name | A string that may be a descriptive name or internal identifier. |
| RequiredGroup/FieldList | A string list of a subset of this Vocabulary/Fields/Field/Name values, separated by whitespace. |

~~A RequiredGroup contains a name attribute which the Provider may populate with a descriptive name or internal identifier. The RequiredGroup element contains a token list of Field Names separated by white space. Although Field Names are strings, a Field Name cannot contain whitespace. A Field Name Vocabulary/Fields/Field/Name value may exist in multiple RequiredGroups.~~

Figure 6.16 Vocabulary RequiredGroup Fragment

```
<RequiredGroup>
  <Id>grp1</Id>
  <Name>cma</Name>
  <FieldList>
    ListingPrice ListingDate Area
  </FieldList>
</RequiredGroup>
```

DisplayGroups provide information about grouping Vocabulary Fields. This information may be used to assist in creating a user interface based on this logical grouping of fields that are used to construct a Query. DisplayGroups have version and timestamp attributes that provide information about the metadata. The version is a URI that changes when the DisplayGroup metadata is no longer backward compatible with the previous version. The timestamp is a timestamp, in ISO8601 format that changes whenever the DisplayGroups are modified.

Figure 6.17 Vocabulary DisplayGroups Fragment

```
<DisplayGroups
  version="/residential/vocabulary/local/display/200601"
  timestamp="2006-03-01T00:00:00Z">
  <DisplayGroup>
    <Id>grp3</Id>
    <Name>Address</Name>
    <FieldList>
      StrNum StrName StrSuffix City State Zip
    </FieldList>
  </DisplayGroup>
  <DisplayGroup>
    <Id>grp7</Id>
    <Name>Listing_Info</Name>
    <FieldList>
      ListingStatus ListingPrice
    </FieldList>
  </DisplayGroup>
```

</DisplayGroups>

Table 8: The VocabularyDisplayGroup element

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|------------------------|---|
| DisplayGroup/Id | A unique identifier for the group |
| DisplayGroup/Name | A string that may be a descriptive name or internal identifier. The name may be used in the presentation of the group, so a Provider that chooses to offer DisplayGroups should provide a human-readable string rather than an internal identifier. |
| DisplayGroup/FieldList | A string list of a subset of this Vocabulary/Fields/Field/Name values, separated by whitespace. |

~~The name may be used in the presentation of the group, so a Provider that chooses to offer DisplayGroups should name the DisplayGroup with a human-readable name. The DisplayGroup element contains a list of Vocabulary Field Name names separated by whitespace. Although a Field Name is a string, a Field Name cannot contain whitespace. A FieldName Vocabulary/Fields/Field/Name value may exist in multiple DisplayGroups.~~

The order of the FieldList values ~~Field Name tokens~~ is significant. That is, the first value token in the list is intended to be the first field displayed in any user interface implementation that uses the DisplayGroup.

Figure 6.18 Vocabulary DisplayGroup Fragment

```
<DisplayGroup>
  <Id>grp3</Id>
  <Name>Address</Name>
  <FieldList>
    StrNum StrName StrSuffix City State Zip
  </FieldList>
</DisplayGroup>
```

While DisplayGroups are intended for presentation to human users, this does not preclude the use in machine to machine systems.

- R0038 A Vocabulary metadata document **MUST** validate against the RETS Vocabulary.xsd schema.
- R0147 A Vocabulary Field Name element **MUST NOT** contain whitespace.
- R0041 If a Vocabulary Field has a Lookup or LookupMulti DataType, that Field **MUST** contain a LookupName element.
- R0042 If a Vocabulary Field contains a MetaEntryId element, it **MUST** reference a valid MetaEntryId for a DataDictionary Field.

- R0039 A Provider **MAY** supply RequiredGroups for a Vocabulary.
- R0044 If RequiredGroups are presented in the Vocabulary document, a Requestor **MUST** use one of the RequiredGroups in a query.
- R0040 If RequiredGroups are presented in the Vocabulary document, a Requestor **MUST** use all of the Fields within one RequiredGroup in a Query.
- R0137 A Provider **MAY** supply DisplayGroups for a Vocabulary.
- R0138 If DisplayGroups are present in the Vocabulary document, a Requestor **MAY** use one or more DisplayGroup to organize the display of Field names.
- R0139 If DisplayGroups are present in the Vocabulary document, a Provider **SHOULD** provide a DisplayGroup name that may be presented to a human user.
- R0140 A Requestor **MAY** use the DisplayGroup name to present the group to a human user.
- R0141 A Provider **MUST NOT** provide any names in a DisplayGroup that are not present in the Vocabulary.
- R0142 A Provider **SHOULD** provide the DisplayGroup names in presentation order.
- R0143 A Requestor **MAY** display the DisplayGroup names in the order that they are listed in the DisplayGroup to a human user.

6.4.3 DataDictionary

The DataDictionary Metadata document contains detailed information for all fields (searchable and non-searchable) within a Resource. The DataDictionary maps most closely to the RETS1 Metadata Table. Unlike the RETS1 Metadata Table, there are no restrictions on the length of the contents of elements in a DataDictionary document (such as LongName and ShortName).

A Resource will map to one DataDictionary. Through the MetaEntryId, a DataDictionary may be related to multiple Vocabulary documents.

The DataDictionary is mostly concerned with the display and interpretation of data.

Table 6.1 DataDictionary

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|--------------------------------------|--|
| DataDictionary@version- Timestamp | A timestamp, in ISO8601 format that changes when the Data-Dictionary schema has changed. |

Table 6.1 DataDictionary

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|--|--|
| DataDictionary/Fields@version | A URI that changes when the Fields are no longer backward compatible with the previous version. |
| DataDictionary/Fields@timestamp | A timestamp, in ISO8601 format that changes whenever one or more Field elements are modified. |
| DataDictionary/Fields/Field/ MetaEntryId | A persistent identifier whose value never changes as long as the semantic definition of this field remains unchanged [dc]. The context of the identity is typically limited to the Provider context. In practice, this means that the MetaEntryId attribute is unique within a given Provider. |
| DataDictionary/Fields/Field/ LongName | The name of the field as it is known to the user. This is localizable, human-readable string. |
| DataDictionary/Fields/Field/ ShortName | An abbreviated field name that is also localizable and human readable. |
| DataDictionary/Fields/Field/ Description | A detailed description of the field. |
| DataDictionary/Fields/Field/ MaximumLength | The maximum display length of the field, in characters. This includes decimal points, whitespace and other punctuation. |
| DataDictionary/Fields/Field/ DataType | An enumerated type indicating the data type of the Field. the values of the enumeration are: Base64Binary, Boolean, Character, Currency, Date, DateTime, Decimal, HexBinary, Integer, Lookup, LookupMulti, UnsignedInteger, Time. |
| DataDictionary/Fields/Field/ Precision | For the DataType Decimal, the number of digits to the right of the decimal point when formatted. |
| DataDictionary/Fields/Field/ SecurityClass | An enumerated type indicating the Information Security Guidelines information class |
| DataDictionary/Fields/Field/ Alignment | An enumerated type indicating the alignment of the data display. The values of the enumeration are: Left, Right, Center, Justify. |
| DataDictionary/Fields/Field/ UseSeparator | A Boolean type indicating that a numeric value MAY be displayed with a thousands separator. |
| DataDictionary/Fields/Field/ LookupName | An external identifier to the Lookup Metadata for this field. LookupName or Map becomes a required element if the DataType is Lookup or LookupMulti. |
| DataDictionary/Fields/Field/ Units | An enumerated type that indicates the units of measure for this Field. The values for the enumeration are: Feet, Meters, SqFt, SqMeters, Acre and Hectare. |

Table 6.1 DataDictionary

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|--------------------------------------|---|
| DataDictionary/Fields/Field/IsUnique | A Boolean type that indicates this field is a unique identifier for the record. |
| DataDictionary/Fields/Field/Charset | A string representing the IANA character set. The default value is ISO-8859-1. |
| DataDictionary/Fields/Field/Locale | A string representing the LTRU [ltrul] registry value indicating the locale. The locale is used to determine character set, text direction, decimal and thousands character, time and date formats and other cultural determinants. The default value is en_US-ISO-98859-1. |

Figure 6.19 Sample DataDictionary Metadata

```

<retsdd:DataDictionary
  xmlns:retsdd="http://www.rets.org/ns/DataDictionary/200604"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.rets.org/ns/DataDictionary/200604 DataDictionary.xsd"
  versionTimestamp="2006-04-07T00:00:00Z">
  <Fields
    version="uri://tempuri.org/residential/datadictionary/200602"
    timestamp="2006-03-21T00:00:00Z">
  <Field>
    <MetaEntryId>PropertyCity</MetaEntryId>
    <LongName>City</LongName>
    <ShortName>City</ShortName>
    <Description>Physical Property City</Description>
    <MaximumLength>100</MaximumLength>
    <DataType>Character</DataType>
    <SecurityClass>Public</SecurityClass>
    <Alignment>Left</Alignment>
    <UseSeparator>>false</UseSeparator>
    <Units>>false</Units>
    <Searchable>>true</Searchable>
    <IsUnique>>false</IsUnique>
    <LookupName>City</LookupName>
    <Charset>ISO-8859-1</Charset>
    <Locale>en_US.ISO-8859-1</Locale>
  </Field>
  <Field>
    <MetaEntryId>MarketArea</MetaEntryId>
    <LongName>Marketing_Area</LongName>
    <ShortName>Area</ShortName>
    <Description>Market Area Designation</Description>
    <MaximumLength>100</MaximumLength>
    <DataType>Character</DataType>
    <SecurityClass>Public</SecurityClass>
  </Field>
  </Fields>
</retsdd:DataDictionary>

```

```

    <Alignment>Left</Alignment>
    <UseSeparator>>false</UseSeparator>
    <Units>>false</Units>
    <Searchable>>true</Searchable>
    <IsUnique>>false</IsUnique>
    <LookupName>Area</LookupName>
    <Charset>ISO-8859-1</Charset>
    <Locale>en_US.ISO-8859-1</Locale>
  </Field>
</Fields>
</retsdd:DataDictionary>

```

R0144 A Provider **MAY** support the DataDictionary metadata format.

R0046 If a Provider supports the DataDictionary metadata format, there **MUST** be one and only one DataDictionary for a specific Resource.

R0047 A DataDictionary Metadata document **MUST** validate against the RETS DataDictionary.xsd schema.

R0048 If a DataDictionary field contains a LookupName, it **MUST** reference a valid entry in a LookupList document.

6.4.4 LookupList

The LookupList [Metadata document](#) provides reusable Maps. Maps can be used across Resources.

Table 6.2 LookupList and Map Elements

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|-----------------------------|--|
| LookupList@versionTimestamp | A timestamp in ISO8601 format that changes whenever the LookupList schema has changed. |
| LookupList/Map | A Map of all of the name, value pairs for a given Lookup. A LookupList MUST contain one or more Maps. |
| LookupList/Map@name | A unique identifier for this Map. It is referenced by the LookupName in the Vocabulary and DataDictionary documents. |
| LookupList/Map@timestamp | A timestamp, in ISO8601 format that changes whenever the Map is modified. |
| LookupList/Map@version | A URI that changes when the Map is no longer backward compatible with the previous version. |
| LookupList/Map/MapEntry | A name, value pair entry for a given Map. Maps MUST contain one or more MapEntry elements. |

Table 6.2 LookupList and Map Elements

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|-----------------------------------|--|
| LookupList/Map/MapEntry/ Name | The human-readable key presented to the user |
| LookupList/Map/MapEntry/ Value | The value that is passed to the Provider in the Query. |

Figure 6.20 Sample LookupList

```
<retsl1:LookupList
  xmlns:retsl1="http://www.rets.org/ns/LookupList/200604"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.rets.org/ns/LookupList/200604 LookupList.xsd"
  versionTimestamp="2006-04-07T00:00:00Z">
  <Map name="City"
    timestamp="2006-03-01T09:30:47.0Z"
    version="http://www.myserver.com/02/01">
    <MapEntry>
      <Name>CLE</Name>
      <Value>Cleveland</Value>
    </MapEntry>
  </Map>
</retsl1:LookupList>
```

R0049 If a Field in a Provider’s Vocabulary or DataDictionary Metadata contains a LookupName, the RETS Provider **MUST** support LookupList Metadata.

R0050 LookupList documents **MUST** validate against the standard LookupList.xsd schema.

6.4.5 Formats and Extensions to Metadata

A Provider may choose to support custom or local metadata. In RETS1 this was accomplished through a naming convention where extensions to the Metadata had names prepended with “X_”.

Providers may define custom metadata as a Schema document. Schemas should follow the Best Practices Guideline in the RETS2 Resources and Payloads document. Custom or local metadata must be advertised in the ResourceList as a MetadataFormat.

Like all other RETS2 Standard payloads, RETS Standard Metadata documents are extensible. Standard metadata documents of ResourceList, LookupList, Vocabulary and DataDictionary can be extended using the *xs:any* tag following the last element in the standard schema.

Figure 6.21 Schema Extension Point Fragment

```
<xs:any namespace="##other"/>
```

R0051 A Provider **MAY** advertise all custom/local Metadata as MetadataFormats in their ResourceList.

R0052 Custom or local metadata documents **MUST** validate against the referenced schema.

6.5 The UserInfo Resource and UserInformation Response

The UserInfo Resource is a searchable resource that provides a UserInformation document containing detailed information about a user. This document contains similar information to the RETS1 Login response. Unlike RETS1, there are no restrictions on the length of the contents of elements in a UserInformation document.

If a Provider supports the UserInfo Resource, it must advertise this using the WellKnown Resource "UserInfo" in its ResourceList. Also, an OutputFormat must be included with a reference to the UserInformation schema.

Figure 6.22 Sample ResourceList with UserInfo Resource

```
<retsrsl:ResourceList version="2006-04-07T00:00:00Z"
  xmlns:retsrsl="http://www.rets.org/ns/ResourceList/200604"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.rets.org/ns/ResourceList/200604">
  <Resource timestamp="2006-03-01T09:30:47.0Z"
    version="http://www.myserver.com/02/01">
    <ResourceName>UserInfo</ResourceName>
    <OutputFormats>
      <OutputFormat>
        <FormatURL>
          http://www.rets.org/schema/UserInformation.xsd
        </FormatURL>
        <Description>
          RETS standard UserInformation schema
        </Description>
      </OutputFormat>
    </OutputFormats>
  </Resource>
  ...
</retsrsl:ResourceList>
```

Table 6.3 UserInformation Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|---|---|
| UserInformation@version- Timestamp | A timestamp, in ISO8601 format that changes whenever the UserInformation schema has changed. |
| UserInformation@id | The internal system user identifier. |
| UserInformation/Level | An implementation dependent level for the user. |
| UserInformation/Class | An implementation dependent class description of the user role. An example might be "ListingAgent". |
| UserInformation/Broker- Code | An internal system identifier for the broker that the user is associated with. |
| UserInformation/Broker- Branch | An internal system identifier for the broker branch that the user is associated with. |
| UserInformation/Member- Name | The full user name, as it would appear on any printed output. |
| UserInformation/Balance | If the Provider exposes account usage information, this value SHOULD represent the user-readable remaining balance , typically money, in the user system access account. |
| UserInformation/Passwor- dExpireDate | A timestamp in ISO8601 format that provides the user system access account password expiration time and date. |

The following is a sample UserInformation document fragment. The UserInformation document returned will only contain information for that user who is currently authenticated and has issued the request.

Figure 6.23 Sample UserInformation Fragment

```

<UserInformation
  xmlns="http://rets.org/ns/UserInformation/200604"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://rets.org/ns/UserInformation/200604 UserInformation.xsd"
  version="2006-04-07T00:00:00Z" id="JS19348">
  <Level>1</Level>
  <Class>Agent</Class>
  <AgentCode>AG7655</AgentCode>
  <BrokerCode>BC8787</BrokerCode>
  <BrokerBranch>BR7655</BrokerBranch>
  <MemberName>Joe Smith</MemberName>
  <Balance>150.00</Balance>
  <PasswordExpireDate>

```

```
2006-03-03T13:03:04-05:00
</PasswordExpireDate>
</UserInformation>
```

R0128 A Provider **MAY** support the WellKnown UserInfo Resource.

R0129 A Provider who supports the WellKnown UserInfo Resource **MUST** advertise a UserInfo Resource in its ResourceList.

R0130 A Provider who supports the WellKnown UserInfo Resource **MUST** advertise the UserInfoInformation.xsd as an OutputFormat for that Resource.

R0131 A UserInfoInformation response document **MUST** validate against the UserInfoInformation.xsd schema.

6.6 Message of the Day

MessageOfDay is a WellKnown Resource that a Provider may use to post messages about the system to users, commonly referred to as Message-of-the-Day. This action is similar in purpose to the RETS1 Action response.

The MessageOfDay is a Searchable resource. If a Provider supports the MessageOfDay resource, it must be advertised using the WellKnown Resource “MessageOfDay” in the ResourceList. Providers who offer the MessageOfDay resource **MUST** support the RETS MessageOfDay.xsd schema as an OutputFormat. Providers **SHOULD** provide a Vocabulary metadata format for the MessageOfDay.

Figure 6.24 Sample ResourceList with MessageOfDay Resource

```
<retsrsl:ResourceList version="2006-04-07T00:00:00Z"
  xmlns:retsrsl="http://www.rets.org/ns/ResourceList/200604"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.rets.org/ns/ResourceList/200604">
  <Resource timestamp="2006-03-01T09:30:47.0Z"
    version="http://www.myserver.com/02/01">
    <ResourceName>MessageOfDay</ResourceName>
    <VocabularyNames>
      <VocabularyName>Local</VocabularyName>
    </VocabularyNames>
    <OutputFormats>
      <OutputFormat>
        <FormatURL>
          http://www.rets.org/schema/MessageOfDay.xsd
        </FormatURL>
        <Description>
          RETS standard Message of Day format to provide
          important system bulletins.
        </Description>
      </OutputFormat>
    </OutputFormats>
  </Resource>
</ResourceList>
```

```

        </Description>
    </OutputFormat>
</OutputFormats>
    <MetadataFormats>
        <MetadataFormat>
            <FormatURL>
                http://www.rets.org/schema/Vocabulary.xsd
            </FormatURL>
            <Description>
                RETS standard Vocabulary metadata schema
            </Description>
        </MetadataFormat>
    </MetadataFormats>
</Resource>
    ...
<retsrsl:ResourceList>

```

Table 6.4 MessageOfDay Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|-------------------------------------|---|
| MessageOfDay@version- Timestamp | A timestamp in ISO8601 format that changes whenever the MessageOfDay schema is changed. |
| MessageOfDay/Messages/ Timestamp | A timestamp in ISO8601 format for the date-time that the message was posted. |
| MessageOfDay/Messages/ Details | An implementation dependent string representing the body of the message for the user or class of users. Requestors SHOULD display the message to the end-user where appropriate, based on the Requesting application . |

Figure 6.25 Sample MessageOfDay Fragment

```

<Messages
  xmlns="http://rets.org/ns/MessageOfDay"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://rets.org/ns/MessageOfDay MessageOfDay.xsd"
  version="2006-04-07T00:00:00Z">
  <Message>
    <Timestamp>2006-03-03T13:03:04-05:00</Timestamp>
    <Details>
      The system will be down for maintenance at 8pm today.
    </Details>
  </Message>
</Messages>

```

R0132 A Provider **MAY** support the WellKnown MessageOfDay Resource.

- R0133 A Provider who supports the WellKnown MessageOfDay Resource **MUST** advertise a MessageOfDay Resource in the ResourceList.
- R0134 A Provider who supports the WellKnown MessageOfDay Resource **MUST** advertise the RETS standard MessageOfDay.xsd as a MetadataFormat for that Resource.
- R0135 A MessageOfDay metadata document **MUST** validate against the RETS MessageOfDay.xsd schema.

7 Metadata Actions

RETS2 supports the following Metadata Actions: GetResourceList, GetLookupList, and GetMetadata.

7.1 GetResourceList - Request Document

The GetResourceList Action may be used with no parameters or one of the optional parameter shown in the table.

Table 7.1 GetResourceList - Request Document Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|----------------------|---|
| Resource | Optional name of the Resource. This matches the ResourceList/ResourceName. |
| Timestamp | A timestamp in ISO8601 format format that corresponds to ResourceList@timestamp |

Figure 7.1 ResourceListRequest Fragment

```
<xsd:element name="ResourceListRequest">
  <xsd:complexType>
    <xsd:choice minOccurs="0">
      <xsd:element name="Resource" type="xsd:NMTOKEN"/>
      <xsd:element name="Timestamp" type="xsd:dateTime"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

7.1.1 Examples

If no parameter is supplied, GetResourceList returns a *ResourceList* document containing all *Resources* supported by the RETS Provider.

Figure 7.2 Example ResourceListRequest document with No parameters

```
<ResourceListRequest/>
```

If only a Resource is supplied, GetResourceList returns a ResourceList document for only that Resource.

Figure 7.3 Example ResourceListRequest document for a specific Resource

```
<ResourceListRequest>
  <Resource>Residential</Resource>
</ResourceListRequest>
```

If a Timestamp is supplied, GetResourceList returns a ResourceList document containing all Resources whose timestamp is newer than or equal to that Timestamp.

Figure 7.4 Example ResourceListRequest document for a Timestamp

```
<ResourceListRequest>
  <TimeStamp>2006-01-02T09:40:04-05:00</TimeStamp>
</ResourceListRequest>
```

7.2 GetLookupList - Request Document

Table 7.2 GetLookupList - Request Document

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|----------------------|---|
| LookupName | An optional name of a Map. This matches the LookupList/Map@name value or the Vocabulary/Fields/Field/LookupName or the DataDictionary/Fields/Field/LookupName |
| TimeStamp | An optional timestamp in ISO8601 format that corresponds to LookupList/Map@timestamp |

Figure 7.5 LookupListRequest Fragment

```
<xsd:element name="LookupListRequest">
  <xsd:complexType>
    <xsd:choice minOccurs="0">
      <xsd:element name="LookupName" type="xsd:string"/>
      <xsd:element name="TimeStamp" type="xsd:dateTime"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

7.2.1 Examples

If no parameter is supplied, GetLookupList returns a LookupList document containing all Maps supported by the RETS Provider.

Figure 7.6 LookupListRequest for all LookupLists

```
<LookupListRequest/>
```

If only a LookupName is supplied, GetLookupList returns a LookupList document for only the Map whose name attribute matches the LookupName.

Figure 7.7 LookupListRequest for a specific LookupName

```
<LookupListRequest>
  <LookupName>City</LookupName>
</LookupListRequest>
```

If a Timestamp is supplied, GetLookupList returns a LookupList document containing all Maps whose timestamp is newer than or equal to that Timestamp.

Figure 7.8 LookupListRequest for a Timestamp

```
<LookupListRequest>
  <Timestamp>2006-01-02T09:40:04-05:00</Timestamp>
</LookupListRequest>
```

7.3 GetMetadata - Request Document

The GetMetadata Action must be used to retrieve Vocabulary documents, DataDictionary documents, or local/custom metadata documents, using the parameters defined in the table below.

Table 7.3 GetLookupList - Request Document Elements

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|----------------------|---|
| Resource | The required name of the <i>Resource</i> that matches entries from <i>ResourceList/Resource/ResourceName</i> . |
| MetadataFormatURL | The required URL to the schema defined in the <i>MetadataFormat</i> for this document. |
| Name | An optional unique reference; for Vocabulary, it is the Vocabulary name: <i>Vocabulary@name</i> ; DataDictionary does not require a <i>Name</i> . |

GetMetadata returns an instance document in the format requested in the MetadataFormat URL parameter.

Figure 7.9 MetadataRequest Fragment

```
<xsd:element name="MetadataRequest">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Resource" type="xsd:NMTOKEN"/>
      <xsd:element name="MetadataFormatURL" type="xsd:anyURI"/>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

7.3.1 Examples

The following *MetadataRequest* will retrieve the WellKnown Vocabulary document for the Residential *Resource*.

Figure 7.10 MetadataRequest for a Vocabulary Document

```
<MetadataRequest>
  <Resource>Residential</Resource>
  <MetadataFormatURL>http://www.rets.org/Vocabulary.xsd</MetadataFormatURL>
  <Name>WellKnown</Name>
</MetadataRequest>
```

The following *MetadataRequest* will retrieve a DataDictionary Document for the Residential *Resource*.

Figure 7.11 MetadataRequest for a DataDictionary Document

```
<MetadataRequest>
  <Resource>Residential</Resource>
  <MetadataFormatURL>http://www.rets.org/DataDictionary.xsd</
MetadataFormatURL>
</MetadataRequest>
```

The following *MetadataRequest* will retrieve a UserInformation Document for the UserInfo *Resource*.

Figure 7.12 MetadataRequest for a UserInformation Document

```
<MetadataRequest>
  <Resource>UserInformation</Resource>
  <MetadataFormatURL>
    http://www.rets.org/UserInformation.xsd
  </MetadataFormatURL>
</MetadataRequest>
```

The following *MetadataRequest* will retrieve a custom or local metadata format document, MyMetadata.xsd, for the Residential *Resource*.

Figure 7.13 MetadataRequest for a Custom/Local Metadata Document

```
<MetadataRequest>
  <Resource>Residential</Resource>
  <MetadataFormatURL>http://www.myserver.com/MyMetadata.xsd</
MetadataFormatURL>
</MetadataRequest>
```

7.4 ObjectResources and ObjectReferenceLists

Object Resources are special *Resource types* that describe available objects. RETS2 Providers must use Object Resources to describe supported objects. A Provider may choose to define one Object *Resource* to handle all objects, or instead may create multiple Object *Resource values* for specific purposes.

An Object may be one of the following well-known Object Types defined in the table below. These *type* names are identical to the same object types in RETS1.

Table 7.4 Well Known Object Resource Type Names

| OBJECT TYPE NAME | DESCRIPTION |
|------------------|---|
| Photo | An image related to the document in a common binary format like jpeg, although other formats are possible. Examples are Agent photos or pictures of the property. |
| Plat | A map or image of the property boundaries and location, typically within a larger context and typically within an ownership context, such as subdivision or township. |
| Video | A time-based image, with or without sound, in a common binary format like mpeg, although other formats are possible. An examples is a video tour of the property. |
| Audio | A sound clip. |
| Thumbnail | A low-resolution image related to the document. |
| Map | A map or image of the property. This is typically a street map or satellite map |
| VRImage | A multiple-view or panorama, possibly interactive image. |

If a Provider supports the retrieval or update of Objects, it must include Object *Resource values* in its *ResourceList*:

Figure 7.14 Metadata ResourceList with Object Resource

```
<retsrsl:ResourceList
  xmlns:retsmeta="http://www.rets.org/ns/Metadata"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.rets.org/ns/ResourceList ResourceList.xsd">
  <Resource
    version="/test/12/17/PropertyImages"
    timestamp="2006-01-02T09:40:04-05:00">
    <ResourceName>PropertyImages</retsmeta:ResourceName>
```

```

<VocabularyNames>
  <VocabularyName>System</VocabularyName>
  <VocabularyName>WellKnown</VocabularyName>
</VocabularyNames>
<OutputFormats>
  <OutputFormat>
    <ObjectType>Photo</ObjectType>
    <Description>jpgs and gifs</Description>
  </OutputFormat>
  <OutputFormat>
    <ObjectType>Thumbnail</ObjectType>
    <Description>
      low resolution jpgs and gifs
    </Description>
  </OutputFormat>
</OutputFormats>
</Resource>
</retsrl:ResourceList>

```

One Object *Resource* can be associated with multiple supported Resources. In order to provide references between Resources and Object Resources, and allow Requestors to retrieve specific object types for a given resource, a standard payload is provided called the *ObjectReferenceList*.

Any *Resource* that supports object retrieval **MUST** include an *ObjectReferenceList* schema in its *OutputFormats*.

Figure 7.15 Metadata Object Resource OutputFormat Fragment

```

<OutputFormat>
  <FormatURL>
    http://rets.org/ObjectReferenceList.xsd
  </FormatURL>
  <Description>
    Standard XML to describe a list of Object References
  </Description>
</OutputFormat>

```

A RETS2 Requestor **SHOULD** issue a search request to the Provider and specify *ObjectReferenceList* as the *OutputFormat* to find a list of objects, by *ObjectResource*, *referenceId*, and *ObjectType* that match the records returned by the search .

Figure 7.16 Sample of ObjectReferenceList Document

```

<object:ObjectReferenceList
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:object="http://rets.org/ns/ObjectReferenceList"
  xsi:schemaLocation="http://rets.org/ns/ObjectReferenceList
    ObjectReferenceList.xsd">
  <object:ObjectResource>

```

```

<object:ResourceName>PropertyImages</object:ResourceName>
<object:Objects>
  <object:Object referenceId="123454">
    <object:ObjectTypes type="Thumbnail">
      <object:ObjectType>
        <object:Offset>0</object:Offset>
        <object:ByteSize>64000</object:ByteSize>
        <object:ObjectId>124</object:ObjectId>
        <object:URL>
          http://myserver.com/pics/124.jpg
        </object:URL>
        <object:Description>
          House Front View
        </object:Description>
        <object:ContentType>
          image/jpeg
        </object:ContentType>
        <object:Copyright/>
        <object:Timestamp>
          2006-03-03T13:03:04-05:00
        </object:Timestamp>
      </object:ObjectType>
    </object:ObjectTypes>
  <object:ObjectTypes type="Photo">
    <object:ObjectType>
      <object:Offset>0</object:Offset>
      <object:ByteSize>128000</object:ByteSize>
      <object:ObjectId>474</object:ObjectId>
      <object:URL>
        http://myserver.com/pics/474.jpg
      </object:URL>
      <object:Description>
        House Front View
      </object:Description>
      <object:ContentType>
        image/jpeg
      </object:ContentType>
      <object:Copyright/>
      <object:Timestamp>
        2006-03-03T13:03:04-05:00
      </object:Timestamp>
    </object:ObjectType>
  </object:ObjectTypes>
</object:Object>
  <object:Object referenceId="323215">
    <object:ObjectTypes type="Thumbnail">
      <object:ObjectType>
        <object:Offset>1</object:Offset>

```

```

    <object:ByteSize>26000</object:ByteSize>
    <object:ObjectId>225</object:ObjectId>
    <object:URL>
      http://myserver.com/pics/225.gif
    </object:URL>
    <object:Description>
      Whole House
    </object:Description>
    <object:ContentType>
      image/gif
    </object:ContentType>
    <object:Copyright/>
    <object:Timestamp>
      2006-03-03T13:03:04-05:00
    </object:Timestamp>
  </object:ObjectType>
</object:ObjectTypes>
<object:ObjectTypes type="Photo">
  <object:ObjectType>
    <object:Offset>0</object:Offset>
    <object:ByteSize>52000</object:ByteSize>
    <object:ObjectId>009998</object:ObjectId>
    <object:URL>
      http://myserver.com/pics/009998.jpg
    </object:URL>
    <object:Description>
      Whole House
    </object:Description>
    <object:ContentType>image/jpeg
  </object:ContentType>
  <object:Copyright/>
  <object:Timestamp>
    2006-03-03T13:03:04-05:00
  </object:Timestamp>
  </object:ObjectType>
</object:ObjectTypes>
</object:Object>
</object:Objects>
</object:ObjectResource>
<object:ObjectResource>
  <object:ResourceName>MyPhotos</object:ResourceName>
  <object:Objects>
    <object:Object referenceId="123454">
      <object:ObjectTypes type="Thumbnail">
        <object:ObjectType>
          <object:Offset>0</object:Offset>
          <object:ByteSize>64000</object:ByteSize>
          <object:ObjectId>124</object:ObjectId>

```

```

<object:URL>
    http://myserver.com/pics/mine/124.jpg
</object:URL>
<object:Description>
    House Front View
</object:Description>
<object:ContentType>
    image/jpeg
</object:ContentType>
<object:Copyright/>
<object:Timestamp>
    2006-03-03T13:03:04-05:00
</object:Timestamp>
</object:ObjectType>
</object:ObjectTypes>
<object:ObjectTypes type="Photo">
<object:ObjectType>
    <object:Offset>0</object:Offset>
    <object:ByteSize>128000</object:ByteSize>
    <object:ObjectId>474</object:ObjectId>
    <object:URL>
        http://myserver.com/pics/mine/474.jpg
    </object:URL>
    <object:Description>
        House Front View
    </object:Description>
    <object:ContentType>
        image/jpeg
    </object:ContentType>
    <object:Copyright/>
    <object:Timestamp>
        2006-03-03T13:03:04-05:00
    </object:Timestamp>
    </object:ObjectType>
</object:ObjectTypes>
</object:Object>
<object:Object referenceId="323215">
    <object:ObjectTypes type="Thumbnail">
    <object:ObjectType>
        <object:Offset>0</object:Offset>
        <object:ByteSize>26000</object:ByteSize>
        <object:ObjectId>225</object:ObjectId>
        <object:URL>
            http://myserver.com/pics/mine/225.gif
        </object:URL>
        <object:Description>
            Whole House
        </object:Description>
    </object:ObjectType>
    </object:ObjectTypes>
</object:Object>

```

```

    <object:ContentType>
      image/gif
    </object:ContentType>
    <object:Copyright/>
    <object:Timestamp>
      2006-03-03T13:03:04-05:00
    </object:Timestamp>
  </object:ObjectType>
</object:ObjectTypes>
<object:ObjectTypes type="Photo">
  <object:ObjectType>
    <object:Offset>0</object:Offset>
    <object:ByteSize>52000</object:ByteSize>
    <object:ObjectId>454</object:ObjectId>
    <object:URL>
      http://myserver.com/pics/mine/454.jpg
    </object:URL>
    <object:Description>
      Whole House
    </object:Description>
    <object:ContentType>
      image/jpeg
    </object:ContentType>
    <object:Copyright/>
    <object:Timestamp>
      2006-03-03T13:03:04-05:00
    </object:Timestamp>
  </object:ObjectType>
</object:ObjectTypes>
</object:Object>
</object:Objects>
</object:ObjectResource>
</object:ObjectReferenceList>

```

The *ObjectReferenceList* is organized first by *ObjectResource* elements. *ObjectResource* elements within this document are those that contain objects that match the records returned by the user's .

Figure 7.17 ObjectResource Fragment

```

<object:ObjectResource>
  <object:ResourceName>PropertyImages</object:ResourceName>
  ...
<object:ObjectResource>
  <object:ResourceName>MyPhotos</object:ResourceName>
  ...

```

Within the *ObjectResource* are the references and descriptions of specific objects that matched the results. Each object has a *referenceId*, which refers to the unique key value for a record. In many cases

that *referenceId* will be a ListingID, however it may be some other internal unique identifier. This unique key is a reference to the object within the Object Resource. It is used to retrieve the actual object.

Figure 7.18 ObjectResource referenceId Fragment

```
<object:ObjectResource>
  <object:ResourceName>PropertyImages</object:ResourceName>
  <object:Objects>
    <object:Object referenceId="123454">
      ...
    </object:Object>
  </object:Objects>
</object:ObjectResource>
```

A given *referenceId* may be used to retrieve multiple object types. Elements of the complexType *ObjectType* are contained within the Object element and are named after the well-known ObjectTypes listed in the Well-Known Object Resource Names Table.

Figure 7.19 Multiple Object ObjectType Fragment

```
<object:Object referenceId="123454">
  <object:ObjectTypes type="Thumbnail">
    <object:ObjectType>
      <object:Offset>0</object:Offset>
      <object:ByteSize>64000</object:ByteSize>
      <object:ObjectId>124</object:ObjectId>
      <object:URL>
        http://myserver.com/pics/124.jpg
      </object:URL>
      <object:Description>
        House Front View
      </object:Description>
      <object:ContentType>
        image/jpeg
      </object:ContentType>
      <object:Copyright/>
      <object:Timestamp>
        2006-03-03T13:03:04-05:00
      </object:Timestamp>
    </object:ObjectType>
  </object:ObjectTypes>
  <object:ObjectTypes type="Photo">
    <object:ObjectType>
      <object:Offset>0</object:Offset>
      <object:ByteSize>128000</object:ByteSize>
      <object:ObjectId>474</object:ObjectId>
      <object:URL>
        http://myserver.com/pics/474.jpg
      </object:URL>
    </object:ObjectType>
  </object:ObjectTypes>
</object:Object>
```

```

</object:URL>
<object:Description>
  House Front View
</object:Description>
<object:ContentType>
  image/jpeg
</object:ContentType>
<object:Copyright/>
<object:Timestamp>
  2006-03-03T13:03:04-05:00
</object:Timestamp>
</object:ObjectType>
</object:ObjectTypes>
</object:Object>

```

Table 7.5 The ObjectType Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|------------------------|---|
| ObjectType/Offset | The required index of the object, beginning with '0'. |
| ObjectType/ByteSize | The required size, in bytes, of the object. |
| ObjectType/Timestamp | The required last-modification timestamp in ISO8601 format, of the Object. |
| ObjectType/ObjectId | The required object identifier. This MUST be unique across the <i>ObjectResource</i> . This identifier may be used as the <i>Objec-tId</i> in the to retrieve this object. |
| ObjectType/URL | An optional URL location for the object. If provided, the Object may be retrieved using an HTTP GET on the URL. |
| ObjectType/Description | An optional description or caption for the object. |
| ObjectType/ContentType | An optional type or subtype of the returned media. An exam-ple is "image/jpg". |
| ObjectType/Copyright | An optional string containing trademark or copyright infor-mation pertaining to the media. |

Figure 7.20 Object Photo Fragment

```

<object:ObjectTypes type="Photo">
  <object:ObjectType>
    <object:Offset>0</object:Offset>
    <object:ByteSize>128000</object:ByteSize>
    <object:ObjectId>474</object:ObjectId>
    <object:URL>
      http://myserver.com/pics/474.jpg

```

```
</object:URL>
<object:Description>
  House Front View
</object:Description>
<object:ContentType>
  image/jpeg
</object:ContentType>
<object:Copyright/>
<object:Timestamp>
  2006-03-03T13:03:04-05:00
</object:Timestamp>
</object:ObjectType>
</object:ObjectTypes>
```

R0053 A Provider **MUST** support at least **ONE** Object Resource if they offer Object retrieval or update.

R0054 A Provider who has an Object Resource **MUST** support the ObjectReferenceList payload for all Resources that reference that Object Resource.

R0055 ObjectReferenceList documents **MUST** validate against the standard ObjectReferenceList.xsd schema.

R0056 An Object Resource **MUST** support well-known ObjectTypes.

R0057 An Object Resource **MAY** support additional ObjectTypes.

R0058 The referenceld of an Object **MUST** correspond to the keyfield attribute value for a Vocabulary.

8 Update Action

The Update Action is similar to the RETS1 Update functionality, encompassing not only the update functionality but also create and delete. Update in RETS2 provides additional functionality in that it can be used to not only add/update records, but also to add/update Objects for a given Resource.

8.1 Update Request

The Update Request consists of two sections: The manifest describing the item and containing the payload content and the attributes describing the resource and actions.

The *resource* attribute corresponds to a *Resource* from the Provider's *ResourceList*. It identifies the *Resource* being updated.

The *format* attribute corresponds to a supported *InputFormat* that describes the format of the content of the Manifest. This *format* must be one of the *InputFormat* elements from the *ResourceList* for the resource identified in the *resource* attribute. It is either a URI for a Schema or a RETS2 *ObjectType*.

The InputFormats a Provider will accept for the updating of items of a given Resource type may be RETS WellKnown payloads, but will likely specify a localized schema. See "RETS SOAP Message" below for reference.

The *type* attribute specifies the type of update action to be applied using the contents of the Manifest.

- "Create" will add the Manifest content as a new item for that Resource.
- "Delete" will delete the Manifest content from that Resource.
- "Merge" and "Replace" are both forms of "Update":
- "Merge" provides an element-by-element overlay functionality, updating only those elements provided in the Manifest content and not updating any other elements within a record. Empty values are not updated.
- "Replace" provides a delete-insert functionality, replacing an entire document for a Resource with the content of the Manifest. Empty values in the Manifest are updated in the Resource document.

R0060 A Provider supporting the Update Action for well-known Resources **MAY** supply the standard payload for that Resource as an InputFormat.

R0061 The UpdateRequest document **MUST** conform to the RETS UpdateRequest WSDL element.

R0062 The UpdateRequest resource attribute **MUST** correspond to a ResourceName in the Provider metadata ResourceList.

R0063 The UpdateRequest format attribute **MUST** correspond to an InputFormat for the named Resource.

- R0064 The UpdateRequest **MUST** specify a format.
- R0065 The UpdateRequest **MUST** specify a resource.
- R0066 The UpdateRequest **MUST** contain a Manifest with content.
- R0067 The UpdateRequest **MUST** specify a type.

Figure 8.1 Update Request Fragment

```

<xsd:element name="UpdateRequest">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Manifest" type="rets:ManifestType"/>
    </xsd:sequence>
    <xsd:attribute name="resource" type="xsd:string" use="required"/>
    <xsd:attribute name="format" type="xsd:string" use="required"/>
    <xsd:attribute name="type" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="Create"/>
          <xsd:enumeration value="Merge"/>
          <xsd:enumeration value="Replace"/>
          <xsd:enumeration value="Delete"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>

```

8.2 UpdateResponse

The Update Response consists of a Responses element containing one or more Response elements.

Table 8.1 The Update Response Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|----------------------|---|
| Response@uid | An optional unique identifier for a record. A ListingId would be an example or other unique system identifier. In the fault case, a response may not have a <i>uid</i> because no record was created. |
| Response@resource | A required attribute that matches the <i>Resource</i> attribute of the <i>UpdateRequest</i> . It is the value of the <i>Resource</i> that was updated for this response |

Table 8.1 The Update Response Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|------------------------|--|
| Response@correlationId | An optional identifier that correlates this <i>Response</i> back to the <i>UpdateRequest taskId</i> if the <i>taskId</i> is set |
| Response@severity | A required value from the enumerated set of None, Warning or Fatal. Successful transactions must have a severity value of Warning or None |
| Response/Faults | An optional container that provides a structured edit error list for <i>UpdateRequests</i> that contain multiple records. Providers may choose to use only SOAP faults if they do not provide a structured edit list from within a <i>Response</i> . A Provider may use either a SOAP Fault or the <i>Response/Faults</i> collection <i>Response Faults</i> but must not use both. |

The following is an example of a successful *UpdateResponse* where the *severity equals* = “None”, indicating no warnings or errors occurred.

Figure 8.2 Example of a Successful UpdateResponse Document with No Warnings

```
<UpdateResponse>
  <Responses>
    <Response uid="5654432"
      resource="Residential"
      correlationTaskID="34"
      severity="None">
    </Response>
  </Responses>
</UpdateResponse>
```

The following is an example of a successful *UpdateResponse* with warnings. The *severity* level equals “Warning”. The warnings did not stop the update transaction from occurring. A *Faults collection list* provides a description of the warnings.

Figure 8.3 Example of a Successful UpdateResponse Document with Warnings

```
<UpdateResponse>
  <Responses>
    <Response uid="5654432"
      resource="Residential"
      correlationTaskID="1"
      severity="Warning">
    <Faults>
      <Fault severity="Warning">
        <Message>
          You cannot update a calculated field.
        </Message>
        <FieldList>
          <Field>DaysOnMarket</Field>
        </FieldList>
      </Fault>
    </Faults>
  </Response>
</UpdateResponse>
```

```

        </FieldList>
    </Fault>
</Faults>
</Response>
</Responses>
</UpdateResponse>

```

The following is an example of an *UpdateResponse* that was not successful. There is no *uid* because no record was created and the *severity* level equals “Fatal”. The Faults describe the error that occurred and the ~~field~~ *Field/Name* values associated with it.

Figure 8.4 Example of a failed UpdateResponse Document with Fatal Faults

```

<UpdateResponse>
  <Responses>
    <Response resource="Residential"
      correlationTaskID="1"
      severity="Fatal">
      <Faults>
        <Fault severity="Fatal">
          <Type>Forbidden</Type>
          <Message>
            Only the listing agent
            may update fields.
          </Message>
          <FieldList>
            <Field>ExpirationDate</Field>
            <Field>ListPrice</Field>
          </FieldList>
        </Fault>
      </Faults>
    </Response>
  </Responses>
</UpdateResponse>

```

Providers may also choose to return Update errors using the SOAP Fault rather than the RETS FaultList. Below is an example of the same [class of error using the SOAP Fault mechanism](#). (Here, a user tried to update a field [they he](#) did not have authorization to update.) ~~using the SOAP Fault mechanism.~~

Figure 8.5 Example of an unsuccessful Update transaction returning a SOAP Fault

```

<env:Fault>
  <env:Code>
    <env:Value>Forbidden</env:Value>
  </env:Code>
  <env:Reason>
    <env:Text xml:lang="en">
      Only the listing agent may update
      fields (ExpirationDate).
    </env:Text>
  </env:Reason>
</env:Fault>

```

```
</env:Reason>
</env:Fault>
```

- R0068 The UpdateResponse document **MUST** conform to the **RETS WSDL** UpdateResponse element.
- R0069 The value of the UpdateResponse resource attribute **MUST** correspond to the value of the related UpdateRequest resource attribute.
- R0070 The UpdateResponse **MAY** contain a uid attribute.
- R0071 The UpdateResponse **MUST** contain a resource attribute.
- R0072 The UpdateRequest **MAY** contain a correlationID attribute.
- R0073 The correlationID attribute value **MUST** correspond to an UpdateRequest taskID attribute value.
- R0074 In the case of errors, the UpdateResponse **MUST** contain either Faults or throw a **SOAP** Fault.
- R0075 In the case of warnings, the UpdateResponse **MAY** contain Faults.

Figure 8.6 Update Response Fragment

```
<xsd:element name="UpdateResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Responses">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Response">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element
                    name="Faults"
                    type="rets:RetsFaultList"
                    minOccurs="0"/>
                </xsd:sequence>
              <xsd:attribute name="uid"
                type="xsd:string"/>
              <xsd:attribute name="resource"
                type="xsd:NMTOKEN"
                use="required"/>
              <xsd:attribute name="correlationID"
                type="xsd:NMTOKEN"/>
              <xsd:attribute name="severity"
                type="rets:SeverityType"
                use="required"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence>
```

```

    </xsd:complexType>
  </xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

```

Faults use the RETSFaultList Structure. Each Fault is composed of the following elements and attributes:

Table 8.2 The RETSFaultList Fault Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|----------------------|--|
| Fault@severity | An optional enumerated type that a Provider may use to identify the error as either Fatal or Warning. Fatal indicates that processing stopped and the action was not completed. Warning indicates that the action completed but warnings were reported |
| Fault/Type | A required value that identifies the <i>Fault</i> type from the list of Update Faults. This element corresponds to the RETS1 error names. |
| Fault/Message | An optional implementation-specific error message for the Type. |
| Fault/FieldList | An optional collection list of the Field/Name values that contained errors for this Response |

Figure 8.7 RetsFaultList Fragment

```

<xsd:complexType
  name="RetsFaultList">
  <xsd:sequence>
    <xsd:element
      name="Fault"
      maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element
            name="Type"
            type="xsd:NMTOKEN" />
          <xsd:element
            name="Message"
            type="xsd:string"
            minOccurs="0" />
          <xsd:element
            name="FieldList"
            minOccurs="0" >
            <xsd:complexType>
              <xsd:sequence>

```

```
        <xsd:element
            name="Field"
            type="xsd:string"
            maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
<xsd:attribute
    name="severity"
    type="rets:SeverityType"
    use="required"/>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
```

R0076 An UpdateResponse Fault **MAY** contain a Message.

R0077 An UpdateResponse Fault **MAY** contain a FieldList.

R0078 An UpdateResponse Fault **MAY** specify a severity attribute.

R0079 An UpdateResponse Fault **MUST** contain a Type.

9 Search Action

Search in RETS2 combines the RETS1 transactions of Search, GetObject and ServerInformation. It simplifies those transactions by reducing **the number of transaction types** ~~their number~~ and **provides providing** a unified method of querying each Resource. Search payloads that map to the RETS1 transaction responses are noted in the RETS2 Resources and Payloads document.

9.1 OutputObject Types

RETS2 OutputObject types will typically be XML documents, defined using the combination of the Resource and schema. To support legacy applications or special needs, **two** additional formats are supported: Delimited and **Encoded-Delimited**.

9.1.1 **Common General Properties of Delimited and Encoded-Delimited OutputObjects**

An OutputObject that is delimited, either the Delimited or Encoded-Delimited, use a defined set of delimiters to simplify processing. The following delimiters are supported: comma, tab and pipe. Only one delimiter token can be used per response.

Delimited or **Encoded-Delimited OutputObjects files** must use UTF-8 encoding. RETS2 uses the convention of the 'C' programming language family (C, C++, C#, Java) of escaping special characters within a field. A backslash is used to escape special characters as described in the following table.

Table 9.1 Delimited OutputFormat Field Character Encoding Delimiters

| ESCAPE ENCODING | DESCRIPTION |
|-----------------|-----------------|
| \b | backspace |
| \f | formfeed |
| \t | horizontal tab |
| \” | double quote |
| \n | newline |
| \’ | single quote |
| \r | carriage return |
| \\ | backslash |

9.1.2 Delimited OutputObject

The Delimited OutputFormat replaces the RETS1 COMPACT-DECODED format. Delimited is a standard delimited file, with column names in the first row and data records in subsequent rows.

String literals must be enclosed in quotes. All delimited data values will be decoded, that is, in final presentation form.

If the field is multi-valued, the values must be separated by a comma character and a single space. The final value does not have a separator appended to the value.

If the field is a Boolean, the values must be either “true” or “false”.

9.1.3 Encoded-Delimited OutputObject

The Encoded-Delimited OutputFormat replaces the RETS1COMPACT format. Encoded-Delimited is a standard delimited file, with column names in the first row and data records in subsequent rows. Lookup values are encoded with the key value from the LookupList\Name element. String literals must be enclosed in quotes. All data values will be encoded, that is, in compact presentation form. Presentation for human or end users may require a translation from the encoded name to the decoded value.

Encoded data is data that is stored as an enumeration, multivalue, boolean, abbreviation or arbitrary string with its meaning defined elsewhere in the system.

If the field is multi-valued, the values must be separated by a comma character. The final value does not have a separator appended to the value.

If the field is a Boolean, the values must be a single character, either ‘1’ for true values and ‘0’ for false values.

9.2 Search Request Document

Table 9.2 The SearchRequestType Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|------------------------|--|
| SearchRequest/Resource | A required name of the Resource to be queried. The single value matches values from the ResourceList/Resource/ResourceName metadata. |

Table 9.2 The SearchRequestType Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|---|--|
| SearchRequest/Query | A required query statement. to be performed The query statement should follow the rules and operators for the queryType attribute and use the Vocabulary/Field/Name values from the Vocabulary/Name that matches the value of the SearchRequest@vocabulary attribute or use qualified Vocabulary/Field/Name values where the Field/Name is prepended with the Vocabulary/Name and separated from the Field/Name by the ‘.’ character. Mixed vocabularies are permitted using qualified Vocabulary/Field/Name values described above. Providers SHOULD respond with a fault for cases where query statements use unknown name values, invalid operators, unknown Vocabulary/Name values or empty query statements. |
| SearchRequest/Query@queryType | An optional type of the query. Providers MUST support RQL. If absent, the value has a default of ‘RQL’. Providers MAY support additional query types. If the value of the query type is not known, Providers SHOULD respond with a fault. |
| SearchRequest/Query@vocabulary | An optional name of the vocabulary used in the query. The single value matches values from the ResourceList/Resource/VocabularyNames/VocabularyName. If absent, the value has a default of ‘WellKnown’. Mixed vocabularies are permitted using the vocabulary as a prefix to the field name, separated from the field name by the character ‘.’ |
| SearchRequest/OutputFormatURL | A required schema name for the search response payload. The single value matches values from the ResourceList/Resource/OutputFormats/OutputFormat/FormatURI. |
| SearchRequest/OutputObject | A required search response payload name. The single value matches values from the ResourceList/Resource/OutputFormats/OutputFormat/OutputType. It may be one of the values Delimited, Encoded-Delimited, the Well-Known Object Types or a system defined value. |
| SearchRequest/OutputObject@accept | An optional MIME media type or subtype as defined in the relevant IETF Request for Comments document [rfc2045], [rfc2046]. Some examples are */*, the default; image/*, application/msword and video/quicktime. |
| SearchRequest/Count | An optional Boolean value. If true, the response returns the number of results matching the query. The default value is false. |

Table 9.2 The SearchRequestType Elements and Attributes

| ELEMENT OR ATTRIBUTE | DESCRIPTION |
|--------------------------|--|
| SearchRequest/Limit | An optional value to limit the number of return records. The value -1 indicates no limit, other than that set by the system. It is the default value. The value of '0' returns no results. This may be used in conjunction with the Count element set to true. In this case, the system will return only the count of records and not the records themselves. |
| SearchRequest/Offset | An optional value that indicates a numeric offset into the result set. The records are marked using ordinal numbers, the first record has the value of 0 and the second has the value 1 and so on. The default value is 0. |
| SearchRequest/Select | An optional value that provides a token list of field names, separated by whitespace, identifying which elements are to be returned in the response. This element is only valid when the ObjectType value is Delimited or Encoded-Delimited . The default value is empty, meaning return all elements. |
| SearchRequest@vocabulary | An optional value that provides the vocabulary of the SearchRequest, for both the Query and Select elements. It matches values from the ResourceList/Resource/VocabularyNames/VocabularyName element. If this attribute is not present the value defaults to the value of WellKnown. Mixed vocabularies are supported by using the vocabulary as a prefix followed by the “:” character, followed by the field name. Mixed vocabularies are permitted using qualified Vocabulary/Field/Name values where the Field/Name is prepended with the Vocabulary/Name and separated from the Field/Name by the ‘:’ character. |
| SearchRequest@taskId | An optional unique task identifier for this request. |
| SearchRequest@version | An optional version of the SearchRequest schema that was used in this request. |

R0080 A Provider **MUST** support Limit.

R0081 A Provider **MUST** support Count.

R0082 A Provider **MAY** support Offset.

R0083 A Requestor **MAY** support Limit.

R0084 A Requestor **MAY** support Count.

R0085 A Requestor **MAY** support Offset.

- R0086 If a SearchRequest@vocabulary value is not specified, and prefixes are not used in the query fields, the Provider **MUST** use the standard vocabulary to evaluate the query.
- R0087 A Provider **MUST** support WellKnown names.
- R0088 If a Requestor wishes to mix vocabularies, it **MUST** append the Vocabulary name + ":" as a prefix to each field name in the Query or Select.
- R0090 An MLS Provider **MUST** provide the Delimited ObjectType.
- R0091 A Provider that does not provide MLS data **MAY** support the Delimited ObjectType.
- R0092 A Provider **MUST** support Select for the Delimited or Encoded-Delimited ObjectTypes only.
- R0093 A Requestor **MUST** support WellKnown names.
- R0088 SearchRequest/Select and SearchRequest/Query statements permit forming the statement using Field/Name values from multiple Vocabularies within a Resource. Mixed vocabulary Field/Name values **MUST** be formed by taking the Vocabulary/Name value, append the ':' character and then appending the Field/Name value.
- R0094 A Requestor **MAY** use multiple vocabularies in a Query or Select.
- R0095 A Provider, who supports multiple vocabularies for a Resource, **MUST** support multiple vocabularies in a Query or Select.
- R0096 A Requestor **MAY** support Select.
- R0097 A Requestor **MAY** support Delimited [OutputObject files](#).
- R0098 A Provider **MUST** encode Delimited [OutputObject files](#) using UTF-8.
- R0099 A Provider **MUST** escape special characters in Delimited [OutputObject files](#) using the backslash.
- R0100 A Provider **MUST** decode all values in Delimited [OutputObject files](#).
- R0101 A Provider **MUST** use only a comma, pipe, or tab as a delimiter in Delimited [OutputObject files](#).
- R0102 A Provider **MUST** enclose all string values in double quotes as string literals in Delimited [OutputObject files](#).
- R0149 A Requestor **MAY** support Encoded-Delimited [OutputObject files](#).

- R0150 A Requestor **MUST** encode Encoded-Delimited OutputObject files using UTF-8.
- R0156 A Provider **MUST** escape special characters in Encoded-Delimited OutputObject files using the backslash.
- R0151 A Provider **MUST** encode all lookup values in Encoded-Delimited OutputObject files using values from the Resource Metadata.
- R0152 A Provider **MUST** encode all multi-lookup values in Encoded-Delimited OutputObject files using values from the Resource Metadata.
- R0153 A Provider **MUST** encode all Boolean values in Encoded-Delimited OutputObject files. The value '1' is true and the value '0' is false
- R0154 A Provider **MUST** use **ONLY** a comma, pipe, or tab as a delimiter in Encoded-Delimited OutputObject files.
- R0155 A Provider **MUST** enclose all string values in double quotes as string literals in Encoded-Delimited OutputObject files.
- R0157 An MLS Provider **MUST** provide the Encoded-Delimited ObjectType.
- R0158 A Provider that does not provide MLS data **MAY** support the Encoded-Delimited ObjectType.
- R0103 A Requestor **MUST** use whitespace as a delimiter when using the Select parameter.
- R0104 A Provider that supports a Resource describing MLS Listing data **MUST** support all of the Listing.xsd, Property.xsd and ListingProperty.xsd standard payloads.
- R0105 A Provider that supports a Resource describing Agent data **MUST** support the standard Agent.xsd payload.
- R0106 A Provider that supports a Resource describing Office data **MUST** support the standard Office.xsd payload.
- R0107 A Provider **MAY** support any of the additional standard payloads.
- R0108 A Provider **MAY** support custom or local payloads defined as either Schema or ObjectTypes.
- R0109 A Provider who does not support unlimited queries using Limit=-1 **MUST** support Offset.

Figure 9.1 SearchRequest complexType Schema Fragment

```
<xsd:complexType name="SearchRequestType">
  <xsd:sequence>
```

```

<xsd:element name="Resource" type="xsd:NMTOKEN" />
<xsd:element name="Query">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="queryType"
          type="xsd:NMTOKEN" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Count"
  type="xsd:boolean"
  minOccurs="0" />
<xsd:choice>
  <xsd:element name="OutputFormatURL"
    type="xsd:anyURI" />
  <xsd:element name="OutputObject">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:NMTOKEN">
          <xsd:attribute name="accept"
            type="xsd:string" />
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
</xsd:choice>
<xsd:element name="Limit"
  type="xsd:integer"
  minOccurs="0" />
<xsd:element name="Offset"
  type="xsd:nonNegativeInteger"
  minOccurs="0" />
<xsd:element name="Select" minOccurs="0">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="vocabulary"
          type="rets:interpretation" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
<xsd:attribute name="vocabulary" type="xsd:NMTOKEN" />
</xsd:complexType>

```

The example SearchTransaction action below queries the Residential resource using RQL, returning at most twenty records using the custom payload MyProperty defined in MyProperty.xsd.

Figure 9.2 Example SearchTransaction Request

```
<SearchRequest taskID="23312" version="1">
  <Resource>Residential</Resource>
  <Query queryType="RQL">
    (ListingID='05-2434')
  </Query>
  <OutputFormatURL>
    http://myserver.com/MyProperty.xsd
  </OutputFormatURL>
  <Limit>20</Limit>
</SearchRequest>
```

The example SearchTransaction action below queries the Residential resource using RQL with the default vocabulary, requesting the Delimited Residential payload. The Select@vocabulary attribute is absent, defaulting to "WellKnown". Thus, in this example, the field names returned in the Delimited Payload will be the WellKnown names.

Figure 9.3 Example Search Request - Delimited

```
<SearchRequest vocabulary="WellKnown">
  <Resource>Residential</Resource >
  <Query queryType="RQL">
    (PostalCode='44144')
  </Query>
  <OutputObject>Delimited</OutputObject>
  <Limit>1</Limit>
  <Select>ListingId Remarks</Select>
</SearchRequest>
```

This is what the Delimited payload would look like:

Figure 9.4 Example Search Response - Delimited Payload with Escape Encoding

```
ListingId,Remarks
"485-303", "\"Like New\" home!\n\r Sits on 8 acres."
"787-99", "Great value.\t \'Must sell\'. Motivated."
```

The example SearchTransaction below queries the Residential resource using RQL and requests a Delimited Residential payload. It requests only three fields, MLS_NUM, PRICE and CITY, and specifies "Local" as the Select@vocabulary. This means that the resulting Delimited payload will have Local field names.

Figure 9.5 Example Search Request - Delimited , Select Output

```
<SearchRequest vocabulary="Local">
  <Resource>Residential</Resource>
  <Query queryType="RQL">
    (MLS_NUM='05-12345')
  </Query>
  <OutputObject>Delimited</OutputObject>
  <Limit>100</Limit>
```

```

    <Offset>3</Offset>
    <Selected>MLS_NUM PRICE CITY</Selected>
</SearchRequest>

```

This is what the Delimited payload would look like:

Figure 9.6 Example Search Response - Delimited payload with Local Names

```

MLS_NUM, PRICE, CITY
"23356-34", 456000.00, "Rockway"
"7654-1", 500000.00, "Atlanta"
"78765556", 405000.00, "Peoria"

```

Searching for Objects in RETS2 requires two search requests.

The first request queries a Resource, requesting an ObjectReferenceList to find the Objects available for those records that match the request query.

The example SearchTransaction below queries the Residential Resource and requests an ObjectReferenceList payload to find references to all Objects types that match the query.

Figure 9.7 Example Search Request - ObjectReferenceList

```

<SearchRequest>
  <Resource>Residential</Resource>
  <Query queryType="RQL">
    (PostalCode='44131')
  </Query>
  <OutputFormatURL>
    http://rets.org/ObjectReferenceList.xsd
  </OutputFormatURL>
</SearchRequest>

```

The response returned is an ObjectReferenceList that returns Thumbnails and Photo elements that match the query. The Object Resource for this data is called PropertyImages. [Note that this example shows that the referenceId may not match the document key identifier. Providers are free to implement Object storage solutions appropriate to their system.](#)

Figure 9.8 Example Search Response - ObjectReferenceList

```

<object:ObjectReferenceList
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:object="http://rets.org/ns/ObjectReferenceList"
  xsi:schemaLocation="http://rets.org/ns/ObjectReferenceList
    ObjectReferenceList.xsd">
  <object:ObjectResource>
    <object:ResourceName>PropertyImages</object:ResourceName>
    <object:Objects>
      <object:Object referenceId="123454">
        <object:ObjectTypes type="Thumbnail">
          <object:ObjectType>
            <object:Offset>0</object:Offset>

```

```

<object:ByteSize>64000</object:ByteSize>
<object:ObjectId>124</object:ObjectId>
<object:URL>
  http://myserver.com/pics/124.jpg
</object:URL>
<object:Description>
  House Front View
</object:Description>
<object:ContentType>
  image/jpeg
</object:ContentType>
<object:Copyright/>
<object:Timestamp>
  2006-03-03T13:03:04-05:00
</object:Timestamp>
</object:ObjectType>
</object:ObjectTypes>
<object:ObjectTypes type="Photo">
  <object:ObjectType>
    <object:Offset>0</object:Offset>
    <object:ByteSize>128000</object:ByteSize>
    <object:ObjectId>474</object:ObjectId>
    <object:URL>
      http://myserver.com/pics/474.jpg
    </object:URL>
    <object:Description>
      House Front View
    </object:Description>
    <object:ContentType>
      image/jpeg
    </object:ContentType>
    <object:Copyright/>
    <object:Timestamp>
      2006-03-03T13:03:04-05:00
    </object:Timestamp>
    </object:ObjectType>
  </object:ObjectTypes>
</object:Object>
<object:Object referenceId="323215">
  <object:ObjectTypes type="Thumbnail">
    <object:ObjectType>
      <object:Offset>1</object:Offset>
      <object:ByteSize>26000</object:ByteSize>
      <object:ObjectId>225</object:ObjectId>
      <object:URL>
        http://myserver.com/pics/225.gif
      </object:URL>
      <object:Description>
        Whole House
      </object:Description>
      <object:ContentType>
        image/gif

```

```

        </object:ContentType>
        <object:Copyright/>
        <object:Timestamp>
            2006-03-03T13:03:04-05:00
        </object:Timestamp>
    </object:ObjectType>
</object:ObjectTypes>
<object:ObjectTypes type="Photo">
    <object:ObjectType>
        <object:Offset>0</object:Offset>
        <object:ByteSize>52000</object:ByteSize>
        <object:ObjectId>009998</object:ObjectId>
        <object:URL>
            http://myserver.com/pics/009998.jpg
        </object:URL>
        <object:Description>
            Whole House
        </object:Description>
        <object:ContentType>
            image/jpeg
        </object:ContentType>
        <object:Copyright/>
        <object:Timestamp>
            2006-03-03T13:03:04-05:00
        </object:Timestamp>
    </object:ObjectType>
</object:ObjectTypes>
</object:Object>
</object:Objects>
</object:ObjectResource>
<object:ObjectResource>
</object:ObjectReferenceList>

```

The second SearchRequest uses the information from the ObjectReferenceList to query the Object Resource. Alternately, if the ObjectReferenceList provides a URL for the Object, the Requestor may choose to use a GET to retrieve the Object instead of another Search request.

The example SearchTransaction below uses information provided by the ObjectReferenceList to query the PropertyImages Resource and requests a payload of all Objects that are Thumbnails. It uses the referenceId that corresponds to the key field ListingId. The request specifies the no-limit to retrieve all Thumbnails. An accept attribute value of */* requests all mime types.

Figure 9.9 Example Search Request - All Thumbnail Objects

```

<SearchRequest>
  <Resource>PropertyImages</Resource >
  <Query queryType="RQL">
    (ListingID='123454')
  </Query>
  <OutputObject accept="*/*">
    Thumbnail
  </OutputObject>

```

```
<Limit>-1</Limit>
</SearchRequest>
```

A Requestor may want to perform a Search that returns a count of matching records without the record documents. The Count element will be returned as a part of the Manifest, but the Manifest Content will not contain a Search payload. To form this type of search, a Requestor would enter a query, set the Count element to true and set the Limit to 0. Even though an OutputFormatURL is specified, a Limit of 0 will prevent a payload from being returned in the SearchResponse Manifest

Figure 9.10 Example SearchTransaction Request, Count Only

```
<SearchRequest id="23312" version="1">
  <Resource>Residential</Resource>
  <Query queryType="RQL">
    (PostalCode='44143')
  </Query>
  <OutputFormatURL>
    http://myserver.com/MyProperty.xsd
  </OutputFormatURL>
  <Count>>true</Count>
  <Limit>0</Limit>
</SearchRequest>
```

A Requestor may want to perform a Search that returns the Message of the Day, if the Provider supports a MessageOfDay Resource. The following Search queries the MessageOfDay for everything greater than or equal to Apr 4, 2006:

Figure 9.11 Example Search Request - Message of the Day

```
<SearchRequest taskID="23312" version="1">
  <Resource>MessageOfDay</Resource>
  <Query queryType="RQL" vocabulary="Local">
    (Timestamp >='2006-04-04T00:00:00Z')
  </Query>
  <OutputFormatURL>
    http://www.rets.org/MessageOfDay.xsd
  </OutputFormatURL>
  <Limit>-1</Limit>
</SearchRequest>
```

9.3 Supported Query Types

RETS2 supports a single query language, RQL. Additional queries languages may be supported in the future or may be supported through non-standard means between [Requestors and Providers endpoints](#).

The query language is defined in the document RQL – The RETS Query Language. This is the actual search criteria passed to the [Provider server](#).

R0110 A Provider **MUST** support the **RQL** query language.

R0111 A Requestor **MUST** support the **RQL** query language.

R0112 A Provider **MAY** support additional query languages not defined in this standard.

R0113 A Requestor **MAY** support additional query languages not defined in this standard.

Providers must treat query values as case-insensitive. Requestors and Providers must treat query field names as case-sensitive.

R0114 Providers **MUST** treat query values as case-insensitive.

R0115 Providers and Requestors **MUST** treat query field names as case-sensitive.

10 Error Handling

- R0116** A Provider **MUST** support the Search and Metadata errors following **SOAP** Faults **in conformity with conformant to** the **SOAP** 1.2 specification.
- R0117** A Provider **MUST** support the Update errors either following **SOAP** Faults **in conformity with conformant to** the **SOAP** 1.2 specification or as UpdateResponse faults.
- R0118** A Provider supporting the Search Action **MUST** support SearchFaults.
- R0119** A Provider supporting the Update Action **MUST** support UpdateFaults.
- R0120** A Provider supporting the Metadata Actions **MUST** support MetadataFaults.
- R0121** Wherever possible, a Provider **MUST** use the **SOAP** Fault Codes listed below.
- R0122** A Provider **MAY** use the Reasons listed below as **SOAP** Fault Reasons.
- R0136** A Provider **MAY** create custom **SOAP** Fault Codes to convey errors that do not map to one of the Supported Faults for that Action.

10.1 Supported Faults

Values from the **SOAP** Fault Code in the tables below **MUST** be used as values for the *Code* element. **Items in the Code element MUST be used as the SOAP Fault Code.** Providers are free to customize the **SOAP** Fault Reason on an implementation basis or to use the corresponding Reason from the table. **Providers are free to extend the SOAP Fault Codes to meet specific system needs, but SHOULD attempt to use the RETS2 standard fault codes wherever possible.**

Figure 10.1 Example of a SOAP Fault

```
<env:Fault>
  <env:Code>
    <env:Value>InvalidResource</env:Value>
  </env:Code>
  <env:Reason>
    <env:Text xml:lang="en">
      The Resource "RES" is not
      supported by this Provider.
    </env:Text>
  </env:Reason>
</env:Fault>
```

10.1.1 Authentication and Authorization Faults

Since each RETS2 request contains authentication, all RETS2 requests may return Authorization and Authentication errors. Authentication errors are normally standard system errors. Examples include, but are not limited to, invalid password and invalid user id. These errors are not predefined by RETS2. A Provider should use [implementation specific](#) [his](#) system and framework [tools](#) to create SOAP faults for [his implementation's](#) Authentication errors.

Authorization errors occur when a user can be properly authenticated, but is not permitted to perform certain actions due to business rules. General RETS2 Authorization [Fault](#) Codes and Reasons are defined below. A Provider must use the Fault Code but may override the Reason with meaningful, implementation-specific information.

Table 10.1 SOAP Authorization Fault Codes

| SOAP FAULT CODE | REASON | NOTES |
|------------------------|---|---|
| ForbiddenAction | The Requesting user does not have permission to use this RETS action. | The reason should specify the action that this fault occurred for: Search, Meta-data, Update. |
| Billing | The user cannot perform this action because of an account billing type fault, such as an unpaid bill. | |
| AccountExpired | The Requestor authorization has expired for this action. | An example might be attempting an action using an expired account or expired account feature. |
| ContactCustomerService | The Requestor should contact customer service. | The fault reason should include some information to permit customer service to resolve the fault. |

10.1.2 SearchFaults

Table 10.2 Soap Search Fault Codes

| SOAP FAULT CODE | REASON | NOTES |
|-----------------|--|-------|
| InvalidResource | The resource provided in the request is not valid. | |

Table 10.2 Soap Search Fault Codes

| SOAP FAULT CODE | REASON | NOTES |
|---------------------------|---|--|
| ResourceUnavailable | The resource provided in the request is currently unavailable. | |
| InvalidType | The type provided in the request is not supported by the Provider. | |
| InvalidSyntax | The provided in the request could not be understood. | |
| UnknownField | The provided in the request has one or more unknown fields for this Provider. | |
| TooComplex | The provided in the request is too complex to be processed. | An example may be that the contains too many nesting levels or too many values for a lookup field. |
| Unauthorized | The provided could not be executed because it refers to one or more fields that the Requestor does not have permission to access. | Providers may choose to omit the unauthorized fields from the fault reason. |
| RequestFormatUnavailable | The Input Format requested is unavailable. | Providers may choose to provide more detailed information about the reason. Causes could be schema version violation, time window violation or other causes. |
| ResponseFormatUnavailable | The OutputFormat requested is unavailable. | Providers may choose to provide more detailed information about the reason. Causes could be schema version violation, time window violation or other causes. |
| InvalidRequestFormat | The Input Format requested is invalid. | Providers may choose to provide more detailed information about the reason. Causes could be invalid schema version, MIME type is invalid for the request resource, or other reasons. |

Table 10.2 Soap Search Fault Codes

| SOAP FAULT CODE | REASON | NOTES |
|-----------------------|---|--|
| InvalidResponseFormat | The Output Format requested is invalid. | Providers may choose to provide more detailed information about the reason. Causes could be invalid schema version, MIME type is invalid for the request resource, or other reasons. |
| Timeout | The request timed out while executing. | |
| Forbidden | The request is forbidden. | |
| RequestTooLarge | The request exceeded a system limit. | |
| NoItemsFound | No matching items were found. | |

10.1.3 UpdateFaults**Table 10.3 SOAP Update Fault Codes**

| SOAP FAULT CODE | REASON | NOTES |
|------------------------|---|-------|
| ForbiddenAction | The request is forbidden. | |
| InvalidResource | The Resource requested is not supported by this system. | |
| InvalidUpdateFormat | | |
| InvalidField | | |
| UpdateTypeNotSupported | | |
| Timeout | The request timed out while executing. | |
| NotAuthenticated | | |
| RequestTooLarge | The request exceeded a system limit. | |
| FailedToSaveRecord | The action failed to save the record. | |

10.1.4 MetadataFaults

Table 10.4 SOAP Metadata Fault Codes

| SOAP FAULT CODE | REASON | NOTES |
|--------------------------|---|-------|
| InvalidMetadataFormatURL | The format requested is not supported by this system. | |
| InvalidResource | The Resource requested is not supported by this system. | |
| InvalidMetadataAction | The Action does not match known services for this system. | |
| InvalidParameter | An invalid parameter is present in the request. | |
| InvalidName | The Name supplied in the request is not valid for the requested Format. | |
| InvalidTimestamp | The Metadata did not have a timestamp equal or newer than the timestamp requested. | |
| ParameterValidationError | A data type mismatch between a parameter and the wsdl datatype for that parameter has occurred. | |

11 Message Transports

11.1 SOAP over HTTP – Request and Response

11.1.1 Data Compression in RETS Transactions over HTTP

Requestors Clients and Providers servers may choose to support data compression in data returned from the Provider server. To indicate its willingness to accept compressed data, a Requestor client should include an *Accept-Encoding* header in its request. If the Provider servers supports one of the compression methods accepted by the Requestor client, it can include a *Content-Encoding* header in its response indicating the compression method it has chose. Requestors Clients and Providers servers choosing to implement compression should at least support GZip compression. This method is implemented by freely-available source code in a number of languages, as well as in several proprietary software development environments. A second freely-available alternative is BZIP. Requestors Clients and Providers servers are free to choose other encoding methods as well.

R0123 A Provider **MAY** support compression.

R0124 A Requestor **MAY** support compression.

R0125 If compression is supported, a Provider ~~an Endpoint~~ **SHOULD** support at least GZIP.

11.1.2 Secure HTTP

The need for secure HTTP transactions cannot be met by authentication schemes. For those needs, HTTP-over-TLS (commonly known as HTTPS) is a more appropriate protocol. A Provider server may support only HTTP-over-SSL.

12 Security

Security for RETS2 is defined in the document RETS2 Security Specification.

13 Service Compliance

Compliance to the Specification is defined in the document RETS2 Service Compliance.

14 RQL Language

The normative reference for the default language is the document

- RQL BNF Located at:

http://www.rets.org/rets2/rql/tags/rql_1_0-bnf.txt

RQL Language

http://www.rets.org/rets2/rql/tags/rql_1_0.html

The documents are mirrored at

<https://code.crt.realtors.org/svn/librets/rql/tags/draft-5/rql.html>

<https://code.crt.realtors.org/svn/librets/rql/tags/draft-5/rql-bnf.txt>

15 Acknowledgments

The creation of this specification would not have been possible without the sponsorship and coordination of efforts provided by the National Association of REALTORS®. In particular, the support and sponsorship of Dale Stinton, CEO and Myron Adams, Vice President, ITS, was essential to the creation of this standard.

This document has benefited greatly from the comments of all those participating in the National Association of REALTORS® Real Estate Transaction Standard Work Group.

Important contributions were made by (in alphabetical order by last name): Gina Accawi (Formerly of First American MLS Solutions formerly MarketLinx®), Dave Dribin (Center for Realtor Technology), Mila Kaliman (First American Title), Matthew McGuire (First American MLS Solutions, formerly MarketLinx®), Steve Verba (Oracle Corporation, formerly of Avantia, Inc.), Libor Viktorin (First American MLS Solutions, formerly MarketLinx®), Wantao Zhou (First American MLS Solutions, formerly Interealty).

The RETS2 documents were reviewed by many people. Their comments have made this a better document. Reviewers were (in alphabetic order by last name): Todd Andrews (REFormsNet), Laurie Chipman (eNeighborhoods Inc. formerly WyldFyre), Sergio Del Rio (Templates for Business Inc.), Keith Garner (Center for Realtor Technology), , Karen Lee (First American Title), Shawn Meissner (Rapattoni Corporation), Ramesh Radhakrishnan (eNeighborhoods, Inc.), Chin Shu (eNeighborhoods, Inc.), Mark Scheel (Siegent, Inc.), Mark Sleeman (dbSage iSolutions), Mark Suchy (First American Title), Bill Tonissen, REFormsNet, Joshua Vosper (Rapattoni Corporation), Dan Woolley (eNeighborhoods, Inc.), and Mourad Zerroug (REBT).

16 Bibliography

[ANSI]

ANSI US-ASCII Coded Character Set – 7 Bit American Standard Code for Information Interchange. Standard ANSI X3.4-1986, ANSI, 1986.

[COMPLIANCE]

Ferris, Chris; Karmarkar, Anish; and Kanyang, Kevin Liu; *Attachment Profile Version 1.0* [online] http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html#conformance_requirements

[DC]

Hillman, Diane; *Using Dublin Core – The Elements (2005-11-07)*, [online] <http://dublincore.org/documents/usageguide/elements.shtml#identifier>

[ITF]

[online]

<http://www.ietf.org/internet-drafts/draft-ietf-ltru-registry-14.txt>

[ISG]

Center for REALTOR® Technology; *National Association of REALTORS® Information Security Guidelines Version 1.0*, [online]

[http://www.realtor.org/realtororg.nsf/files/NARInternetSecurityGuide.pdf/\\$FILE/NARInternetSecurityGuide.pdf](http://www.realtor.org/realtororg.nsf/files/NARInternetSecurityGuide.pdf/$FILE/NARInternetSecurityGuide.pdf)

[LTRU1]

Phillips, A; Davis, M; (Ed.) *Tags for Identifying Languages - draft-ietf-ltru-registry-14* <http://www.ietf.org/internet-drafts/draft-ietf-ltru-registry-14.txt>

[MTOM1]

Gudgin, Martin; Mendelsohn, Noah; Nottingham, Mark; Ruellan, Hervé (Ed.) *SOAP Message Transmission Optimization Mechanism* [online]

<http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/>

[MTOM2]

Gudgin, Martin; Mendelsohn, Noah; Nottingham, Mark; Ruellan, Hervé (Ed.) *SOAP Message Transmission Optimization Mechanism Errata* [online]

<http://www.w3.org/2005/01/soap12-mtom-errata.html>

[NAMESPACES]

Bray, Tim; Hollander, Dave; Layman, Andrew; *Namespaces in XML* [online] <http://www.w3.org/TR/REC-xml-names/>

[RFC2045]

Freed, N; Borenstein, N; *Request for Comments: 2045 – Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, [online]

<http://www.apps.ietf.org/rfc/rfc2045.html>

<http://www.ietf.org/rfc/rfc2045.txt>

[RFC2046]

Freed, N; Borenstein, N; *Request for Comments: 2046 – Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, [online]

<http://www.apps.ietf.org/rfc/rfc2046.html>

<http://www.ietf.org/rfc/rfc2046.txt>

[RFC2119]

Bradner, S; *Request for Comments: 2119 - Key words for use in RFCs to Indicate Requirement Levels*, [online]

<http://www.apps.ietf.org/rfc/rfc2119.html>

<http://www.ietf.org/rfc/rfc2119.txt>

[RFC2234]

Crocker, D; (Ed.); *Request for Comments: 2234 – Augmented BNF for Syntax Specifications: ABNF*, [online]

<http://www.apps.ietf.org/rfc/rfc2234.html>

<http://www.ietf.org/rfc/rfc2234.txt>

[RFC2822]

Resnick, P; (Ed.); *Request for Comments: 2822 - Internet Message Format*, [online]

<http://www.apps.ietf.org/rfc/rfc2822.html>

<http://www.ietf.org/rfc/rfc2822.txt>

[RFC3066]

Alvestrand, H; *Tags for the Identification of Languages*, [online]

<http://www.apps.ietf.org/rfc/rfc3066.html>

<http://www.ietf.org/rfc/rfc3066.txt>

[RFC4234]

Crocker, D; Overell, P (Ed.); *Request for Comments: 4234 – Augmented BNF for Syntax Specification, DRAFT*, [online]

<http://www.apps.ietf.org/rfc/rfc4234.html>

<http://www.ietf.org/rfc/rfc4234.txt>

[SEC1]

Nadalin, Anthony; Kaler, Chris; Hallam-Baker, Phillip; Monzillo, Ronald (Ed.) *Web Services Security: SOAP Message Security 1.0* [online]

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

[SEC2]

Nadalin, Anthony; Griffin, Phil; Kaler, Chris; Hallam-Baker, Phillip; Monzillo, Ronald (Ed.) *Web Services Security: UsernameToken Profile 1.0* [online]

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>

[SEC3]

Hallam-Baker, Phillip; Kaler, Chris; Monzillo, Ronald; Nadalin, Anthony (Ed.) *Web Services Security: X509 Certificate Token Profile* [online]

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf>

[SEC4]

Hallam-Baker, Phillip; Kaler, Chris; Monzillo, Ronald; Nadalin, Anthony (Ed.) *Web Services Security: SAML Token Profile* [online]

<http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf>

[SOAP1]

Mitra, Nilo (Ed.) *SOAP Version 1.2 Part 0: Primer* [online]

<http://www.w3.org/TR/soap12-part0/>

[SOAP2]

Gudgin, Martin; Hadley, Marc; Mendelsohn, Noah; Moreau, Jean-Jaques, Frystyk, Henrik; (Ed.) *SOAP Version 1.2 Part 1: Messaging Framework* [online]

<http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>

[SOAP3]

Gudgin, Martin; Hadley, Marc; Mendelsohn, Noah; Moreau, Jean-Jaques, Frystyk, Henrik; (Ed.) *SOAP Version 1.2 Part 2: Adjuncts* [online]

<http://www.w3.org/TR/2003/REC-soap12-part2-20030624/>

[WSDL1]

Christensen, Erik; Curbera, Francisco; Meredith, Greg; Weerawarana, Sanjiva *Web Services Description Language 1.1* [online]

<http://www.w3.org/TR/wsdl>

[XML1]

Bray, Tim; Paoli, Jean; Sperberg-McQueen, CM; Maler, Eve; Yergeau, François (Ed.) *Extensible Markup Language (XML) 1.0 (Third Edition)* [online]

<http://www.w3.org/TR/2004/REC-xml-20040204/>

[XML2]

Bray, Tim; Paoli, Jean; Sperberg-McQueen, C. M; Maler, Eve; Yergeau, François (Ed.) *XML 1.0 Third Edition Specification Errata* [online]

<http://www.w3.org/XML/xml-V10-3e-errata>

[XML3]

Fallside, David C; Walmsley Priscilla (Ed.) *XML Schema Part 0: Primer Second Edition* [online]

<http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/>

[XML4]

Thompson, Henry S; Beech, David; Maloney, Murray; Mendelsohn, Noah (Ed.) *XML Schema Part 1: Structures Second Edition* [online]

<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>

[XML5]

Biron, Paul V; Malhotra, Ashok *XML Schema Part 2: Datatypes Second Edition* [online]

<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>

[XML6]

Thompson, Henry S; Beech, David; Maloney, Murray; Mendelsohn, Noah, Biron, Paul V; Malhotra, Ashok *XML Schema 1.0 Second Edition Errata* [online]

<http://www.w3.org/2004/03/xmlschema-errata>

17 Footnotes

[1]

Hass, Hugo; Web Services Activity Statement [online]

<http://www.w3.org/2002/ws/Activity>

[2]

Mitra, Nilo (Ed.) *SOAP Version 1.2 Part 0: Primer* [online]

<http://www.w3.org/TR/soap12-part0/>

18 Glossary

Document/literal

MAY

MUST

MUST NOT

Namespace

Normative

Open Standards

OPTIONAL

Qualified Name (QName)

RECOMMENDED

Remote Procedure Call

REQUIRED

Semantics

Service Provider

Service Requestor

SHALL

SHALL NOT

SHOULD

SHOULD NOT

SOAP

19 Appendix A – Supporting Documents, Schema and WSDL Files

The following list of files represents the ancilliary payloads and WSDL file for the RETS2 specification.

rets.wsdl
Activity.xsd
Agency.xsd
Contact.xsd
Course.xsd
DataDictionary.xsd
Documents.xsd
ical.xsd
Listing.xsd
ListingHistory.xsd
ListingProperty.xsd
LookupList.xsd
MemeberFinancial.xsd
MemberRecord.xsd
MemberTransmittal.xsd
NRDSCommons.xsd
Offer.xsd
OfficeRoster.xsd
Offices.xsd
OfficeTransmittal.xsd
Participants.xsd
Property.xsd
PublicRecord.xsd
Referral.xsd
ResourceList.xsd
RETSCCommons.xsd
ServiceOrder.xsd
Transaction.xsd
TransactionList.xsd
Vocabulary.xsd

20 Appendix B - Requirements Summary

- R0001 All rets2 Messages must use the soap Document/Literal Model. Page 3-3
- R0002 Services, Messages and Payloads must conform to the specifications defined in all of soap 1.2, wsdl 1.1, xml 1.0 and xml Schema. Page 4-1
- R0003 Providers may use the soap Message Transmission Optimization Method when sending payloads Page 4-1
- R0004 Requestors must accept the soap Message Transmission Optimization Method when receiving payloads Page 4-1
- R0005 Messages must conform to the soap Message Security specification. Page 4-2
- R0006 Messages should use Username Token Profile Page 4-2
- E0001 Messages may support x.509 Token Profile or saml Token Profile or additional profiles for authentication. Page 4-2
- R0007 Messages that use Username Token authentication must conform to the Web Services Security: Username Token Profile Document V1.0. Page 4-2
- R0126 Messages that use Username Token authentication should use Password_Digest instead of Password_PlainText. Page 4-2
- R0127 A Provider that uses Username Token Profile authentication and implement Password_Digest must reject any Username token that does not use both Nonce and Created elements. Page 4-3
- E0004 It is recommended that a Provider that uses Username Token Profile authentication and implement Password_Digest reject any token that is older than a certain age. As a guideline, an age of five minutes for a valid Created element can be used to dete Page 4-3
- E0005 It is recommended that a Provider that uses Username Token Profile authentication and implement Password_Digest cache Nonce values for a period at least as long as the age of the Created element above and that any token that is reuses a nonce be rej Page 4-3
- E0002 Messages that use x.509 Token authentication must conform to the Web Services Security: x.509 Certificate Token Profile Document 200401. Page 4-3
- E0003 Messages that use xml Token authentication must conform to the Web Services Security: saml Token Profile Document 200412. Page 4-3
- R0017 A Request Format must conform to the xml 1.0 Specification and the xml Schema Structures and Datatypes specifications. Page 6-1
- R0018 An xml Response Document must conform to the xml 1.0 Specification and the xml Schema Structures and Datatypes specifications. Page 6-1
- E0006 A Provider must have a versioned target namespace uri and a versionTimestamp for an xml OutputFormat schema. Page 6-1
- E0007 A Provider should use extension to create new versions of existing document schema. Page 6-2
- R0019 A Provider must change both the uri and the timestamp for an xml OutputFormat schema when a new version is not backward compatible. Page 6-2

- R0020 A Provider must change ONLY the timestamp for an xml OutputFormat schema when a new version is backward compatible. Page 6-2
- R0021 An xml document that contains elements or attributes in place of the any or other schema elements and attributes should be processed by ignoring the unknown element or attribute. Page 6-2
- R0022 A Provider may issue a fault when a document or request does not validate against the schema for that document. Page 6-2
- R0146 A Provider must return the number of records for the Search payload in the Manifest Count element when the SearchRequest Count is set to "true". Page 6-4
- R0023 Where a Resource name corresponds to a rets well-known name, the Provider must use the well-known name. See the "rets 2 Well-Known Names" document for the list. Page 6-8
- R0024 A Provider must provide ResourceList Metadata. Page 6-12
- R0025 A Provider must provide at least one Vocabulary for each Resource. Page 6-12
- R0026 A Resource must provide the WellKnown Vocabulary for that Resource. Page 6-12
- R0027 Wherever a well-known Resource can map a field to a WellKnown name, as defined in the Well-Known names document, it must provide a WellKnown Vocabulary containing that field using the WellKnown name. Page 6-12
- R0028 For any Resource, a Provider may provide only the WellKnown Vocabulary. Page 6-12
- R0029 A Provider that operates as WellKnown names only provider must provide a ResourceList Metadata document. Page 6-12
- R0030 A Provider that operates as a WellKnown names provider is not required to provide any other standard Metadata documents including Vocabulary, DataDictionary or Lookup. Page 6-12
- R0031 A Search Action Resource must provide at least one OutputFormat. Page 6-12
- R0032 A Search Action Resource may provide multiple OutputFormats. Page 6-12
- R0033 A Provider may provide Display elements for an OutputFormat. Page 6-12
- R0034 A Requestor may use a Display element to format payloads for presentation. Page 6-12
- R0035 An Update Action Resource must provide at least one InputFormat. Page 6-12
- R0036 An Update Action Resource may provide multiple InputFormats. Page 6-12
- R0037 A ResourceList document must validate against the ResourceList.xsd schema. Page 6-12
- R0148 A Provider should return a fault when an invalid Field Name is used in a Query by the Requestor. Page 6-13
- R0038 A Vocabulary metadata document must validate against the RETS Vocabulary.xsd schema. Page 6-20
- R0147 A Vocabulary Field Name element must not contain whitespace. Page 6-20
- R0041 If a Vocabulary Field has a Lookup or LookupMulti DataType, that Field must contain a LookupName element. Page 6-20
- R0042 If a Vocabulary Field contains a MetaEntryId element, it must reference a valid MetaEntryId for a DataDictionary Field. Page 6-20
- R0039 A Provider may supply RequiredGroups for a Vocabulary. Page 6-21
- R0044 If RequiredGroups are presented in the Vocabulary document, a Requestor must use one of the RequiredGroups in a query. Page 6-21

- R0040 If RequiredGroups are presented in the Vocabulary document, a Requestor must use all of the Fields within one RequiredGroup in a Query. Page 6-21
- R0137 A Provider may supply DisplayGroups for a Vocabulary. Page 6-21
- R0138 If DisplayGroups are present in the Vocabulary document, a Requestor may use one or more DisplayGroup to organize the display of Field names. Page 6-21
- R0139 If DisplayGroups are present in the Vocabulary document, a Provider should provide a DisplayGroup name that may be presented to a human user. Page 6-21
- R0140 A Requestor may use the DisplayGroup name to present the group to a human user. Page 6-21
- R0141 A Provider must not provide any names in a DisplayGroup that are not present in the Vocabulary. Page 6-21
- R0142 A Provider should provide the DisplayGroup names in presentation order. Page 6-21
- R0143 A Requestor may display the DisplayGroup names in the order that they are listed in the DisplayGroup to a human user. Page 6-21
- R0144 A Provider may support the DataDictionary metadata format. Page 6-24
- R0046 If a Provider supports the DataDictionary metadata format, there must be one and only one DataDictionary for a specific Resource. Page 6-24
- R0047 A DataDictionary Metadata document must validate against the RETS DataDictionary.xsd schema. Page 6-24
- R0048 If a DataDictionary field contains a LookupName, it must reference a valid entry in a LookupList document. Page 6-24
- R0049 If a Field in a Provider's Vocabulary or DataDictionary Metadata contains a LookupName, the RETS Provider must support LookupList Metadata. Page 6-25
- R0050 LookupList documents must validate against the standard LookupList.xsd schema. Page 6-25
- R0051 A Provider may advertise all custom/local Metadata as MetadataFormats in their ResourceList. Page 6-26
- R0052 Custom or local metadata documents must validate against the referenced schema. Page 6-26
- R0128 A Provider may support the WellKnown UserInfo Resource. Page 6-28
- R0129 A Provider who supports the WellKnown UserInfo Resource must advertise a UserInfo Resource in its ResourceList. Page 6-28
- R0130 A Provider who supports the WellKnown UserInfo Resource must advertise the UserInformation.xsd as an OutputFormat for that Resource. Page 6-28
- R0131 A UserInformation response document must validate against the UserInformation.xsd schema. Page 6-28
- R0132 A Provider may support the WellKnown MessageOfDay Resource. Page 6-29
- R0133 A Provider who supports the WellKnown MessageOfDay Resource must advertise a MessageOfDay Resource in the ResourceList. Page 6-30
- R0134 A Provider who supports the WellKnown MessageOfDay Resource must advertise the RETS standard MessageOfDay.xsd as a MetadataFormat for that Resource. Page 6-30
- R0135 A MessageOfDay metadata document must validate against the RETS MessageOfDay.xsd

- schema. Page 6-30
- R0053 A Provider must support at least one Object Resource if they offer Object Retrieval or Update. Page 7-13
- R0054 A Provider who has an Object Resource must support the ObjectReferenceList payload for all Resources that reference that Object Resource. Page 7-13
- R0055 ObjectReferenceList documents must validate against the standard ObjectReferenceList.xsd schema. Page 7-13
- R0056 An Object Resource must support well-known ObjectTypes. Page 7-13
- R0057 An Object Resource MAY support additional ObjectTypes. Page 7-13
- R0058 The referenceld of an Object must correspond to the keyfield attribute value for a Vocabulary. Page 7-13
- R0060 A Provider supporting the Update Action for well-known Resources may supply the standard payload for that Resource as an InputFormat. Page 8-1
- R0061 The UpdateRequest document must conform to the RETS UpdateRequest WSDL element. Page 8-1
- R0062 The UpdateRequest resource attribute must correspond to a ResourceName in the Provider metadata ResourceList. Page 8-1
- R0063 The UpdateRequest format attribute must correspond to an InputFormat for the named Resource. Page 8-1
- R0064 The UpdateRequest must specify a format. Page 8-2
- R0065 The UpdateRequest must specify a resource. Page 8-2
- R0066 The UpdateRequest must contain a Manifest with content. Page 8-2
- R0067 The UpdateRequest must specify a type. Page 8-2
- R0068 The UpdateResponse document must conform to the rets wsdl UpdateResponse element. Page 8-5
- R0069 The value of the UpdateResponse resource attribute must correspond to the value of the related UpdateRequest resource attribute. Page 8-5
- R0070 The UpdateResponse may contain a uid attribute. Page 8-5
- R0071 The UpdateResponse must contain a resource attribute. Page 8-5
- R0072 The UpdateRequest may contain a correlationID attribute. Page 8-5
- R0073 The correlationID attribute value must correspond to an UpdateRequest taskID attribute value.. Page 8-5
- R0074 In the case of errors, the UpdateResponse must contain either Faults or throw a soap Fault. Page 8-5
- R0075 In the case of warnings, the UpdateResponse may contain Faults. Page 8-5
- R0076 An UpdateResponse Fault may contain a Message. Page 8-7
- R0077 An UpdateResponse Fault may contain a FieldList Page 8-7
- R0078 An UpdateResponse Fault may specify a severity attribute. Page 8-7
- R0079 An UpdateResponse Fault must contain a Type. Page 8-7
- R0080 A Provider must support Limit. Page 9-4

- R0081 A Provider must support Count. Page 9-4
- R0082 A Provider may support Offset. Page 9-4
- R0083 A Requestor may support Limit. Page 9-4
- R0084 A Requestor may support Count. Page 9-4
- R0085 A Requestor may support Offset. Page 9-4
- R0086 If a Query@vocabulary is not specified, and prefixes are not used in the query fields, the Provider must use the standard vocabulary to evaluate the query. Page 9-5
- R0087 A Provider must support WellKnown names. Page 9-5
- R0090 An mls Provider must provide the Delimited ObjectType. Page 9-5
- R0091 A Provider that does not provide mls data may support the Delimited ObjectType. Page 9-5
- R0092 A Provider must support Select for delimited ObjectTypes only. Page 9-5
- R0093 A Requestor must support WellKnown names. Page 9-5
- R0088 Mixed vocabulary Field/Name values must be formed by taking the Vocabulary/Name value, append the ':' character and then appending the Field/Name value. Page 9-5
- R0094 A Requestor may use multiple vocabularies in a query. Page 9-5
- R0095 A Provider, who supports multiple vocabularies for a Resource, must support multiple vocabularies in a query. Page 9-5
- R0096 A Requestor may support Select. Page 9-5
- R0097 A Requestor may support Delimited OutputObject files. Page 9-5
- R0098 A Provider must encode Delimited OutputObject files using utf-8. Page 9-5
- R0099 A Provider must escape special characters in Delimited OutputObject files using the backslash. Page 9-5
- R0100 A Provider must decode all values in Delimited OutputObject files. Page 9-5
- R0101 A Provider must use only a comma, pipe, or tab as a delimiter in Delimited OutputObject files. Page 9-5
- R0102 A Provider must enclose all string values in double quotes as string literals in Delimited OutputObject files. Page 9-5
- R0149 A Requestor may support Encoded-Delimited OutputObject files. Page 9-5
- R0150 A Requestor must encode Encoded-Delimited OutputObject files using utf-8. Page 9-6
- R0156 A Provider must escape special characters in Encoded-Delimited OutputObject files using the backslash. Page 9-6
- R0151 A Provider must encode all lookup values in Encoded-Delimited OutputObject files using values from the Resource Metadata. Page 9-6
- R0152 A Provider must encode all multi-lookup values in Encoded-Delimited OutputObject files using values from the Resource Metadata. Page 9-6
- R0153 A Provider must encode all Boolean values in Encoded-Delimited OutputObject files. The value '1' is true and the value '0' is false Page 9-6
- R0154 A Provider must use only a comma, pipe, or tab as a delimiter in Encoded-Delimited OutputObject files. Page 9-6

- R0155A Provider must enclose all string values in double quotes as string literals in Encoded-Delimited OutputObject files. Page 9-6
- R0157An mls Provider must provide the Encoded-Delimited ObjectType. Page 9-6
- R0158A Provider that does not provide mls data may support the Encoded-Delimited ObjectType. Page 9-6
- R0103 A Requestor must use whitespace as a delimiter when using the Select parameter. Page 9-6
- R0104 A Provider that supports a Resource describing mls Listing data must support all of the Listing.xsd, Property.xsd and ListingProperty.xsd standard payloads. Page 9-6
- R0105 A Provider that supports a Resource describing Agent data must support the standard Agent.xsd payload. Page 9-6
- R0106 A Provider that supports a Resource describing Office data must support the standard Office.xsd payload. Page 9-6
- R0107 A Provider may support any of the additional standard payloads. Page 9-6
- R0108 A Provider may support custom or local payloads defined as either Schema or Object-Types. Page 9-6
- R0109 A Provider who does not support unlimited queries using Limit=-1 must support Offset. Page 9-6
- R0110 A Provider must support the rql query language. Page 9-12
- R0111 A Requestor must support the rql query language. Page 9-13
- R0112 A Provider may support additional query languages not defined in this standard. Page 9-13
- R0113 A Requestor may support additional query languages not defined in this standard. Page 9-13
- R0114 Providers must treat query values as case-insensitive. Page 9-13
- R0115 Providers and Requestors must treat query field names as case-sensitive. Page 9-13
- R0116 A Provider must support the Search and Metadata errors following soap Faults in conformity with the soap 1.2 specification. Page 10-1
- R0117 A Provider must support the Update errors either following soap Faults in conformity with the soap 1.2 specification or as UpdateResponse faults. Page 10-1
- R0118 A Provider supporting the Search Action must support SearchFaults. Page 10-1
- R0119 A Provider supporting the Update Action must support UpdateFaults. Page 10-1
- R0120 A Provider supporting the Metadata Actions must support MetadataFaults. Page 10-1
- R0121 Wherever possible, a Provider must use the soap Fault Codes listed below. Page 10-1
- R0122 A Provider may use the Reasons listed below as soap Fault Reasons. Page 10-1
- R0136 A Provider may create custom soap Fault Codes to convey errors that do not map to one of the Supported Faults for that Action. Page 10-1
- R0123 A Provider may support compression. Page 11-1
- R0124 A Requestor may support compression. Page 11-1
- R0125 If compression is supported, a Provider should support at least GZIP. Page 11-1

21 Appendix C – RETS 2 WSDL

```
<wSDL:definitions name="rets" targetNamespace="urn:retsWsdL"
  xmlns:rets="urn:retsWsdL"
  xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
  xmlns:soap12="http://schemas.xmlsoap.org/wSDL/soap12/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:rl="http://rets.org/ns/ResourceList/"
  xmlns:ll="http://rets.org/ns/LookupList/">
<!--
  xmlns:wSDLsoap="http://schemas.xmlsoap.org/wSDL/soap/"
-->
<!-- ***** ->

<wSDL:types>
  <xsd:schema elementFormDefault="qualified"
    targetNamespace="urn:retsWsdL"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <!--
      <xsd:import
        namespace="http://rets.org/ns/LookupList/"
        schemaLocation="../xsd/LookupList.xsd"/>
      <xsd:import
        namespace="http://rets.org/ns/ResourceList/"
        schemaLocation="../xsd/ResourceList.xsd"/>
    -->

    <!-- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ->
    <xsd:complexType name="ResourceList">
      <xsd:sequence>
        <xsd:element
          name="Resource"
          maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element
                name="ResourceName"
                type="xsd:NMTOKEN" />
              <xsd:element
                name="InformationURL"
                type="xsd:anyURI"
                minOccurs="0" />
              <xsd:element
                name="VocabularyNames"
                minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element
                      name="VocabularyName"
                      type="xsd:NMTOKEN"
                      maxOccurs="unbounded" />

```

```

        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element
    name="OutputFormats"
    minOccurs="0">
    <xsd:complexType>
    <xsd:sequence>
    <xsd:element
        name="OutputFormat"
        maxOccurs="unbounded" >
        <xsd:complexType>
        <xsd:sequence>
        <xsd:choice>
        <xsd:element
            name="FormatURL"
            type="xsd:anyURI" />
        <xsd:element
            name="ObjectType"
            type="xsd:NMTOKEN" />
        </xsd:choice>
        <xsd:element
            name="Description"
            type="xsd:string" />
        <xsd:element
            name="XPathNames"
            type="xsd:boolean"/>
        </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element
    name="InputFormats"
    minOccurs="0">
    <xsd:complexType>
    <xsd:sequence>
    <xsd:element
        name="InputFormat"
        maxOccurs="unbounded" >
        <xsd:complexType>
        <xsd:sequence>
        <xsd:choice>
        <xsd:element
            name="FormatURL"
            type="xsd:anyURI" />
        <xsd:element
            name="ObjectType"
            type="xsd:NMTOKEN" />
        </xsd:choice>
        <xsd:element

```

```

                name="Description"
                type="xsd:string" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element
    name="MetadataFormats"
    minOccurs="0">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element
                name="MetadataFormat"
                maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element
                            name="FormatURL"
                            type="xsd:anyURI" />
                        <xsd:element
                            name="Description"
                            type="xsd:string" />
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:sequence>
<xsd:attribute
    name="version"
    type="xsd:anyURI" />
<xsd:attribute
    name="timestamp"
    type="xsd:dateTime" />
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType
    name="LookupList">
    <xsd:sequence>
        <xsd:element
            name="Map"
            maxOccurs="unbounded">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element
                        name="MapEntry"
                        maxOccurs="unbounded">

```

```

        <xsd:complexType>
          <xsd:sequence>
            <xsd:element
              name="Name"
              type="xsd:NMTOKEN" />
            <xsd:element
              name="Value"
              type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute
      name="version"
      type="xsd:anyURI"
      use="required" />
    <xsd:attribute
      name="timestamp"
      type="xsd:dateTime"
      use="required" />
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
<xsd:attribute
  name="version"
  type="xsd:anyURI"
  use="required" />
<xsd:attribute
  name="timestamp"
  type="xsd:date"
  use="required" />
</xsd:complexType>

<!-- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX -->

<xsd:complexType
  name="SearchRequestType">
  <xsd:sequence>
    <xsd:element
      name="Resource"
      type="xsd:NMTOKEN" />
    <xsd:element
      name="Query">
      <xsd:complexType>
        <xsd:simpleContent>
          <xsd:extension
            base="xsd:string">
            <xsd:attribute
              name="queryType"
              type="xsd:NMTOKEN" />
            <xsd:attribute
              name="vocabulary"

```

```

        type="xsd:NMTOKEN" />
    </xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>
<xsd:element
    name="Count"
    type="xsd:boolean"
    minOccurs="0" />
<xsd:choice>
    <xsd:element
        name="OutputFormatURL"
        type="xsd:anyURI" />
    <xsd:element
        name="OutputObject">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension
                    base="xsd:NMTOKEN">
                    <xsd:attribute
                        name="accept"
                        type="xsd:string" />
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
</xsd:choice>
<xsd:element
    name="Limit"
    type="xsd:integer"
    minOccurs="0" />
<xsd:element
    name="Offset"
    type="xsd:nonNegativeInteger"
    minOccurs="0" />
<xsd:element
    name="Select">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension
                base="xsd:string">
                <xsd:attribute
                    name="vocabulary"
                    type="xsd:NMTOKEN" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
</xsd:sequence>
<xsd:attribute
    name="taskID"
    type="xsd:ID" />

```

```

<xsd:attribute
  name="version"
  type="xsd:string" />
</xsd:complexType>

<xsd:complexType
  name="ManifestType">
  <xsd:sequence>
    <xsd:element
      name="Reference"
      maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element
            name="Description">
            <xsd:complexType>
              <xsd:simpleContent>
                <xsd:extension
                  base="xsd:string">
                  <xsd:attribute
                    name="lang"
                    type="xsd:NMTOKEN" />
                </xsd:extension>
              </xsd:simpleContent>
            </xsd:complexType>
          </xsd:element>
          <xsd:element
            name="Count"
            type="xsd:nonNegativeInteger"
            minOccurs="0"/>
          <xsd:element
            name="Content">
            <xsd:complexType>
              <xsd:simpleContent>
                <xsd:extension
                  base="xsd:base64Binary">
                  <xsd:attribute
                    name="id"
                    type="xsd:ID" />
                  <xsd:attribute
                    name="type"
                    type="xsd:string" />
                </xsd:extension>
              </xsd:simpleContent>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute
          name="id"
          type="xsd:ID"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute
    name="id"
    type="xsd:ID"/>
</xsd:complexType>
</xsd:element>

```

```

    </xsd:sequence>
  </xsd:complexType>

  <xsd:element
    name="SearchRequest"
    type="rets:SearchRequestType" />

  <xsd:element
    name="SearchResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element
          name="Manifest"
          type="rets:ManifestType"/>
      </xsd:sequence>
      <xsd:attribute
        name="correlationID"
        type="xsd:NCName"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element
    name="UpdateRequest">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element
          name="Manifest"
          type="rets:ManifestType"/>
      </xsd:sequence>
      <xsd:attribute
        name="resource"
        type="xsd:string"
        use="required"/>
      <xsd:attribute
        name="format"
        type="xsd:string"
        use="required"/>
      <xsd:attribute
        name="type"
        use="required">
        <xsd:simpleType>
          <xsd:restriction
            base="xsd:string">
            <xsd:enumeration
              value="Create"/>
            <xsd:enumeration
              value="Merge"/>
            <xsd:enumeration
              value="Replace"/>
            <xsd:enumeration
              value="Delete"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>

```

```

        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>
</xsd:element>

<xsd:element
    name="UpdateResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element
                name="Responses">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element
                            name="Response">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element
                                        name="Faults"
                                        type="rets:RetsFaultList"
                                        minOccurs="0"/>
                                </xsd:sequence>
                                <xsd:attribute
                                    name="uid"
                                    type="xsd:string"/>
                                <xsd:attribute
                                    name="resource"
                                    type="xsd:NMTOKEN"
                                    use="required"/>
                                <xsd:attribute
                                    name="correlationTaskID"
                                    type="xsd:NMTOKEN"/>
                                <xsd:attribute
                                    name="severity"
                                    type="rets:SeverityType"
                                    use="required"/>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:simpleType
    name="SeverityType">
    <xsd:restriction
        base="xsd:string">
        <xsd:enumeration
            value="None"/>
    </xsd:restriction>
</xsd:simpleType>

```

```

        value="Warning"/>
    <xsd:enumeration
        value="Fatal"/>
</xsd:restriction>
</xsd:simpleType>

<xsd:complexType
    name="RetsFaultList">
    <xsd:sequence>
        <xsd:element
            name="Fault"
            maxOccurs="unbounded">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element
                        name="Type"
                        type="xsd:NMTOKEN" />
                    <xsd:element
                        name="Message"
                        type="xsd:string"
                        minOccurs="0" />
                    <xsd:element
                        name="FieldList"
                        minOccurs="0" >
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element
                                    name="Field"
                                    type="xsd:string"
                                    maxOccurs="unbounded" />
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:attribute
                name="severity"
                type="rets:SeverityType"
                use="required"/>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:element
    name="ResourceListRequest">
    <xsd:complexType>
        <xsd:choice
            minOccurs="0">
            <xsd:element
                name="Resource"
                type="xsd:NMTOKEN" />
            <xsd:element

```

```

        name="TimeStamp"
        type="xsd:dateTime" />
    </xsd:choice>
</xsd:complexType>
</xsd:element>

<xsd:element
    name="ResourceList"
    type="rets:ResourceList" />

<xsd:element
    name="LookupListRequest">
    <xsd:complexType>
        <xsd:choice
            minOccurs="0">
            <xsd:element
                name="LookupName"
                type="xsd:NMTOKEN" />
            <xsd:element
                name="TimeStamp"
                type="xsd:dateTime" />
        </xsd:choice>
    </xsd:complexType>
</xsd:element>

<xsd:element
    name="LookupList"
    type="rets:LookupList" />

<xsd:element
    name="MetadataRequest">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element
                name="Resource"
                type="xsd:NMTOKEN" />
            <xsd:element
                name="MetadataFormatURL"
                type="xsd:anyURI" />
            <xsd:element
                name="Name"
                type="xsd:NMTOKEN"
                minOccurs="0" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element
    name="MetadataResponse" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:any

```

```

        namespace="##other" />
    </xsd:sequence>
</xsd:complexType>
</xsd:element>

</xsd:schema>
</wsdl:types>

<!-- ***** -->

<wsdl:message
    name="searchRequestMsg">
    <wsdl:part
        element="rets:SearchRequest"
        name="request" />
</wsdl:message>
<wsdl:message
    name="searchResponseMsg">
    <wsdl:part
        element="rets:SearchResponse"
        name="response" />
</wsdl:message>

<wsdl:message
    name="updateRequestMsg">
    <wsdl:part
        element="rets:UpdateRequest"
        name="request" />
</wsdl:message>
<wsdl:message
    name="updateResponseMsg">
    <wsdl:part
        element="rets:UpdateResponse"
        name="response" />
</wsdl:message>

<wsdl:message
    name="resourceListRequestMsg">
    <wsdl:part
        element="rets:ResourceListRequest"
        name="request" />
</wsdl:message>
<wsdl:message
    name="resourceListResponseMsg">
    <wsdl:part
        element="rets:ResourceList"
        name="response" />
</wsdl:message>

<wsdl:message
    name="lookupListRequestMsg">
    <wsdl:part

```

```

        element="rets:LookupListRequest"
        name="request" />
</wsdl:message>
<wsdl:message
    name="lookupListResponseMsg">
    <wsdl:part
        element="rets:LookupList"
        name="response" />
</wsdl:message>

<wsdl:message
    name="metadataRequestMsg">
    <wsdl:part
        element="rets:MetadataRequest"
        name="request" />
</wsdl:message>
<wsdl:message
    name="metadataResponseMsg">
    <wsdl:part
        element="rets:MetadataResponse"
        name="response" />
</wsdl:message>

<!-- ***** -->

<wsdl:portType
    name="retsTransaction">
    <wsdl:operation
        name="search">
        <wsdl:input
            name="SearchRequestDoc"
            message="rets:searchRequestMsg" />
        <wsdl:output
            name="SearchResponseDoc"
            message="rets:searchResponseMsg" />
    </wsdl:operation>

    <wsdl:operation
        name="update">
        <wsdl:input
            name="UpdateRequestDoc"
            message="rets:updateRequestMsg" />
        <wsdl:output
            name="UpdateResponseDoc"
            message="rets:updateResponseMsg" />
    </wsdl:operation>

    <wsdl:operation
        name="getResourceList">
        <wsdl:input
            name="ResourceListRequestDoc"
            message="rets:resourceListRequestMsg" />

```

```

        <wsdl:output
            name="ResourceListResponseDoc"
            message="rets:resourceListResponseMsg" />
    </wsdl:operation>

    <wsdl:operation
        name="getLookupList">
        <wsdl:input
            name="LookupListRequestDoc"
            message="rets:lookupListRequestMsg" />
        <wsdl:output
            name="LookupListResponseDoc"
            message="rets:lookupListResponseMsg" />
    </wsdl:operation>

    <wsdl:operation
        name="getMetadata">
        <wsdl:input
            name="MetadataRequestDoc"
            message="rets:metadataRequestMsg" />
        <wsdl:output
            name="MetadataResponseDoc"
            message="rets:metadataResponseMsg" />
    </wsdl:operation>

</wsdl:portType>

<!-- ***** -->

<wsdl:binding
    name="RetsSoapBinding"
    type="rets:retsTransaction">

    <soap12:binding
        style="document"
        transport="http://schemas.xmlsoap.org/soap/http" />

    <wsdl:operation
        name="search">
        <soap12:operation
            soapAction="" />
        <wsdl:input
            name="SearchRequestDoc">
            <soap12:body
                use="literal" />
        </wsdl:input>
        <wsdl:output
            name="SearchResponseDoc">
            <soap12:body
                use="literal" />
        </wsdl:output>
    </wsdl:operation>

```

```

<wsdl:operation
  name="update" >
  <soap12:operation
    soapAction="" />
  <wsdl:input
    name="UpdateRequestDoc" >
    <soap12:body
      use="literal" />
  </wsdl:input>
  <wsdl:output
    name="UpdateResponseDoc" >
    <soap12:body
      use="literal" />
  </wsdl:output>
</wsdl:operation>

<wsdl:operation
  name="getResourceList">
  <soap12:operation
    soapAction="" />
  <wsdl:input
    name="ResourceListRequestDoc" >
    <soap12:body
      use="literal" />
  </wsdl:input>
  <wsdl:output
    name="ResourceListResponseDoc" >
    <soap12:body
      use="literal" />
  </wsdl:output>
</wsdl:operation>

<wsdl:operation
  name="getLookupList">
  <soap12:operation
    soapAction="" />
  <wsdl:input
    name="LookupListRequestDoc" >
    <soap12:body
      use="literal" />
  </wsdl:input>
  <wsdl:output
    name="LookupListResponseDoc" >
    <soap12:body
      use="literal" />
  </wsdl:output>
</wsdl:operation>

<wsdl:operation
  name="getMetadata">

```

```

    <soap12:operation
      soapAction="" />
    <wsdl:input
      name="MetadataRequestDoc" >
      <soap12:body
        use="literal" />
    </wsdl:input>
    <wsdl:output
      name="MetadataResponseDoc" >
      <soap12:body
        use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>

<!-- ***** -->

<wsdl:service
  name="RetsTransactionService">
  <wsdl:port
    binding="rets:RetsSoapBinding"
    name="Rets">
    <soap12:address
      location="http://localhost:8080/rets2server" />
  </wsdl:port>
</wsdl:service>

</wsdl:definitions>

```