RESO™
REAL ESTATE STANDARDS ORGANIZATION

RETS 1.9 Specification

RETS 1.9 Specification

Chapters

Additional Sections

DTDs Related to this Version for Download

NOTE The links below are meant to be downloaded and are not intended for use when linking an application to a DTD. See Appendix A for the complete list of DTD references.

RETS Version	RETS Schema Format	Response Type	Download Link
1.9.0	Compact	Metadata	Download Schema (TBD)
1.9.0	Standard	Metadata	Download Schema (TBD)
1.9.0	Compact	Search	Download Schema (TBD)
1.9.0	Standard	Search	Download Schema (TBD)

RETS 1.9.0 Recent News & Updates

Recently Updated

As you and your team create content this area will fill up and display the latest updates.

Section 1 - Introduction

1.1 Purpose

The Real Estate Transaction Standard (RETS) is a specification for a standard communication method between computer systems exchanging real estate information. It defines a standard interface for use by applications such as agent desktop software, IDX (Internet Data Exchange) systems, data aggregation systems, and many other systems that store, display or operate on real estate listing, sales and other data.

This specification describes the Real Estate Transaction Standard communication protocol. Together with the companion XML DTDs (Document Type Definitions) listed in Appendix A, it constitutes the specification for the standard.

1.2 Scope

This specification is intended to define only the minimum a product or service must do in order to be considered "compliant". This specification is extensible and nothing in the specification precludes a vendor from adding data or functionality over and above that detailed here. However, when a function is provided or a data element is stored by a compliant system, it must offer access to the function or mechanism in a way that complies with the specification in order to be considered compliant.

1.3 Requirements

1.3.1 Required Features

This specification uses the same words as RFC 1123(1) for defining the significance of each particular requirement. These words are:

MUST	This word or the adjective "required" means that the item is an absolute requirement of the specification. A feature that the specification states MUST be implemented is required in an implementation in order to be considered compliant.
SHOULD	This word or the adjective "recommended" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course. A feature that the specification states SHOULD be implemented is treated for compliance purposes as a feature that may be implemented.
MAY	This word or the adjective "optional" means that this item is truly optional. A feature that the specification states MAY be implemented need not be implemented in order to be considered compliant. However, if it is implemented, the feature MUST be implemented in accordance with the specification.

An implementation is not compliant if it fails to satisfy one or more of the MUST requirements for the protocols it implements. An implementation that satisfies all the MUST and all the SHOULD requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST requirements but not all the SHOULD requirements for its protocols is said to be "conditionally compliant."

Client and server implementations should generally follow the Internet protocol convention of being strict in what they generate, but tolerant in what they accept. However, in cases where tolerance of deviations from the specification could result in an incorrect interpretation of user data or intentions, implementers are urged to reject transactions rather than supplying possibly-incorrect defaults.

1.3.2 Compatibility with Prior Versions

The RETS 1.9.0 specification supersedes previous versions of the RETS specification. There is no *requirement* for a client or server that advertises itself as "compliant with RETS 1.9.0" to interoperate with earlier versions. However, client and server implementers are urged to support the prior versions, RETS 1.8.0, RETS 1.7.2, RETS 1.7 and RETS 1.5, in order to insure a smooth transition.

1.4 Terminology

Class	A subset of data elements within a Resource that share common metadata elements.
Client	The system requesting data. This may well be a server seeking to update itself from another server. The specification does not assume any particular kind of client.
Endpoint	Either a server or client.
Metadata	The set of data that describes data fields in detail.
Metadata Dictionary	The set of data that describes the available metadata. It is used to determine the different classes of accessible data on the server and does not describe the fields within the those classes. It also defines what different types of searches are available (tax, open house, etc.)
Object	For purposes of RETS and its GetObject transaction, a collection of octets treated as a unit and associated with a unique resource element.
Optional	A field or feature described by this specification but not required for an endpoint to be considered compliant. The specification states the action to be taken by a compliant system in the absence of an optional field. The fact that the specification designates a field as optional does not mean that the recipient of a transaction that is missing optional fields is required to provide all services that could be required if the field were present.
Required	A compliant server or client MUST include any field designated as required. A transaction that does not include every required field MUST be rejected by the recipient.
Resource	A collection of data having the external appearance of belonging to a single database and being accessible for search or update via RETS transactions.
Resource Element	An individual record from a resource identified by a Resource Key.
Resource Key	The unique key that identifies a resource element.
Server	The system providing data (also referred to as the "host").
Request ID	A client-provided character string of up to 64 printable characters which uniquely identifies a request to a client. The contents are implementation-defined. Defined in Section 3.4, "Optional Client Request Header Fields".
StandardName	The name of a data field as it is known in the Real Estate Transaction Standard Data Dictionary.
SystemName	The name of a data field as it is known in the metadata.

Section 2 - Notational Conventions

2.1 Augmented BNF

This document expresses message layouts and character sequences in an augmented Backus-Naur Form (BNF) similar to that used by RFC 2822(4) and defined in RFC 2234(22).

2.2 Typographic Conventions

Parsing constructs and examples are set in a `monospaced` font:
`Server: Microsoft-IIS/4.0`

In parsing constructs, textual elements that are required exactly as shown are indicated by **`boldface`**., while textual elements that represent placeholders for actual data are indicated by a *`italic font:`*
`Server:`*`server identifier`*

Entities designated by a textual definition contain that definition enclosed in angle brackets:
<any 8-bit sequence of data>

Atoms and primitive entities are indicated by *`ITALIC CAPS`*
`1*64ALPHANUM`

Two nonprinting characters also have significance in some RETS constructs. These may be represented by special printing graphics to assist in understanding. Note that the printing graphics ARE NOT the octet representation. They are a placeholder for the correct octet representation;

	Tab character, ASCII HT, an octet with a value of **`09`**
	Space character, ASCII SP, an octet with a value of **`32`**. The symbol is used where needed for clarity.

Certain features of the standard may be superseded as the standard develops. These features should be avoided and are indicated by the text [deprecated] which will follow the first use of the feature terminology. Future releases of the standard may remove deprecated features.

2.3 Rules

The following rules are used throughout this specification to describe basic parsing constructs. The US-ASCII coded character set is defined by ANSI X3.4-1986(5).

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:
`m * n`

where both *m* and *n* are integers. *m* represents the minimum allowed number of repetitions, and *n* represents the maximum. If *m* is omitted, it is presumed to be zero; if *n* is omitted, it is presumed to be infinite. For example, the syntactic construct

`1*64ALPHANUM`

means a string of *`ALPHANUM`* s containing at least 1 and at most 64.

When a parsing construct is represented by a string of entities, some of which are optional, the optional entities are enclosed in square brackets. For example, in the string

*`error-number [error-code]`*the *`error-number`* entity is required, while the *`error-code`* entity is optional.

Elements separated by the vertical bar are alternatives. The entity description
`ALPHA | DIGIT`

means "either an *`ALPHA`* or a *`DIGIT`* ".

2.4 Atoms And Primitive Entries

`NOTE:` The definitions for `ALPHA` , `CHAR` , `CTL` , `DIGIT` , `HEXDIG` and `OCTET` are derived from RFC 2234.

ALPHA	::= %x41-5A | %x61-7A ; A-Z | a-z
CHAR	::= %x01-7F ; ANY 7-BIT US-ASCII CHARACTER, ; EXCLUDING NUL
CTL	::= %x00-1F | %x7F ; controls
DIGIT	::= %x30-39 ; 0-9
HEXDIG	::= *DIGIT* | "A" | "B" | "C" | "D" | "E" | "F"
OCTET	:= %x00-FF ; any 8-bit sequence of data
BOOLEAN	::= *TRUE* | *FALSE*
TRUE	::= "1"
FALSE	::= "0"
RETSID	::= *1*32ALPHANUM*
RETSNAME	::= *1*64IDALPHANUM*
rets-version-type	*1*2DIGITS* . *1*2DIGITS* . *1*5DIGITS* ; A convention to represent the version as a "<major>.<minor>.<release>" numbering scheme.
IDALPHANUM	::= *ALPHANUM* | "_"
ALPHANUM	::= *ALPHA* | *DIGIT*
SQLFIELDNAME	::= *ALPHA *31ALPHANUM* <except ANSI SQL 92 reserved words>
CR	::= <US-ASCII CR, carriage return (13)>
LF	::= <US-ASCII LF, linefeed (10)>
SP	::= <US-ASCII SP, space (32)>
HT	::= <US-ASCII HT, horizontal-tab (9)>
<"> or "	::= <US-ASCII double-quote mark (34)> ; this is a problematic use of the double quote in the BNF, use QUOTE for new BNF entries
QUOTE	::= %x22
NULL	::= <no character>
CRLF or	::= *CR LF* ; special character U+21B5 crarr
LWS	::= *[CRLF] 1*(SP | HT)*
HEX	::= "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f" | *DIGIT*
LHEX	::= "a" | "b" | "c" | "d" | "e" | "f" | *DIGIT*
OPTNONNEGATIVENUM	::= *NULL* | *NONNEGATIVENUM* ; null or >= 0
OPTPOSITIVENUM	::= *NULL* | *POSITIVENUM* ; null or >= 1
NONNEGATIVENUM	::= "0" | *POSITIVENUM* ; also known as cardinal numbers or counting numbers ; consisting of integers greater than 0

NONZERODIGIT	::= %x31-39 ; 1-9
PLAINTEXT	::= <any *OCTET* except *CTL* s>
POSITIVENUM	::= *NONZERODIGIT *DIGIT* ; > 0
SERIAL	::= "-1" | *NONNEGATIVENUM* ;
TEXT	::= <any *OCTET* except *CTL*s, but including LWS >

NOTE: Implementers are cautioned that the definition of the TEXT atom may conflict with certain outputs, in particular a collision between the delimiter octet of Section 7.2.1 and the output information when using the formats COMPACT or COMPACT-DECODED. Further, the definition may conflict with escaping rules for well-formed XML responses. The responsibility for resolving these conflicts lies with the transmitting party. In particular, the responses to Search, Update and GetMetadata may have this conflict.

TOKENCHAR	::= <any CHAR except CTL s or TSPECIALS >
TOKEN	::= *1*TOKENCHAR*
TSPECIALS	::= "(" | ")" | "<" | ">" | "@" | "," | ";" | ":" | "\\" | <"> | "/" | "[" | "]" | "?" | "=" | "}" | "{" | SP | HT
quoted-string	::= (<"> *(*QDTEXT*) <">)
QDTEXT	::= <any *TEXT* except <">>
RETSDATETIME	::= *date-time* | *partial-date-time*
RETSTIME	::= *full-time* | *partial-time*
DATE	::= <Date using the format defined in RFC 2616 as *rfc1123-date*>

NOTE: The definitions for the date and time are derived from RFC 3339.

date-fullyear	::= 4*DIGIT*
date-month	::= 2*DIGIT* ; 01 - 12
date-mday	::= 2*DIGIT* ; 01 - 28, 01-29, 01-30, 01-31, based on month/year
time-hour	::= 2*DIGIT* ; 00 - 23
time-minute	::= 2*DIGIT* ; 00 - 59
time-second	::= 2*DIGIT* ; 00 - 58, 00 - 59, 00 - 60 based on leap second rules
time-secfrac	::= "."*1DIGIT*
time-numoffset	::= ("+"|"-") *time-hour* ":" *time-minute*
time-offset	::= "Z" | *time-numoffset*
partial-time	::= *time-hour* ":" *time-minute* ":" *time-second* [*time-secfrac*]
full-date	::= *date-fullyear* " - " *date-month* " - " *date-mday*
full-time	::= *partial-time time-offset*
date-time	::= *full-date* "T" *full-time*
partial-date-time	::= *full-date* "T" *partial-time*

NOTE: ISO 8601, RFC 3339 and the W3C note provide for additional constraints to the formats. Based on common usage patterns, this standard applies the following additional constraints to improve interoperability and compatibility. The representation of the time offset UTC character 'Z' and the date-time separator character 'T' MUST be upper case.

The time-secfrac is limited to one digit only. The date and time representations are intended for machine processing, therefore, no whitespace is expected in any of the atoms. Examples of the format are similar to that of the W3C note, for example, 1997-07- 16T19:20:30.4+01:00 or

1997-07-16T18:20:30.4Z. Servers and Clients MUST treat the time-offset 'Z' and "+00:00" as identical times. Servers and Clients MAY use the interpretation of RFC 3339 section 4.3 Unknown Local Offset Convention where the time-offset "-00:00" is semantically different from "+00:00" and represents a known UTC time but unknown local time.

URI	::= scheme ":" hier-part ["?" query] ["#" fragment]
hier-part	::= "//" authority path-abempty | path-absolute | path-rootless |path-empty
scheme	::= ALPHA *(ALPHA |DIGIT |"+" |"-" |".")
authority	:= [userinfo "@"] host [":" port]
userinfo	::= *(unreserved |pct-encoded |sub-delims |":")
host	::= IP-literal |IPv4address |reg-name
port	::= *DIGIT
IP-literal	::= "[" (IPv6address |IPvFuture) "]"
IPvFuture	::= "v" 1*HEXDIG "." 1*(unreserved |sub-delims |":")
IPv6address	::= 6(h16 ":") ls32 |"::" 5(h16 ":") ls32 |[h16] "::" 4(h16 ":") ls32 |[*1(h16 ":") h16] "::" 3(h16 ":") ls32 |[*2(h16 ":") h16] "::" 2(h16 ":") ls32 |[*3(h16 ":") h16] "::" h16 ":" ls32 |[*4(h16 ":") h16] "::" ls32 |[*5(h16 ":") h16] "::" h16 |[*6(h16 ":") h16] "::"
h16	::= 1*4 HEXDIG
ls32	::= (h16 ":" h16) / IPv4address
IPv4address	::= dec-octet "." dec-octet "." dec-octet "." dec-octet
dec-octet	::= DIGIT ; 0-9 |%x31-39 DIGIT ; 10-99 |"1" 2DIGIT ; 100-199 |"2" %x30-34 DIGIT ; 200-249 |"25" %x30-35 ; 250-255
reg-name	::= *(unreserved / pct-encoded / sub-delims)
path	::= path-abempty ; begins with "/" or is empty | path-absolute ; begins with "/" but not "//" |path-noscheme ; begins with a non-colon segment |path-rootless ; begins with a segment |path-empty ; zero characters
path-abempty	::= *("/" segment)
path-absolute	:= "/" [segment-nz *("/" segment)]
path-noscheme	::= segment-nz-nc *("/" segment)
path-rootless	::= segment-nz *("/" segment)
path-empty	::= 0<pchar>
segment	::= *pchar
segment-nz	::= 1*pchar
segment-nz-nc	::= 1*(unreserved |pct-encoded |sub-delims |"@") ; non-zero-length segment without any colon ":"
pchar	::= unreserved |pct-encoded |sub-delims |":" |"@"

query	`::=` *(pchar |"/" |"?")
fragment	`::=` *(pchar |"/" |"?")
pct-encoded	`::=` "%" *HEXDIG HEXDIG*
unreserved	`::=` *ALPHA|DIGIT* |"-" |"." |"_" |"~"
reserved	`::=` gen-delims |sub-delims
gen-delims	`::=` ":" |"/" |"?" |"#" |"[" |"]" |"@"
sub-delims	`::=` "!" |"$" |"&" |"'" |"(" / ")"| "*" |"+" |","| ";" |"="

Note: The definition for URI is derived from RFC 3986.

Note: An Approved RCP is Related to this Section
Section 2.4 is related to the following approved RCP(s):

RETS 1.7.2

- RCP 71 Time Zone Data

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Section 3 - Message Format

RETS uses HTTP(2) (23) for sending messages between clients and servers. It defines three additional HTTP headers, and some RETS transactions constrain the values of certain headers defined by HTTP and/or make certain headers designated as optional in HTTP mandatory when used for RETS. In addition, RETS requests use HTML 4.01(16) form encoding to encapsulate request parameters. In addition, a compliant RETS client MUST implement cookie handling as specified in RFC 2109 and are encouraged to implement cookie handling as specified in RFC 6265(15).

The information below summarizes some of the requirements of HTTP and HTML 4.01 for ease of reference. However, in all cases, the underlying standards are the normative references for message formats.

3.1 General Message Format

3.1.1 RETS HTTP/1.1 Encapsulation

RETS messages are encapsulated as the bodies of HTTP[2] [23] requests and responses. The request body may be null, depending on the request. The response body is never null.

As defined in RFC 2822[4] , keywords in header key-value pairs are not case-sensitive. The values, however, may be case-sensitive depending on context.

3.1.2 Request Arguments

RETS requests are HTML 4.01- compliant form submissions, following all of the specifications in the HTML 4.01 recommendation. Note that the HTML 4.01 specification provides that:

- Key names in key/value pairs are not case-sensitive.
- Both key names and key values MUST be encoded as specified in HTML 4.01 section 17.13.4, with + characters replacing spaces, and then reserved characters being escaped per RFC 2396[13]

3.1.3 Response Bodies

The body of a response to most RETS requests is a well-formed XML document; the exceptions are the Get transaction (section 8) and the GetObject transaction (section 5). This means that servers must construct the body in accordance with the XML specification[17], and that clients must parse the body in accordance with that specification.

Note: RETS 1.9.0: An approved RCP is related to this section
Section 3.1 is related to the following approved RCP(s):

- RCP 107 IETF HTTP RFC Updates

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

3.2 Request Format

A RETS request is either an HTTP GET request or an HTTP POST request. In the case of the GET-request the Argument-List is appended to the Request-URI after a delimiting question mark ("?"). For the post-request the Argument-List is sent as the first entity body for the POST method.

get-request	::= **GET** *Request-URI* [**?** *Argument-List*] *HTTP-Version* *CRLF* *message-header *CRLF*

post-request	::= **POST** *Request-URI HTTP-Version CRLF* **message-header* *CRLF* [*Argument-List*]

The *`Request-URI`*, *`HTTP-Version`* and *`message-header`* are defined in the HTTP Specification[2] [23]. The detailed construction of the *`Argument-List`* is defined in HTML 4.01[16]

.

Note: RETS 1.9.0: An approved RCP is related to this section
Section 3.1 is related to the following approved RCP(s):

- RCP 107 IETF HTTP RFC Updates

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

3.3 Required Client Request Header Fields

The HTTP header of any messages sent from the client MUST contain the following header fields:

`User-Agent`	This header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations, as well as providing enhanced capabilities to some user-agents. All client requests MUST include this field. This is a standard HTTP header field as defined in the HTTP Specification[2] [23].
User-Agent	::= **User-Agent:** 1* *product*
product	::= *token* [/ *product-version*]
product-version	::= *token*

Example: `User-Agent: CMAZilla/4.00`

Product tokens should be short and to the point: use of them for advertising or other non-essential information is explicitly forbidden. Although any token character may appear in a product-version, this token SHOULD only be used for a version identifier (i.e., successive versions of the same product SHOULD only differ in the product-version portion of the product value). For more information about User-Agent see the HTTP Specification[2] [23].

A server MAY advertise additional capabilities based on the client's User-Agent, and MAY refuse to proceed with the authorization if an acceptable User-Agent has not been supplied. A server MAY also choose to authenticate the client's identity cryptographically using the RETS-UA-Authorization header; see section 3.4 for additional information.

`X-RETS-Version`	The client MUST send the X-RETS-Version.The use of the *X-RETS-Version* as a maker for versioning metadata may require additional digits to correctly represent the version of metadata. Specifically, implementers should be permissive in the use of *X-RETS-Version* and should accept values where there are more than a single digit for the release or minor positions.The convention used is a numbering scheme similar to the HTTP-version of the Protocol Versioning section [2] [23]. The version of a RETS message is indicated by a X-RETS-Version field in the header of the message.
X-RETS-Version	::= **"X-RETS-Version:"** *version-info*
version-info	::= **"RETS/"** 1**DIGIT* **"."** 1**DIGIT* **"."** 1**DIGIT*

Example: `X-RETS-Version: RETS/1.9.0`

The behaviour of a request may depend on the correct handling of header Cookies are handled as specified in RFC 6265 15.

`Cookie`	Servers and clients are encouraged to implement cookie handling as specified in RFC 6265 [15]. The client MUST implement cookie handling as specified in RFC 2109 and MAY implement cookie handling as specified in RFC 6265. Since RFC 6265 is the current version of the standard, current libraries will use this over the obsolete RFC 2109 and will handle cookies as described in RFC 6265. Future versions of this standard will obsolete the reference to RFC 2109. If any server response has included a valid Set-Cookie header, and the cookie in that header has not expired, the client MUST return the corresponding Cookie header. See RFC 2109, 6265 for the full specification.

Note: RETS 1.8.0: An approved RCP is related to this section
Section 3.3 is related to the following approved RCP(s):

- RCP 87 RETS 1.7.2 Errata Document

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Note: RETS 1.9.0: An approved RCP is related to this section
Section 3.3 is related to the following approved RCP(s):

- RCP 106 Client cookie support for RFC 6265
- RCP 107 IETF HTTP RFC Updates
- RCP 108 HTTP User Space Headers

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

3.4 Optional Client Request Header Fields

`Authorization`	Authorization header field as defined in the HTTP Specification [2 23]. See Section 4.1, "Security", as well as the HTTP Specification [2 23], for additional information.
`X-RETS-Request-ID`	A character string of printable characters which the client can use to identify this request. The contents are implementation-defined. If this field is included in a request from the client then the server MUST return it in the response.
`X-RETS-Request-ID`	*`::= 1*64ALPHANUM`*
`Accept-Encoding`	A comma-separated list of MIME types indicating the content encoding schemes that the client is willing to accept. This is intended to support the use of compression in data returns; see section 3.8 for additional information.
`Accept-Encoding`	*`::= 1*64ALPHANUM/1*64ALPHANUM *[,1*64ALPHANUM/1*64ALPHANUM...]`*
`X-RETS-UA-Authorization`	A client MAY support authentication of its User-Agent value by including the X-RETS-UA-Authorization header. Servers MAY require this header with a valid value before providing services.
`X-RETS-UA-Authorization`	`::= ua-method ua-digest-response`
`ua-method`	`::=` **`Digest`**

`ua-digest-response`	::= "*LHEX " See section 3.10 for the method of computing the `ua-digest-response` value. The client MAY send this header under any circumstances. It need not send this header if the server has not indicated that it requires user-agent authentication by responding to a transaction with a RETS error code of 20037.

In addition to the header fields listed here, the client may send any header compliant with the HTTP Specification [2] [23]

Note: RETS 1.9.0: An approved RCP is related to this section
Section 3.4 is related to the following approved RCP(s):

- RCP 107 IETF HTTP RFC Updates
- RCP 108 HTTP User Space Headers

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

3.5 Response Format

The general server response to a request is either a well-formed XML document returning RETS-encapsulated data or error information, or, for the `Get` transaction and for successful `GetObject` transactions, the content of the requested object in the format given in the response's HTTP Content-Type header. Note that this is an ordinary HTTP response as described in the HTTP Specifications [2] [23].

The more common HTTP *Status-Codes* are provided in Section 3.9 , though any status code defined in the HTTP Specifications [2] [23] is permissible. Servers MUST use appropriate predefined status codes when communicating with the client.

The *Status-Code* is intended to provide HTTP level errors to the client (Authorization, URI, etc.). Software level errors (search queries, invalid argument values, etc.) should be returned in the reply-code. If the server is unable to determine that a particular request is in fact a RETS request, it MUST return an HTTP status code indicating the type of error.

Except in those transactions specifically stating otherwise, a RETS response body is a well-formed XML document with the following general form:

response-body	::= RETS-response
RETS-response	::= body-start-line response [rets-status] [body-end-line]
body-start-line	::= **<RETS** 1*SP **ReplyCode=** quoted-reply-code 1*SP **ReplyText=** quoted-string *SP**>**
response	::= {key-value-body | data}
key-value-body	::= **<RETS-RESPONSE>**CRLF *(key = value CRLF) **</RETS-RESPONSE>**
rets-status	::= **<RETS-STATUS** [1*SP **ReplyCode=**quoted-end-reply-code 1*SP **ReplyText=**quotedstring *SP]**/>**

The *rets-status* MAY be included in the response if the ReplyCode or ReplyText given in the *body-start-line* becomes invalid during the creation of the response. If the server includes a *rets-status* in its reply, the client MUST use the ReplyCode and ReplyText from the rets-status rather than from the body-start-line.

body-end-line	::= **</RETS>**

If a *body-start-line* is returned in the response then the *body-end-line* MUST also be returned.

quoted-reply-code	::=**<">***1*5DIGITS***<">**

The *reply-code* is included to provide a mechanism to pass additional information to the client in the event that the request is processed OK (Status-Code = 200) but some condition still exist that may require an action by the client. A value of '0' indicates success. *reply-codes* are specific to a transaction. Please refer to the applicable transaction for the meaning of the *reply-code* or refer to Appendix C of this document for a consolidated list.

quoted-end-reply-code	::= **<">***1*5DIGITS***<">**

The *`end-reply-code`* is included to provide a mechanism to pass additional information to the client in the event that the request being processed by the server errors before the request has been completed. This allows the server to start streaming out data before it has completed processing the request. A value of **`0`** indicates success, however the server SHOULD only send an *`end-reply-code`* if there is an error. The valid <*`key`*> , <*`value`*> and <*`data`*> elements are defined in the Response Arguments section for each transaction.

NOTE

> RETS 1.8.0 requires all server responses to be well-formed XML, In addition, this specification requires that clients parse RETS responses as XML, not as simple text streams. The response formats shown here are normative with respect to content, but not normative with respect to form. 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 white space. See the W3C *XML Recommendation 1.0, Third Edition*, for full information on XML.

NOTE

> This section describes the general form of a response. Please refer to individual sections and the metadata to see the exact form of a response for a specific request.

An example server-reply where the reply body consists of key-value pairs:

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Sun, 20 Mar 2005 12:03:38 GMT
Content-Type: text/xml
Cache-Control: private
RETS-Version: RETS/1.8.0
CRLF
<RETS ReplyCode="0" ReplyText="SUCCESS" >
<RETS-RESPONSE>
Key1=Value1
Key2=Value2
</RETS-RESPONSE>
</RETS> CRLF
```

> **Note: RETS 1.8.0: An Approved RCP is Related to this Section**
> Section 3.5 is related to the following approved RCP(s):
>
> - RCP 87 RETS 1.7.2 Errata Document
>   Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

> **Note: RETS 1.9.0: An approved RCP is related to this section**
> Section 3.5 is related to the following approved RCP(s):
>
> - RCP 107 IETF HTTP RFC Updates
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

3.6 Required Server Response Header Fields

The HTTP header of any messages sent from the server MUST contain the following header fields:

`Date`	The server MUST send the date using the format defined in the HTTP Specification[2] [23]. using format *`rfc1123-date`*.

Example: `Date: Sun, 20 Mar 2005 12:03:38 GMT`

As defined by *`rfc1123-date`*, the Date MUST be represented in Greenwich Mean Time (GMT), without exception.

`Cache-Control`	The HTTP Specification[2] [23] standard general-header field is used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain. The directives specify behavior intended to prevent caches from adversely interfering with the request or response. This field SHOULD be set to "private" for all transaction in this specification.

Example: `Cache-Control: private`

`Content-Type`	This is a standard HTTP header field as defined in the HTTP Specification[2] [23]. It specifies the media type of the underlying data. The server MUST return this field in all replies. For most replies this will be set to **`"text/xml"`**. See Section 5.5 in the GetObject Transaction for exceptions and more information on this field.

Example: `Content-Type: text/xml`

`X-RETS-Version`	The server MUST send the X-RETS-Version. The convention used is a "<major>.<minor>.<revision>" numbering scheme similar to the HTTP-version of the Protocol Versioning section[2] [23]. The version of a RETS message is indicated by a`RX-RETS-Version field` in header of the message.
`X-RETS-Version`	`::=` **`"X-RETS-Version:"`** *`version-info`*
`version-info`	`::=` **`"RETS/"`** `1*`*`DIGIT`* **`"."`** `1*`*`DIGIT`* **`"."`** `1*`*`DIGIT`*

Example: `X-RETS-Version: RETS/1.9.0`

Applications sending request or response messages, as defined by this specification, MUST include a X-RETS-Version of `"RETS/1.9.0"`. Use of this version number indicates that the sending application is compliant with this specification.

Note: RETS 1.7.2: An Approved RCP is Related to this Section
Section 3.6 is related to the following approved RCP(s):

- RCP 71 Time Zone Data

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Note: RETS 1.9.0: An approved RCP is related to this section
Section 3.6 is related to the following approved RCP(s):

- RCP 107 IETF HTTP RFC Updates
- RCP 108 HTTP User Space Headers

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

3.7 Optional Server Response Header Fields

`Content-Length`	The Content-Length entity-header field indicates the size of the message-body, in decimal number of octets. This is a standard header field defined in the HTTP Specifications(2) (23) and is required for all requests containing a message-body not using Chunked transfer encoding.
`Transfer-Encoding`	The Transfer-Encoding entity-header field when set to the Chunked value, indicates the size of the message-body is in the chunk stream. This is a standard header field defined in the HTTP Specifications(2) (23) and is required for all responses with a body not using `Content-Length` or a `Content-Type: Multipart` response.
`Content-Encoding`	The Content Encoding entity-header field MAY be returned by the server if the client has included an AcceptEncoding header in its request () indicating that it can accept one or more compression types supported by the server. It is recommended that servers accept at least **`application/gzip`** (see 3.8 Data Compression in RETS Transactions).
`Content-Encoding`	*`::= 1*64ALPHANUM / 1*64ALPHANUM`*
`X-RETS-Request-ID`	The contents of the `X-RETS-Request-ID` header, if any, sent by the client in the request. If an `X-RETS-Request-ID` is included in a request from the client then the server MUST return it in the response.
`X-RETS-Request-ID`	*`::=1*64ALPHANUM`*
`Server`	The server standard response-header field contains information about the software used to handle the request. The format of this field specified in the HTTP Specifications(2) (23).

Example: `Server: Microsoft-IIS/4.0`

`X-RETS-Server`	The RETS server vendor and server-controlled version number. This is not necessarily the same as the Server response-header field; it will be different if the HTTP server is separate from the RETS server. The format of this field is specified in the HTTP Specifications(2) (23).

Example: `X-RETS-Server: AcmeRETS/1.0`

`Set-Cookie`	The server MAY use HTTP cookies to maintain state information. See RFC 6265(15) (obsolete standard 2109) for the format of the `Set-Cookie` header. A cookie having a name of `RETS-Session-ID` defines the RETS session ID, which is used in calculating the RETS User-Agent Authentication (section 3.10). Cookies with other names have no special meaning in RETS but MAY be used when necessary.

`X-RETS-MetadataVersion`	::=metadata-identity / rets-version ;The X-RETS-MetadataVersion header permits the server to communicate to the client what metadata identity and version is applied to the response of the transaction. This is primarily intended to communicate metadata change to the client in an immediately discoverable manner.
`metadata-identity`	::= *1*128ALPHANUM* ;The metadata-id for the METADATA-SYSTEM that this response is built from.
rets-version	::=rets-version-type ;The version of RETS that this response is built for. See Section 2.4.

Example: `X-RETS-MetadataVersion: 14576/1.9.0`

In addition to the header fields listed here, the server may send any header compliant with the HTTP Specifications(2) (23).

Note: RETS 1.9.0: An approved RCP is related to this section
Section 3.7 is related to the following approved RCP(s):

- RCP 107 IETF HTTP RFC Updates
- RCP 106 Client Cookie Support
- RCP 108 IETF HTTP User Space Headers
- RCP 112 - RETS Metadata Version Header

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

3.8 Data Compression in RETS Transactions

Clients and servers may choose to support data compression in data returned from the server. To indicate its willingness to accept compressed data, a client includes an `AcceptEncoding` header in its request. If the server supports one of the compression methods accepted by the client, it can include a `Content-Encoding` header in its response indicating the compression method it has chosen.

Clients and 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. Clients and servers are free to choose other encoding methods as well.

Note: RETS1.8.0: An Approved RCP is Related to this Section
Section 3.8 is related to the following approved RCP(s):

- RCP 87 RETS 1.7.2 Errata Document
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

3.9 General Status Codes

Any of the following status codes (in addition to the others provided in the HTTP Specifications(2) (23) may be returned by a server in response to any request:

Table 3-1 General Status Codes

Status	Meaning
200	Operation successful.
400	Bad Request The request could not be understood by the server due to malformed syntax.
401	Not Authorized Either the header did not contain an acceptable Authorization or the username/password was invalid. The server response MUST include a WWW-Authenticate header field.
402	Payment Required The requested transaction requires a payment which could not be authorized.
403	Forbidden The server understood the request, but is refusing to fulfill it.
404	Not Found The server has not found anything matching the Request-URI.
405	Method Not Allowed The method specified in the Request-Line is not allowed for the resource identified by the Request-URI.
406	Not Acceptable The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request.

408	Request Timeout The client did not produce a request within the time that the server was prepared to wait.
411	Length Required The server refuses to accept the request without a defined Content-Length.
412	Precondition Failed Transaction not permitted at this point in the session
413	Request Entity Too Large The server is refusing to process a request because the request entity is larger than the server is willing or able to process.
414	Request-URI Too Long The server is refusing to service the request because the Request-URI is longer than the server is willing to interpret. This error usually only occurs for a GET method.
500	Internal server error. The server encountered an unexpected condition which prevented it from fulfilling the request.
501	Not Implemented The server does not support the functionality required to fulfill the request.
503	Service Unavailable The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.
505	HTTP Version Not Supported The server does not support, or refuses to support, the HTTP protocol version that was used in the request message.

HTTP error status returns are only to be used for system level, transport syntax, and invalid transaction errors. RETS error status codes are used to indicate errors in the request arguments or the transaction processing.

3.10 Computing the RETS-UA-Authorization Value

The RETS User Agent Authorization digest response value is used in the `X-RETS-UA-Authorization` header specified in section 3.4 . It is computed as follows:

`a1`	`::= MD5( product ":" UserAgent-Password )`
`ua-digest-response`	`::= LHEX( MD5( LHEX(a1)":" RETS-Request-ID ":" session-id ":" version-info))`

where:

`product`	The first *`product`* value taken from the User-Agent header (section 3.3). Note that the *`product`* value consists of both the product token and version.
`UserAgent-Password`	`::=`*`TOKEN`* This value is a secret shared between the client and server.
`RETS-Request-ID`	`::=`*`X-RETS-Request-ID`* This value MUST be the same as that sent with the `X-RETS-Request-ID` header. If the client does not use the `X-RETS-Request-ID` header, this token is empty in the calculation.
`session-id`	`::=` If the server has sent a `Set-Cookie` header with a cookie name of `X-RETS-Session-ID`, `session-id` is the value of that cookie. If the server has not sent a cookie with that name, or if the cookie by that name has expired, this token is empty in the calculation.
`version-info`	`::=` The value of the `X-RETS-Version` header sent by the client with this transaction.

Each individual value in the concatenated string is included with whitespace removed from the beginning and end of that element, that is, there is no whitespace on either side of the delimiting colon characters.

The method of performing the MD5 calculation is given in RFC 1321.

Note: RETS 1.8.0: An Approved RCP is Related to this Section
Section 3.6 is related to the following approved RCP(s):

- RCP 87 RETS 1.7.2 Errata Document

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Note: RETS 1.9.0: An approved RCP is related to this section
Section 3.10 is related to the following approved RCP(s):

- RCP 108 IETF HTTP User Space Headers

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Section 4 - Login Transaction

A client MUST issue a login request prior to proceeding with any other request. The Login transaction verifies all login information provided by the user and begins a RETS session. Subsequent session control may be mediated by HTTP cookies or any other method, though clients are required to support at least session control via HTTP cookies. Session Protocol describes the session protocol in detail.

The server's response to the Login transaction contains the information necessary for a client to issue other requests. It includes URLs that may be used for other RETS requests, and may also contain identity and parameter information if required by the functions supported by the server

4.1 Security

4.1.1 User Authentication

While this specification does not require the use of security — it is permissible, for example, to operate a publicly-accessible RETS server — most operators of RETS servers will wish to authenticate users. A server that requires that users be authenticated MAY implement RFC 7235 (2) , HTTP Authentication. The use of at least digest authentication is strongly recommended.

4.1.2 Client Authentication

Client authentication may be performed through the use of the optional X-RETS-UA-Authorization header (section 3.4) . Prior versions of this specification used a specially-calculated cnonce value in the Authorization header to implement this function. A server implementing this version of the RETS specification MUST accept the X-RETS-UA-Authorization header for client authentication. It MAY accept RFC 7235 (2) -style authentication as in prior versions of the RETS specification.

4.1.3 Data Security

The need for secure HTTP transactions cannot be met by authentication schemes. Vendors MAY require authentication and transport to be secured. For those needs, HTTP-over-TLS (commonly known as HTTPS) is a more appropriate protocol (24). A compliant server MAY support only HTTP-over-TLS. In this case, the server SHOULD listen on port 12109 rather than the standard RETS port, 6103. Clients should use the Capability URL List to determine the correct port and protocol to talk to the server for any transaction.

Note: RETS 1.9.0: An approved RCP is related to this section
Section 4.1 is related to the following approved RCP(s):

- RCP 107 IETF HTTP RFC Updates
- RCP 109 - Update TLS specification references

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

4.2 Authorization Example

The following example assumes that a client application is trying to access the Login URI on the server using the POST method, and without using client authentication. The URI is `"http://www.example.com/login"`. Both client and server know that the username is "joesmith", and the password is "SuperAgent". The example also assumes the use of authentication using RFC 2617.

The first time the client requests the document, no Authorization header is sent, so the server responds with:

```
HTTP/1.1 401 Unauthorized
```

```
WWW-Authenticate: Digest realm="Users@example.com",

nonce="dcd98b7102dd2f0e8b11d0f600bfb0c0"
opaque="5ccdef346870ab04ddfe0412367fccba"
```

The client may prompt the user for the username and password, after which it will respond with a new request, including the following Authorization header:

```
Authorization: Digest username="joesmith",
realm="Users@example.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c0",
opaque="5ccdef346870ab04ddfe0412367fccba",
uri="/login",
response="13258d9b0bc217c9502b47e32dff8ee9"
```

4.3 Required Request Arguments

There are no required request arguments.

4.4 Optional Request Arguments

4.4.1 BrokerCode Argument

brokerCodeArgument	::= **Broker =** *broker-code* [**,** *broker-branch*]

Some servers may support the scenario where a user belongs to multiple brokerages. If this is the case then the broker information (broker-code and broker-branch) must be input during login. If they are not included then the list of broker codes/branches is passed back to the client application through the response along with a "20012 Broker Code Required" reply-code.

broker-code	::= *1*24ALPHANUM*
broker-branch	::= *1*24ALPHANUM*

Note: RETS 1.8.0: An Approved RCP is Related to this Section
Section 4.4.1 is related to the following approved RCP(s):

- RCP 65 Session information tokens
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

4.4.2 SavedMetadataTimestamp Argument

savedMetadataTimestamp	::= **SavedMetadataTimestamp =** *saved-timestamp*

The client MAY inform the server of the timestamp associated with the version of metadata that it has currently saved. The server MAY use this to adapt to an earlier version of metadata than it chooses to advertise, or simply log the value to note out-of-date client metadata, or ignore the value entirely. In particular, the server is not required to alter its behavior in any way based on the value of this argument.

saved-timestamp	::= *RETSDATETIME*

Note: RETS 1.7.2: An Approved RCP is Related to this Section
Section 4.4.2 is related to the following approved RCP(s):

- RCP 71 Time Zone Data

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

4.5 Optional Response Header Fields

There are no additional optional response header fields.

4.6 Login Response Body Format

The body of the login response has three basic formats when replying to a request. The simplest form is when there is an error:

```
<RETS 1*SP ReplyCode= quoted-reply-code 1*SP
ReplyText= quoted-string *SP />
```

The second case is where the user belongs to more than one broker and they have not provided broker information as part of the login. The reply contains a list of all brokerages the user belongs to.

```
<RETS ReplyCode = "20012" 1*SP ReplyText = quoted-string SP >
<RETS-RESPONSE>CRLF
2*( brokerCodeArgument CRLF )
</RETS-RESPONSE>
</RETS>
```

The definition for brokerCodeArgument is provided in section 4.4.1.

The third case is the normal "OK" response. In this case several arguments are passed back to the client in the response.

```
<RETS 1*SP ReplyCode= quoted-reply-code 1*SP
ReplyText= quoted-string *SP>
<RETS-RESPONSE>
[member-name-key ]
[user-info-key ]
[broker-key ]
[metadata-ver-key ]
[metadata-timestamp-key ]
[min-metadata-timestamp-key ]
[office-list-key ]
[balance-key ]
[timeout-key ]
[pwd-expire-key ]
*( info-token-key )
capability-url-list
</RETS-RESPONSE>
[<RETS-STATUS [1*SP ReplyCode= quoted-end-reply-code 1*SPReplyText=
quoted-string *SP]/>
</RETS>
CRLF
```

> **Note: RETS 1.8.0: An Approved RCP is Related to this Section**
> Section 4.6 is related to the following approved RCP(s):
>
> - RCP 65 Session information tokens
>   Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

4.7 Required Response Arguments

4.7.1 Session Information Tokens

info-token-key	::= **Info =** *info-token-name* [; *info-token-type*] ; *info-token-value CRLF*
info-token-name	::= *RETSNAME*
info-token-type	::= *TOKEN*
info-token-value	::= *TEXT*

An information token is a named and typed piece of information about the current session. This information is sent by the server to the client for use in various contexts. For example, session and password management, creating search queries targeted to the current user or in validation expressions (see Table 11-43).

Any number of information tokens can be sent in the Login response, provided all of them have unique names.

The *info-token-type* may be any of the DataTypes defined in Table 11-15. The *info-token-value* must conform to the token's data type.

If the *info-token-type* is null or missing, the data type of the token is Character. In this case, the *info-token-value* MUST NOT include semicolons. When the *info-token-name* is defined in Table 4-1, Servers MUST use the DataType described therein. Clients SHOULD be permissive, that is, when a Server has omitted an info-token-type for a well-known name, Clients should infer the DataType based on the name defined in Table 4-1.

Names and types of well known tokens are listed in Table 4-1. The server MUST send all *info-token-name* names shown in bold in that table. The server MUST use the well-known name for optional *info-toke-name* names when providing information for those arguments (see Section 4.8). For forward compatibility, Clients MUST use the Session Information Token *info-token-value.*

Names, data types and the corresponding response argument for well-known tokens are listed in Table 4-1.

Table 4-1 Well-Known Information Tokens

Token name	Data type	Deprecated argument
USERID	*`user-id`*	user-id
USERCLASS	*`user-class`*	user-class
USERLEVEL	*`user-level`*	user-level
AGENTCODE	*`agent-code`*	agent-code
BROKERCODE	*`broker-code`*	broker-code
BROKERBRANCH	*`broker-branch`*	broker-branch
MEMBERNAME	*`member-name`*	member-name
MetadataID	*`metadata-id`*	none
MetadataVersion	*`metadata-version`*	metadata-version
MetadataTimestamp	*`metadata-timestamp`*	metadata-timestamp-key
MinMetadataTimestamp	*`min-metadata-timestamp`*	min-metadata-timestamp-key
Balance	*`balance`*	balance
TimeoutSeconds	*`timeout`*	timeout-key
PasswordExpiration	*`pwd-expire-date`*	pwd-expr
WarnPasswordExpirationDays	*`pwd-expire-warn`*	expr-warn-per
OfficeList	Character	office-list-key The value of the OfficeList token will be comma-delimited, rather than semicolon-delimited as it was in the case of the OfficeList response argument
StandardNamesVersion	Character	*standard-names-version*
VendorName	Character	none
ServerProductName	Character	none
ServerProductVersion	Character	none
OperatorName	Character	none
RoleName	Character	none
SupportContactInformation	Character	none

The user tokens contain basic information about the user that is stored on the server. If a server does not support one of these fields then it MUST set the returned value to empty (a zero-length string).

`user-id`	*`::= 1*30ALPHANUM`*
`user-class`	*`::= 1*30ALPHANUM`*
`user-level`	*`::= 1*5DIGIT`*
`agent-code`	*`::= 1*30ALPHANUM`*

The agent-code is the code that is stored in the property records for the listing agent, selling agent, etc. In some implementations this may be the same as the user-id. The fields user-class and user-level are implementation dependent and may not exist on some systems, in which case, an empty string should be returned. These parameters are used in the validation routines of the Update transaction (see Section 10 for more information).

broker-code	::= 1*24ALPHANUM
broker-branch	::= 1*24ALPHANUM

The *broker-code* and *broker-branch* parameters are used in the validation routines of the Update transaction (see Section 10 for more information).

member-name	::= 1*48TEXT

The *member-name* is the member's full name (display name) as it is to appear on any printed output, for example "Jane T. Row".

metadata-version	::= 1*2DIGITS . 1*2DIGITS [. 1*5DIGITS]

The *metadata-version* is the most current version of the metadata that is available on the server. It uses a "<major>.<minor>.<release>" numbering scheme. The version is advisory and is not used by the metadata currency scheme.

metadata-timestamp	::= RETSDATETIME

The *metadata-timestamp* is the time stamp associated with the current version of metadata on the host. If the client has cached an earlier version of metadata, it SHOULD take whatever action is necessary to load the current version of metadata.

min-metadata-timestamp	::= RETSDATETIME

The *min-metadata-timestamp* is the earliest version of the metadata that the host will support. If the version of the metadata being used by the client has a time stamp earlier than this time the client SHOULD retrieve the newer metadata from the host. In any case, the client MUST NOT send transactions using metadata older than *min-metadata-timestamp*.

The reasoning behind the definition of the minimum version of the metadata is to permit clients to ignore non-essential changes to components such as help text and user-readable descriptions.

metadata-id	::= 1*128(IDALPHANUM)

The *metadata-id* is a persistent ID associated with the metdata applied to the current user session. The *metadata-id* advertised by the server MUST match the Metadata ID attribute defined in Section 11.2.1 System Metadata. This requirement explicitly binds the metadata advertised by the Login Transaction response to the metadata advertised by the GetMetadata Transaction response. This relationship is necessary to eliminate confusion and assist metadata updates. If, following a Login response, the Metadata ID does not match the Login response *metadata-id* value, the client MUST update the client metadata using the GetMetadata Transaction and recognize that the metadata for the current session has changed.

------

balance	::= 1*32ALPHANUM

If the server supports an active billing account then the *balance* value SHOULD represent a user-readable indication of the money balance in the account.

timeout	::= 1*5DIGIT CRLF

The *timeout* is the number of seconds after a transaction that a session will remain alive, after which the server will terminate the session automatically (e.g. invalidate the session-id). This is commonly referred to as the inactivity timeout. A server need not provide this capability; however, if it does use session timeouts in order to prevent monopolization of resources, it MUST inform the client of the timeout interval by returning this response field.

pwd-expire-date	::= RETSDATETIME
pwd-expire-warn	::= ["-"]1*3DIGIT

The *pwd-expire-date* value is the date that the current user password becomes invalid. The *pwd-expire-warn* value is the number of days before the expiration date that the user should be warned of the upcoming password expiration. A *pwd-expire-warn* value of "-1" indicates that the password expiration is disabled.

standard-names-version	::= 1*128TEXT CRLF

The *standard-names-version* indicates the date version of StandardNames that this system supports. A system is only expected to support a single version of the StandardNames and, in most cases, this will be the current version.

Server systems that do not provide this optional field make no representation about the version of StandardNames that they support, therefore, client applications should not assume any specific version of the StandardNames.

Server systems that do provide this optional field **MUST** return a value for the standard-names-version that matches one of the values from the Adopted StandardNames List from Real Estate Transaction Standard website.

The format of the *standard-names-version* is a string *YYYY-MM* where *YYYY* is the year of adoption and *MM* is the month of adoption. For example, a version of the StandardNames is *2010-04*

VendorName is the name of the product vendor. It is required.

ServerProductName is the name of the server product provided by the vendor. It is required.

ServerProductVersion is the version of the server product. It is required.

OperatorName is the name of the MLS or Association operating the system. It is required.

RoleName is the name of the role restriction where the metadata may be restricted. It is optional.

SupportContactInformation is free text that provides a contact email, phone or website for development support. It is optional.

Vendors may provide additional *info-token-name* Session Information Tokens to meet local business needs. Clients MUST ignore Session Information Tokens that they do not understand.

More well-known Session Information Tokens may be added in later version of this document.

> **Note: An Approved RCP is Related to this Section**
> Section 4.7.1 is related to the following approved RCP(s):
>
> RETS 1.8.0
>
> - RCP 65 Session information tokens
> - RCP 70 Metadata Role Support
> - RCP 91 StandardNames Version Information in Login Transaction Role Support
> - RCP 98 Additional Information Fields in METADATA-SYSTEM and Login
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

4.7.2 Capability URL List

`capability-url-list`	`::=` see Section 4.10 for format information

The server MUST return a capability list that includes at least Search, Login and GetMetadata. The server MAY in addition return any of the other types in Section 4.10 . If the server supports any of the additional functions (and the client is entitled to access the function by virtue of the supplied login information), it MUST provide URLs for those functions. The server MAY supply URLs in addition to those in Section 4.10 based on the user-agent. If it does, it MUST follow the format specified in Section 4.10.

4.8 Optional Response Arguments

There are no optional response arguments. See Section 4.7.1 Session Information Tokens.

> **Note: RETS 1.8.0: An Approved RCP is Related to this Section**
> Section 4.8.3 is related to the following approved RCP(s):
>
> - RCP 65 Session information tokens
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

4.9 Well-Known Names

Some fields returned from the login are considered "Well-Known" and are used in the validation routines of the Update transaction. Those fields are as follows:

Table 4-2 Well-Known Names for Input Fields

Well-Known name	Input Return Field
`.USERID.`	user-id
`.USERCLASS.`	user-class
`.USERLEVEL.`	user-level
`.AGENTCODE.`	agent-code
`.BROKERCODE.`	broker-code
`.BROKERBRANCH.`	broker-branch

The client MUST assume a blank value for any well-known name for which the server does not supply an input field.

These values are used in Table 11-37, "Validation Expression Special Operand Tokens" .

4.10 Capability URL List

The capability-url-list is the set of functions or URLs to which the login grants access. A capability consists of a key and a URL. The list returned from the server in the login reply has the following format:

```
[Action = action-URL CRLF]
[ChangePassword = change-password-URL CRLF]
[GetObject = get-object-URL CRLF]
Login = login-URL CRLF
[LoginComplete = login-complete-URL CRLF]
[Logout = logout-URL CRLF]
Search = search-URL CRLF
GetMetadata = get-metadata-URL CRLF
[Update = update-URL CRLF]
[PostObject = postobject-URL CRLF]
[GetPayloadList = getpayloadlist-URL CRLF]
```

Table 4-3 Capability URL Descriptions

Parameter	Purpose
action-URL	A URL on which the client MUST perform a GET immediately after login. This might include a bulletin or the notification of email. The client application SHOULD provide a means for the user to view the retrieved document. A server is not required to supply an Action URL.
change-password-URL	A URL for the ChangePassword transaction.
get-metadata-URL	A URL for the Get Metadata transaction.
get-object-URL	A URL for the Get Object transaction.
login-URL	A URL for the Login Transaction. The client software should use this URL the next time it performs a Login. If this URL is different from the one currently stored by the client the client, MUST update the stored one to the new one. This provides a mechanism to move the Login server.
login-complete-URL	RESERVED
logout-URL	A URL for the Logout transaction.
search-URL	A URL for the Search transaction.
update-URL	A URL for the Update transaction.
postobject-URL	A URL for the PostObject transaction.
getpayloadlist-URL	A URL for the GetPayloadList transaction.

The URLs in the capability-url-list may be specified in any order. Since the list is returned in the body, servers MAY include whitespace between the parameter, equals sign and URL. Clients SHOULD be prepared to receive the capability-url-list either with or without whitespace in the response. The format of each URL follows the pattern defined in the *URL* atom. In addition, the table is extensible; servers may define additional transactions for clients to access. If a transaction is offered only to particular user agents, the keys for those additional transactions MUST begin with the user-agent token, followed by a dash "-", followed by an implementation-defined function name. Note that this definition does not permit spaces in the additional-transaction definition per the ABNF rules.

`additional-transaction`	*`::=`* `("X" |` *`user-agent-token`* `) "-"` *`function-name CRLF`*
`user-agent-token`	*`::=`* <token portion of the *`User-Agent`* (Section 3.3) >
`function-name`	*`::= 1*ALPHA`*

Example: `MLSWindows-special = /special_function`

Example: `X-Delete = http://www.example.com:6103/deletemyrecord`

A compliant client need not recognize any transaction that is not included in this specification. If some extended transactions are offered to any user-agent, the keys for those transactions MUST begin with an "X" followed by a dash, followed by an implementation-defined function name. Server implementers who implement potentially-unrestricted extension transactions are urged to register their keys and service descriptions on the RETS web site to encourage wider adoption.

URLs other than the Login URL may be relative URLs. The Login URL MUST be an absolute URL. If a URL is not absolute, the client application should canonicalize it according to the rules in RFC 2396, section 5. The "base URL" (as defined in RFC 2396, section 5.1.1) for this operation is the URL used for the *current* login transaction, not the new Login URL.

URLs MUST be URL-encoded per RFC 2396.

Note: RETS 1.8.0: An Approved RCP is Related to this Section
Section 4.10 is related to the following approved RCP(s):

- RCP 63 Object Data and Upload
- RCP 76 GetPayloadList
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

4.11 Reply Codes

Table 4-4 Login Transaction Reply Codes

Reply Code	Meaning
0	Operation successful
20003	Zero Balance The user has zero balance left in their account.
20004 thru 20011	RESERVED
20012	Broker Code Required The user belongs to multiple broker codes and one must be supplied as part of the login. The broker list is sent back to the client as part of the login response (see section 4.6).
20013	Broker Code Invalid The Broker Code sent by the client is not valid or not valid for the user
20014 thru 20019	RESERVED
20022	Additional login not permitted There is already a user logged in with this user name, and this server does not permit multiple logins.
20036	Miscellaneous server login error The quoted-string of the body-start-line contains text that SHOULD be displayed to the user
20037	User-agent authentication failed. The server requires the use of user-agent authentication (section 4.1.2), and the client either did not supply the correct user-agent password or did not properly compute its challenge response value.

20041	User-agent authentication required. The server requires the use of user-agent authentication (section 4.1.2), and the client did not supply the user-agent header values.
20050	Server Temporarily Disabled The server is temporarily offline. The user should try again later

NOTE: RETS does not require that a server maintain user accounts.

Section 5 - GetObject Transaction

To retrieve objects the client MAY first retrieve the metadata that describes the Resources and Objects that are available with the GetMetadata transaction described in section 12 . A full description of the Metadata Dictionary is provided in Section 11.

RETS does not require that any particular type of object be made available by a server. However, a server MUST use a standard well-known name under which to make its data available if a suitable well-known name is defined in the standard.

5.1 Required Client Request Header Fields

In addition to the Required Client Request Header Fields specified in Section 3.3 , the header of any messages sent from the client MUST contain the following header fields:

`Accept`	The client MUST request a media type using the standard HTTP Accept header field. Media-type formats (subtypes) are registered with the Internet Assigned Number Authority (IANA) and use a format outlined in RFC 2045 [8]. When submitting a request the client MUST specify the desired type and format. If the server is unable to provide the desired format it SHOULD return a "406 Not Acceptable" status. However, if there are no objects of any subtype available for the requested object the server SHOULD return "404 Not Found." The format of the Accept field is as follows:
Accept	`::=` **`Accept:`** *type* **/** *subtype* [**;** *parameter*] *(**,** *SP type* **/** *subtype* [**;** *parameter*])
type	`::=` * | <a publicly-defined type>
subtype	`::= *	<A publicly-defined extension token that has been registered with IANA>`
parameter	`::=` **q =**< qvalue scale from 0 to 1 >

A complete list of media types and subtypes is available at:

`http://www.iana.org/assignments/media-types/`

The qvalue is used to specify the desirability of a given media type/subtype, with "1" being the most desirable, "0" being the least desirable, and a range in between. The default qvalue is "1".

Example: `Accept: image/jpeg, image/tiff;q=0.5, image/gif;q=0.1`

Verbally, this would be interpreted as "image/jpeg is the preferred media type, but if that does not exist, then send the image/tiff entity, and if that does not exist, send the image/gif entity."

The types supported by the server are defined in the Metadata Dictionary as defined in section 11.4.1 .

5.2 Optional Client Request Header Fields

The GetObject transaction has no optional request header fields.

5.3 Required Request Arguments

`Resource`	A resource defined in the metadata dictionary (see Section 11.2.2)

The resource from which the object should be retrieved is specified by this entry. For more information see 5.9 . The resource MUST be a resource defined in the metadata (section 11.4.1).

`Type`	The object type as defined in the metadata (see section 11.4.1)

The grouping category to which the object belongs. Type MUST be an `ObjectType` defined in the Object metadata for this Resource. For more information see section 11.4.1 .

`ID`	*`UID`* `|` *`OBJECT_SET`* ; A string identifying the object or objects being requested

`UID`	`::= TOKEN *( "," TOKEN )`
`OBJECT_SET`	`::= resource-set *( "," resource-set)`
`resource-set`	`::= resource-entity [":" object-id-list ]`
`resource-entity`	`::= 1*ALPHANUM`
`object-id-list`	`::= "*" | object-id *( ":" object-id)`
`object-id`	`::= 1*5DIGIT`

The *`UID`* argument is a string identifying the object(s) being requested. It allows an object to be requested by their *`UID`* as described in Table 5-1. Either the *`OBJECT_SET`* or *`UID`* MUST be present in the request, but not both of them.

If the server does not support *`UID`* for the requested type of objects and the client submits a *`UID`* instead of the *`OBJECT_SET`*, the server MUST respond with an error. The preferred error code is 20403: No Object Found. Servers that do not implement the PostObject functionality (Section 13) MAY respond with a 20402: Invalid Identifier. If the requested type of object has an ObjectData class linked in the metadata, the server MUST support this argument.

For objects, the *`resource-entity`* is a value (e.g., MLS number, AgentID) from the `KeyField` of the Resource for which the object is to be retrieved.

The *`object-id`* is the particular object to be retrieved. Objects are assumed to be stored sequentially on the host beginning with an *`object-id`* of "1". If the *`object-id`* is 0 (zero or not provided), the designated preferred object of the given type is returned. If the *`object-id`* is set to "*" then all objects corresponding to the `resource-entity` are returned. This parameter can be used to specify the photo number, e.g. a value of "3" would indicate photo number 3.

If multiple *`resource-entity`* or *`object-id`* values are sent, or if any object-id-list is "*", then the host MUST respond with a multipart MIME [8] response. See 5.11, "Multipart Responses" , for more detail.

Note: An Approved RCP is Related to this Section
Section 5.3 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 63 Object Data and Upload

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.4 Optional Request Arguments

5.4.1 Location

`Location`	**`0`** `|` **`1`**

This parameter indicates whether the object or a URL to the object should be returned. This is used to provide access to the semi-permanent storage location of information for access outside of the transaction (e.g. for use in email to a customer). The lifetime of this semi-permanent storage is not defined by this specification.

If `Location` is set to "1" the server should return a URL to the given object. The default is "1". The server SHOULD support this functionality (`Location=1`) but MAY support `Location=0`. In other words, some servers may store the objects in a database or generate them dynamically. Therefore, it may not be possible for those servers to return a permanent URL to the requested object. In these cases the server MAY choose not to support `Location=1`. However, all servers MUST support a method to get the object and therefore, MUST support either the case where `Location=1` OR `Location=0`. Client vendors should be prepared to handle HTTP 410 responses when the URL expires or is removed and should not depend on the URL existing for all time.

When the Location=1, the message body SHOULD contain a RETS response as described in Section 3.5 .

> **Note: An Approved RCP is Related to this Section**
> Section 5.4.1 is related to the following approved RCP(s):
>
> RETS 1.8.0
>
> - RCP 102 GetObject URL as Default Location
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.4.2 ObjectData

`ObjectData`	`*`|*`FieldName *(,FieldName)`*
`FieldName`	::= *`RETSNAME`*

This parameter indicates that data relevant to the object should be sent as HTTP headers in the server response. If `ObjectData` is set to **`"*"`**, the server MUST include a header line for each field in the ObjectData class linked to the requested Object type, as described in Section 5.6. If `ObjectData` is set to a list of fields, ObjectData headers for the requested fields only MUST be sent in the response. If this argument is missing or is *`NULL`*, no ObjectData will be sent.

> **Note: An Approved RCP is Related to this Section**
> Section 5.4 is related to the following approved RCP(s):
>
> RETS 1.8.0
>
> - RCP 63 Object Data and Upload
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.5 Required Server Response Header Fields

In addition to the other Required Server Header Fields specified in Section 3.6 the following response header fields are required.

5.5.1 Content-Type

`Content-Type`	The media type of the underlying data. The server MUST return this field in all replies. Additionally, this field MUST be returned as part of the header for each body part. This field MUST be set to the type of media returned. See Section 5.1 for more information on <*`type`*> and <*`subtype`*>.

`Content-Type`	`::=` **`Content-Type:`** *`type /subtype`*

Example: `Content-Type: image/jpeg`

If the client has requested multiple IDs, the server MUST return a multipart message. If it does, it MUST return a Content-Type of "multipart/parallel" along with a boundary delimiter in the response header. See Section 5.11 for more information on multipart responses.

Example: `Content-Type: multipart/parallel; boundary=AAABBBCCC`

5.5.2 Content-ID

`Content-ID`	An ID for the object. This field MUST be returned as part of the header for each body part in a multipart response. A value for this field MUST be returned for each body part. This value is the resource-entity from the GetObject request and MUST match the corresponding Resource KeyField value.

`Content-ID`	`::=` **`Content-ID:`** *`1*128PLAINTEXT`*

Example: `Content-ID: 123456`

5.5.3 Object-ID

`Object-ID`	The object number being returned. This field MUST be returned as part of the header for each body part in a multipart response.

`Object-ID`	`::=` **`Object-ID:`** *`1*5DIGIT`* `|  "*"`

Example: `Object-ID: 2`

Note: The Object-ID may only have the value of "*****" in the response when there is an error in the response and the request was for all objects using the wildcard request of "*****".

5.5.4 MIME-Version

MIME-Version	All responses MUST include a MIME-Version of "1.0" in the response header.

Example: `MIME-Version: 1.0`

5.6 Optional Server Response Header Fields

In addition to the other Optional Server Header Fields specified in Section 3.7 the following response header fields are also optional.

5.6.1 Location

Location	If the client has submitted a request with "`Location=1`" the header of any non-error response MUST contain the Location header field. If the server does not support this functionality for a specific object, then "`Location:`" without a *`URI`* MUST be returned. If the server does not support this functionality for any object, the server should return an error type of 20414.

Location	`::=` **`Location:`** *`URI`*

Example: `Location: http://www.example.com/pic/123456.jpg`

If the server is returning a multipart response, then this header MUST be included in the MIME part headers for each object successfully requested.

5.6.2 Description

Description	A text description of the object.

`Description`	`::=` **`Content-Description:`** *`*1024<PlainTEXT, excluding CR/LF>`*

Example: `Content-Description: Front View`

If the object does not have a description or if the server does not support this feature, the header MAY not be returned. If the object has a description and the server is returning a multipart response, then this header MUST be included in the MIME part headers for the object.

5.6.3 Preferred

Preferred	If the requested object is determined by the server to be the preferred object for the requested record, the server MAY return the **Preferred:** header with a value of 1 (true). If the server does not support this functionality or if the requested object is not the preferred object, the server MUST return either **Preferred:** with a value of 0 (false) or omit the **Preferred:** header.

`Preferred`	`::=` **`Preferred:`** *`BOOLEAN`*

Example: **Preferred: 1**

If the server is returning a multipart response, this header MAY be included in the MIME part headers for each object it applies to and MUST NOT be included in the MIME part headers for objects it doesn't apply to.

If the client is sending a request with an object-id of 0, the server SHOULD only include a “Preferred” header if the server further supports identifying preferred or primary objects when not using an object-id of 0.

Note: An Approved RCP is Related to this Section
Section 5.6.3 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 79 Add Preferred Flag to GetObject Responses

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.6.4 UID

UID	The identifier of the object. This field is required and MUST be returned if the request used *UID* rather than the *ID* argument and MAY be sent if the *ID* argument has been used.

UID	`::=` **`UID:`** *`TOKEN`*

Note: An Approved RCP is Related to this Section
Section 5.6.4 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 63 - Object Data and Upload

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.6.5 ObjectData

ObjectData	If the client has submitted a request with ObjectData, the header of the response MUST contain the ObjectData header fields. The server MUST include a header line for each requested field in the ObjectData class linked to the requested Object Type. Each such header will have the name of "ObjectData" and its value will be the SystemName of the field followed by an equals sign, followed by the value of the field.

ObjectData	`::=` **`ObjectData:`** *`RETSNAME`*`=`*`*TEXT`*

Note that the value *RETSNAME* is one of the metadata defined system field names and *TEXT* will be constrained to that defined in the metadata for the particular field. See Table 11-15 for the definition.

Example:
ObjectData: PropMediaCaption=caption for kitchen
ObjectData: PropMediaDescription=details about kitchen

Note: An Approved RCP is Related to this Section
Section 5.6.5 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 63 - Object Data and Upload

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.6.6 Sub-Description

Sub-Description	A secondary description of the object.

Sub-Description	`::=` **`Content-Sub-Description:`** *`*1024<PlainTEXT, excluding CR/LF>`*

Example: `Content-Sub-Description: Enjoy the evening sunsets from the front porch`

If the object does not have a sub-description or if the server does not support this feature, the header MAY not be returned. If the object has a sub-description and the server is returning a multipart response, then this header MUST be included in the MIME part headers for the object.

Note: An Approved RCP is Related to this Section
Section 5.6.5 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 93 - Add Content-Sub-Description to GetObject
  Content in this section has been updated or modified since the previous RETS version.
  Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.6.7 RETS-Error

RETS-Error	If a server is unable to deliver a requested object which generates an error, "RETS-Error" MUST be included and have a value of 1 (true). Otherwise, RETS-Error MUST NOT be included as a header.

RETS-Error	`::=` **`RETS-Error:`** *`BOOLEAN`*

Example: `RETS-Error: 1`

If the server is returning a multipart response, this header MUST be included in the MIME part headers for each object it applies to and MUST NOT be included in the MIME part headers for objects it doesn't apply to.

See Section 5.11.2 Error Handling for examples of error responses.

5.7 Required Response Arguments

There are no required response arguments.

5.8 Optional Response Arguments

There are no optional response arguments.

5.9 Multipart Responses

As described in Section 5.3 , in the case where the client has requested multiple resource-entity or object-id values or if any object-id-list is "*", the server MUST return a multipart response. In the case of multipart responses, in which one or more different sets of data are combined in a single body, a "multipart" media type field must appear in the entity's header.

5.9.1 General Construction

RFC 2045 describes the format of an Internet message body containing a MIME message. The body contains one or more body parts, each preceded by a boundary delimiter line, and the last one followed by a closing boundary delimiter line. After its boundary delimiter line, each body part then consists of a header area, two blank lines, and a body area.

Example:

```
HTTP/1.1 200 OK
Server: Apache/2.0.13
Date: Fri, 18 APR 2014 12:03:38 GMT
Cache-Control: private
X-RETS-Version: RETS/1.9.0
MIME-Version: 1.0
Content-type: multipart/parallel; boundary="simple boundary"


--simple boundary
```

```
Content-Type: image/jpeg
Content-ID: 123456
Object-ID: 1

<binary data>
--simple boundary
Content-Type: image/jpeg
Content-ID: 123457
Object-ID: 1

<binary data>
--simple boundary--
```

Note: An Approved RCP is Related to this Section
Section 5.9.1 is related to the following approved RCP(s):

- RCP 71 Time Zone Data
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.9.2 Error Handling

When a client requests multiple objects in a single transaction, one or more of those objects may be unavailable. In this case, the server communicates the failure by including a RETS return message in place of the unavailable object. In this case, the Content-Type will be text/xml, and the content will be a RETS response:
Example:

```
HTTP/1.1 200 OK
Server: Apache/2.0.13
Date: Fri, 18 APR 2014 12:03:38 GMT
Cache-Control: private
X-RETS-Version: RETS/1.9.0
MIME-Version: 1.0
Content-type: multipart/parallel; boundary="simple boundary"


--simple boundary
Content-Type: image/jpeg
Content-ID: 123456
Object-ID: 1

<binary data>
--simple boundary
Content-Type: text/xml
Content-ID: 123457
Object-ID: 1

<RETS ReplyCode="20403" ReplyText="There is no listing with that ListingID"/>

--simple boundary--
```

If the server is supplying an error message for a wild-card object request (`Object-ID` of *), the `Object-ID` for the error part SHOULD be * as well.

Note: An Approved RCP is Related to this Section
Section 5.9.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 94 - Improved Error Handling in GetObject

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Note: RETS 1.9.0: An approved RCP is related to this section
Section 5.9 is related to the following approved RCP(s):

- RCP 108 HTTP User Space Headers

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.10 ObjectData Classes

The server MAY expose additional data for Objects. If it does, data MUST be exposed in a Class linked to an Object type via the ObjectData metadata field in the Object metadata. Any Table linked this way to the Objects MUST hold exactly one record for each Object file available through the GetObject transaction on the linked Resource and ObjectType. The data must correctly describe the Object. For example, if a FileSize field is exposed, the value MUST be the length of the file sent by the GetObject transaction.

The standard does not specify how a server exposes this information. One implementation might expose this information for all such tables within one Resource named OBJECT; however, the server is free to use any Resource and ClassName as it sees fit.

Any Class linked to an Object metadata item MAY provide additional information about the Object; however, if the data is one of the fields listed in Table 5-1, the Class MUST use the standard name~~s~~ for the data from the Table. Required fields of ObjectData are shown in **bold**.

Data from the ObjectData Table will be communicated via HTTP headers (see Section 5.4.2 and Section 13.1.2).

Table 5-1 ObjectData Content

Standard Name	DataType	Description
UID	CHARACTER	Unique ID. This field must be unique within the class and its resource.
ObjectType	CHARACTER	ObjectType of this object. This is the Type parameter in the GetObject request.
ResourceName	CHARACTER	Standard name of the resource this object belongs to. This is the Resource parameter in the GetObject request.
ResourceID	CHARACTER	Value of the Key Field identifying a record within the resource described by ResourceName. This is the first part of the ID parameter in the GetObject request.
ObjectID	INT	Ordinal number of this object within all objects belonging to the record identified by ResourceName and ResourceID. This is the ObjectID in the GetObject request.
MimeType	CHARACTER	MimeType of the object
IsDefault	*BOOLEAN*	1 if this object is the default one (sent when an object with ObjectId= 0 was requested). This is the main object that should be displayed for the ObjectType.
ObjectModificationTimestamp	*RETSDATETIME*	Time of the last modification of the object
ModificationTimestamp	*RETSDATETIME*	Time of the last modification of this data record (including the object modification)
OrderHint	INT	Provides an override for the ObjectId value so a client can specify an alternate ordering of ObjectData while preserving the one specified by the ObjectIds. This allows clients performing updates using the UID field perform operations such as resource reordering more easily. Unlike the ObjectId, where ordinal values are defined by starting with the number 1 and using the formula *n+1* to arrive at each consecutive term in the sequence, the OrderHint values can form any integer set provided that sequence of items within the set is preserved. If A and B are two objects in a collection then the following expression must be true: If ObjectA.OrderHint > ObjectB.OrderHint then ObjectA.ObjectId > ObjectB.ObjectId If the OrderHint is supported, then the server must maintain the number as sent by the client.
Description	CHARACTER	Description of the object.
Caption	CHARACTER	Short title of the object.
FileSize	INT	Size, in bytes, of the object
WidthPix	INT	Width of an image, in pixels
HeightPix	INT	Height of an image, in pixels

Duration	INT	Length of a movie or audio, in seconds
WidthInch	INT	Width of an image, in inches
HeightInch	INT	Height of an image, in inches

Searching for objects: ObjectData classes are searchable like any other class. The object data may be searched by any searchable field specified in the class. The ResourceName, ResourceID and Order or UID of the matching results may be used as parameters in a subsequent GetObject transaction requesting the matching objects.

Updating object data: The server MAY permit updates for the ObjectData class. In this case, the data MUST stay consistent with the object files. For example, a server must not allow changing the WidthPix field unless it is able to crop or resize the underlying image file.

A server MUST NOT permit an ***Add*** UpdateAction on the Update Transaction for the ObjectData class, since objects will be added via the PostObject transaction.

The system MAY expose a ***Delete*** UpdateAction on the Update Transaction, which will have a similar effect as the PostObject Transaction with the UpdateAction=***Delete*** (namely, it will remove a record from the Object table and delete the object file).

The system MAY also expose a ***Clone*** UpdateAction on the PostObject Transaction to allow for reusing an existing object in more than one record.

A server MAY provide an ***Upload*** UpdateAction on the PostObject Transaction. Clients MUST NOT use this UpdateAction directly; if they do, the server MUST refuse it. This UpdateAction type is used to associate validation expressions and update help with the PostObject Transaction. If the ***Upload*** UpdateAction exists, both client and server SHOULD use it to check data sent via the PostObject transaction.

Clients are cautioned that the validation expressions for the ***Upload*** UpdateAction on the PostObject Transaction may place constraints on any field defined in the Object table, with the effect of providing additional acceptable file characteristics.

Note: An Approved RCP is Related to this Section
Section 5.10 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 63 Object Data and Upload

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

5.11 Reply Codes

Table 5-2 GetObject Reply Codes

Reply Code	Meaning
20400	Invalid Resource The request could not be understood due to an unknown resource.
20401	Invalid Type The request could not be understood due to an unknown object type for the resource.
20402	Invalid Identifier The identifier does not match the KeyField of any data in the resource.
20403	No Object Found No matching object was found to satisfy the request.
20406	Unsupported MIME type The server cannot return the object in any of the requested MIME types.
20407	Unauthorized Retrieval The object could not be retrieved because it requests an object to which the supplied login does not grant access.
20408	Resource Unavailable The requested resource is currently unavailable.
20409	Object Unavailable The requested object is currently unavailable.

20410	Request Too Large No further objects will be retrieved because a system limit was exceeded.
20411	TimeoutThe request timed out while executing
20412	Too many outstanding requests The user has too many outstanding requests and new requests will not be accepted at this time.
20413	Miscellaneous error The server encountered an internal error.
20414	URL Location Not Supported The server does not support retrieving Objects of this type by URL.
20415	Objects in the response body, Location=0 Not Supported The server does not support retrieving Objects in the response body.

Note: An Approved RCP is Related to this Section
Section 5.11 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 102 GetObject URL as Default Location

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Section 6 - Logout Transaction

The Logout transaction terminates a session. Except in cases where connection failure prevents it or the user has requested an immediate shutdown of the client, the client SHOULD send the Logout transaction. If the client sends a Logout transaction, the server MUST attempt to send a response before terminating the session.

The server MAY send accounting information back to the client in the response to this transaction. The client is not required to display or otherwise process the accounting information.

There are no required or optional request arguments. There are no required response arguments.

6.1 Optional Response Arguments

ConnectTime	The amount of time that the client spent connected to the server, specified in seconds.

connect-time	::= **ConnectTime=**1*9DIGITS CRLF

`Billing`	If the server supports an active billing account, this is total amount billed for this session, specified as TEXT which is implementation-defined

billing	::= **Billing=***<TEXT, excluding CR/LF> CRLF

`SignOffMessage`	Any text. The client MAY display this message, if the server includes it in the response. Servers should not expect, however, that users would read or see the message, since communication failure may make it impossible for the client to receive the Logoff response.

sign-off-message	::= **SignOffMessage=***<TEXT, excluding CR/LF> CRLF

6.2 Logout Response Body Format

The Logout response body is a key/value response (see section 3.5, "Response Format").

```
<RETS 1*SP ReplyCode= quoted-reply-code 1*SP
ReplyText= quoted-string *SP >
[<RETS-RESPONSE>
[connect-time]
[billing]
[sign-off-message]
</RETS-RESPONSE>] [<RETS-STATUS [1*SP ReplyCode= quoted-end-reply-code 1*SP ReplyText=quoted-string *SP]/
>]
</RETS>
```

6.3 Reply Codes

Table 6-1 Logout Reply Codes

Reply Code	Meaning
0	Operation successful

20701	Not logged in The server did not detect an active login for the session in which the Logout transaction was submitted.
20702	Miscellaneous error. The transaction could not be completed. The ReplyText gives additional information.

Section 7 - Search Transaction

The Search transaction requests that the server search one or more searchable databases and return the list of qualifying records. The body of the response contains the records matching the query, presented in the requested format. The data can be returned in one of three formats: COMPACT, COMPACT-DECODED or STANDARD-XML.

- 7.1 Search Types
- 7.2 Search Terminology
- 7.3 Required Request Arguments
- 7.4 Optional Request Arguments
- 7.5 Search Response Body Format
- 7.6 Query language
- 7.7 Reply Codes

7.1 Search Types

Searches are performed on logical groupings of records called Resources. The definition of the grouping of records for a specific Resource is determined by the server implementation. Different server implementations may have different available Resource, depending on local rules, practices or conditions. Servers may further group the records by Class. Different users or different client applications may be provided with different sets of Resources and different sets of Classes. A specific value for Resource or Class is referred to in this document as a type. For example, a type of Resource is Property using the Standard Names definition. Another example may be a type of Resource called Appraisers, being a collection of locally licensed real estate property value appraisers. As defined below, a server only searches on a single Resource and Class per request. A server MAY provide more than one type of Resource in the metadata. The server MUST support searching at least one type of Resource. The types of Resource supported by the server MUST be specified in the metadata. Each of the Resource searches may by conducted against different databases or tables depending on the server implementation.

Some Resource are specified by well-known names. If a server implementation supports searches of any of these Resource, it MUST use the well-known Resource name to identify that Resource. The list of well-known Resource names is provided in Table 11-4, Well-Known Resource Names ; well-known Classes for those Resource are given in Table 11-10, Metadata Content: Resource Class .

StandardNames for Classes are given in Table 11-10, Metadata Content: Resource Class .

NOTE: RETS does not require that a server support any specific Resource type or Class. The user or maintainer of a server is responsible for deciding which Resource should be made searchable.

7.2 Search Terminology

7.2.1 Field Delimiter

A server may designate a particular OCTET to be used as a delimiter for separating entries in both the COLUMNS list and the DATA returned using the COMPACT and COMPACT-DECODED formats. The octet should be chosen to avoid the need to escape data within a record

field-delimiter	::= *HEX HEX*

7.2.2 Field Name

A field is the keyword or code that the server uses to identify a particular column in the database table. Each field may be either a System-Name, as defined in the metadata, or a Standard-Name, as defined in the Real Estate Transaction XML DTD. The server MUST accept either set of names interchangeably.

7.2.3 Record Count

This value indicates the number of records on the server matching the search criteria sent in the search query.

record-count	::= *1*9DIGITS*

Note that this value may be greater than the number of records returned, if the server has limited the size of the return for any reason.

7.2.4 Other terms

XML-data-record	::= <A data record as defined by the RETS Data XML DTD>.

7.3 Required Request Arguments

7.3.1 Search Type and Class

The SearchType and Class arguments specify the data that the server is to search.

SearchType	::= *RETSID*

The *SearchType* argument specifies the resource to search. It MUST be a *ResourceID*, as defined in Table 11-6 or a Well-Known Resource Name, Table 11-4 based on value of the StandardNames (Section 7.4.7) argument. It MUST be a resource existing in the metadata for this system. Not all system provide all Well-Known Resources.

Class	:: = *1*32ALPHANUM*

The *Class* argument specifies the class of the resource to search. It MUST be a *ClassName*, as defined in (section 11.3.1) Table 11-13 or a Well-Known Class Name, Table 11-12a based on the value of the StandardNames (Section 7.4.7). It MUST be a class existing in the metadata for this system. Not all systems provide all Well-Known Class Names.

If the resource represented by the *SearchType* has no classes, the Class parameter will be ignored by the server and MAY be omitted by the client. If the client does include the Class parameter for a classless search, the value SHOULD be the same as the *ResourceID* in order to insure forward compatibility.

Note that if StandardNames (Section 7.4.7) is set to 1, ~~both~~ the *SearchType* and *Class* arguments SHOULD be specified using the StandardName. If no StandardName exists for a specific Resource or Class, the server MUST accept the Resource and Class values by their SystemName even when StandardNames is set to 1. A server MAY reject a request with a 20203 error when StandardNames=1 and a SystemName is used where a StandardName exists in the metadata for that resource or class.

Note: An Approved RCP is Related to this Section
Section 7.3 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 99 Mixing StandardNames and SystemNames

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

7.4 Optional Request Arguments

7.4.1 Count

The Count argument controls whether the server's response includes a count.

Count	::= **0** | **1** | **2**

If the Count argument is not present or set to zero ("0") there is no record count returned. If this argument is set to one ("1"), then a record-count is returned in the response in addition to the data, all matches are counted regardless of any Offset or Limit parameter. Note that on some servers this will cause the search to take longer since the count must be returned before any records are received. If this entry is set to two ("2") then only a record-count is returned; no data is returned, but all matches are counted regardless of any Offset or Limit parameter.

Example: `Count=2`

Instructs the server to return only a count of the records matching the query.

7.4.2 Format

The Format argument selects one of the supported data return formats for the query response.

Format	::= **COMPACT** | **COMPACT-DECODED** | **STANDARD-XML** | **STANDARD-XML**:*dtd-version* | **JSON** | **JSON:**dd-version | **JSON-DECODED** | **JSON-DECODED:**dd-version | *payload-name* | *payload-name":"payload-version*
dtd-version	1*64*PLAINTEXT*
dd-version	1*64*PLAINTEXT*
payload-name	RETSNAME

payload-version	1*64PLAINTEXT

"COMPACT" means a field list <COLUMNS> followed by a delimited set of the data fields <DATA>.

"COMPACT-DECODED" is the same as COMPACT except the data for any field with an interpretation of Lookup, LookupMulti, LookupBitString or LookupBitMask, is returned in a fully-decoded format using the LongValue. See Section 13 for more information on the COMPACT formats and section 11.4.3 for more information on the Lookup types.

"STANDARD-XML" means an XML presentation of the data in the format defined by the RETS Data XML DTD. Optionally, a dtd-version may be added to the literal "STANDARD-XML:" to indicate a specific version of the DTD.

"JSON" means a JSON presentation of the data in the format defined by the METADATA-TABLE information. This allows legacy systems to represent their metadata as JSON. The intent of the change is to allow the Data Dictionary to be used within existing RETS1 implementations. Optionally, where the implementation supports multiple versions of the data dictionary, a version may be added to the literal "JSON:" to indicate a specific version of the data dictionary.

A Server that supports Data Dictionary names does not have to support all valid Data Dictionary payload-name values for a specific payload-version.

Servers MUST support COMPACT, COMPACT-DECODED and Data Dictionary formats. A Server MAY support STANDARD-XML. A Server MAY support JSON, JSON-DECODED, Payloads, data dictionary versions or any valid combination of them. If the format is not specified, the server MUST return the Data Dictionary response body for the Resource/Class combination.

Example: `Format=COMPACT-DECODED`

If the client requests STANDARD-XML, it MAY also append a preferred DTD version. Where the server supports STANDARD-XML, the server MUST support the current version and SHOULD additionally support at least the prior version.

Example: `Format=STANDARD-XML:1.0`

If the client requests a payload or JSON from the Data Dictionary, if MAY also append a preferred version of the Data Dictionary. Where the server supports Data Dictionary names, the server MUST support the current version and SHOULD additionally support at least the prior version.

A Client MUST only request the Format argument, and optionally the Select argument OR the Payload argument.

A Server MUST return an error code 20216 Invalid Argument Combination, Format and Payload when a Client submits a request with both Format and Payload optional arguments.

A Server MUST return an error code 20219 Invalid FORMAT when a Client submits a request with an optional format that is not supported on this Server.

A Server MUST return an error code 20220 Requested Data Dictionary version unavailable when a Client submits a request with an optional format data dictionary version that is not supported on this Server.

Note: An Approved RCP is Related to this Section
Section 7.4.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 76 GetPayloadList
- RCP 100 Alternate Standard Names

RETS 1.9.0

- RCP 113 - Search Transaction JSON Format

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

7.4.3 Limit

The Limit argument requests the server to apply or suspend a limit on the number of records returned in the search.

Limit	::= **"NONE"** | *1*9DIGIT*

In general, the `Limit` argument operates without consideration of other factors like the settings in the system metadata or the fields selected in the Select argument. A special case when the Limit="NONE" is described below.

If this entry is set to a number greater than zero, the server MUST not return more than the specified number of records. If the request results in more matches than the server returns, the <MAXROWS> tag MUST be sent at the end of the data stream, regardless of any `Limit` parameter specified in the client request.

In general, if this entry is set to ("NONE") or is not present, the server SHOULD treat this as a request to suspend enforcement of any internal download limit. Servers that permit the suspension of the limit MUST disable both the <MAXROWS> tag and the return code 20208, Maximum Records Exceeded when responding to a `Limit=`"NONE" request. Servers that do not permit the suspension of the limit MUST apply the <MAXROWS> and return code 20208 in the cases where the query results in more rows than permitted. Client implementers should be aware that some server implementations might not honor the request to disable the limit or may restrict the request to the selection of certain fields as described below; the server operator's business rules take precedence over the request to waive the system download limit.

A server may only support the suspension of the limit for a certain scenario of requests. When a server has Classes with a HasKeyIndex value of TRUE in the Class Metadata the server MUST suspend enforcement of the download limit for such a Class when the `Limit=`"NONE" and the Select argument contains only field names that have the InKeyIndex value of TRUE in the Table Metadata. A server SHOULD support HasKeyIndex for each Class and MUST have the InKeyField value of TRUE for at least the KeyField of the Class when the HasKeyIndex is TRUE for that Class. A server MAY have more than one field with the InKeyField value of TRUE for any Class.

Any request that sets a numeric Limit disables support for unlimited key index results as described in section 7.4.5 Select.

Note: An Approved RCP is Related to this Section
Section 7.4.3 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 68 Search Has Key Index Support

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

7.4.4 Offset

The client may specify that a retrieval start at other than the first record in the set of records matching the query by specifying the Offset argument

`Offset`	*`::= 1*9DIGIT`*

This argument indicates to the server that it SHOULD start sending the data to the client beginning with the record number indicated, with a value of "1" indicating to start with the first record. This can be useful when requesting records in batches, however, client implementers should be aware that data on the server MAY change as they iterate through the batches and it is possible that some records may be missed or added. In other words, the server is not required to maintain a cursor to the data.

Any time an Offset argument is supplied, the resulting data SHOULD be returned in a consistent order based on an ordering of the KeyField of the Resource. This ordering should be applied to the entire data set and not just the returned data which may be less that the total number of records matching the criteria. It is a recommended practice that an ascending order be used as the ordering scheme when the KeyField value is a sequentially increasing unique identifier, however, servers MAY choose to implement some other ordering scheme. This practice will help to ensure subsequent requests will not contain duplicate records. Ascending order of the KeyField in this case will also provide assurance that newly added records will be more reliably contained in the final Offset record set.

Clients iterating over the entire record set on systems that implement this practices MUST provide Offset=1 in the first request to assist the server to order the results.

The offset value of '0' is not defined in this standard.

7.4.5 Select

By default, the server MUST return all fields accessible to the client. The client may select a subset of those fields by specifying the Select argument.

`Select`	*`::= field *( , field)`*

This parameter is used to set the fields that are returned by the query. If this entry is not present then all allowable fields for the search/class are returned. The server MAY return an error when there are unknown fields in the select list. The server MUST NOT return more fields than are specified in the Select argument when the client requests COMPACT or COMPACT-DECODED data. It MAY return fewer if some of the field names are invalid or if a requested field is unavailable to the user based on security or other restrictions.

If the requested Class advertises HasKeyIndex as True in the Class Metadata and the client only selects fields advertised with InKeyIndex as True in the Table Metadata, the Server MUST return all the matching records unless the Client has declared a Limit other than NONE.

A Client MUST only request the Select argument, and optionally the Format argument OR the Payload argument.

A Server MUST return an error code 20217 Invalid Argument Combination, Select and Payload when a Client submits a request with both Select and Payload optional arguments.

In requests where the `StandardNames` argument is set to 0 and values are given within the Select argument, the client and server MUST reference fields by their `SystemName`.

In requests where the `StandardNames` argument is set to 1 and values are given within the Select argument, the client SHOULD reference fields by their `StandardName` when StandardName labels exist but MAY send `SystemNames` for fields that do not have a `StandardName`. In the request, the server MUST interpret fields in the Select as the `StandardName` for the field. If no field exists with the provided `StandardName`, the server MUST attempt to interpret the field using the value as a `SystemName`. In the response, for appropriate Format values, the server MUST used the provided field name for the <COLUMNS> list. That is, if a client provided a `StandardName` for a field, then the server should return the <COLUMNS> list value using that `StandardName`. Where the client provided a `SystemName` for a field, then the server should return the <COLUMNS> list value using the `SystemName`.

Note: An Approved RCP is Related to this Section
Section 7.4.5 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 68 Search Has Key Index Support
- RCP 76 GetPayloadList
- RCP 99 - Mixing StandardNames and SystemNames

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

7.4.6 Restricted Indicator

In some instances, the server may withhold the values of selected fields on selected records. When business rules withhold the value but the field is returned as part of a response, a `RestrictedIndicator` MUST be used in place of the value.

RestrictedIndicator	*::= 1*9TOKENCHAR*

This entry indicates to the server that it MUST set the restriction indicator to the value specified by this tag. The default restricted indicator is a *`NULL`* value.

Example: `RestrictedIndicator` = `####`

This would mean that all fields that the user is not allowed to see within a record (e.g. ExpirationDate) are returned with a value of ####.

Note that if the client requests fields that the server would withhold for every record, the server MAY choose to omit the field from the list returned rather than use the RestrictedIndicator for every record.

7.4.7 StandardNames

Queries may use either standard names or system names in the query (Section 7.6). If the client chooses to use standard names, it MUST indicate this using the StandardNames argument.

StandardNames	::= **0 | 1 | DataDictionary**

If this argument is set to 0 (zero) or is not present, the field names passed in the search are the `SystemName`, as defined in Table 11-15 of the metadata.

If this argument is set to 1 (one), the client MUST reference fields by their `StandardName` as declared in the metadata for the default Standard-XML format when `StandardName` labels exist but MAY send `SystemName`.

If this argument is set to DataDictionary, the client SHOULD reference fields by their data dictionary name as declared by the metadata for the DataDictionary format when DataDictionary labels exist but MAY send SystemName.

In the response, the server MUST reference fields by their `StandardName` when `StandardName` labels exist and by their SystemNames where no StandardName label exists. The `StandardNames` designation applies to all names used in the `SearchType`, `Class`, `Query` and `Select` arguments.

Note: An Approved RCP is Related to this Section
Section 7.4.7 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 99 - Mixing StandardNames and SystemNames
- RCP 100 - Alternate Standard Names

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

7.4.8 Payload

The Client may request a specific XML format for the return set.

Payload	`::=< a valid RESO schema name as found in the GetPayloadList Transaction >`

This entry must match a schema name discovered using the GetPayloadList. Please note that there are restrictions between the use of this optional argument and the Format argument (Section 7.4.2) and the Select argument (Section 7.4.5).

A Client MUST only request the Payload argument, OR the Format and optionally, the Select arguments.

A Server MUST return an error code 20216 Invalid Argument Combination, Format and Payload when the Client submits a request with both the Format and Payload optional arguments.

A Server MUST return an error code 20217 Invalid Argument Combination, Select and Payload when a Client submits a request with both the Select and Payload optional arguments.

A Server MUST return an error code 20218 Invalid Argument Combination when the Client submits a request with the Format, Select and Payload optional arguments.

Note: An Approved RCP is Related to this Section
Section 7.4.8 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 76 GetPayloadList

Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

7.4.9 Query

Query	`::=` <The query to be executed by the server>

The query is specified by the language denoted in the `QueryType` parameter. For DMQL2, the language is described in (Section 7.6) .

Clients are not required to provide a Query parameter unless one or more fields in the requested Class are marked as Required (see Metadata Content - Tables) in which case an error type of 20203 MUST be returned if the server is refusing to accept this request. If the server accepts this request, it MUST interpret the absence of a Query parameter as a request by the client to forfeit its option of filtering records past the filters that may be automatically applied by the server.

In requests where the `StandardNames` argument is set to 0 and values are given within the Select argument, the client and server MUST reference the fields of the query by their `SystemName`.

In requests where the `StandardNames` argument is set to 1 and values are given within the Query argument, the client MUST reference fields by their `StandardName` when StandardName labels exist but MAY send `SystemNames` for fields that do not have a `StandardName`. The server MUST interpret fields in the `Query` as the `StandardName` for the field. If no field exists with the provided `StandardName`, the server MUST attempt to interpret the field using the value as a `SystemName`.

Note: An Approved RCP is Related to this Section
Section 7.4.9 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 80 Optional Query
- RCP 99 - Mixing StandardNames and SystemNames

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

7.4.10 QueryType

The QueryType designates the query language used in the Query

QueryType	::= **DMQL2**

An enumeration giving the language in which the query is presented. The only valid value for RETS 1.8 is "DMQL2" which indicates the query language described in (Section 7.6). In the future, other query languages may be permitted.

Clients MUST provide the QueryType parameter if the Query parameter is sent.

7.5 Search Response Body Format

NOTE

> RETS 1.8 requires all server responses to be well-formed XML, and additionally requires search transaction responses to be valid XML. In addition, RETS requires that clients parse server responses as XML, not as simple text streams. The response formats shown here are normative with respect to content, but not normative with respect to form. That is, servers are free to produce response XML in any format that complies with the W3C XML 1.0 recommendation, so long as it is valid with respect to the appropriate DTD. So, for example, when the response format below calls out an empty XML tag, either the abbreviated tag format (**<MAXROWS/>**) or the full format (**<MAXROWS></MAXROWS>**) may be sent by the server and should be interpreted appropriately by the client. In addition, XML escaping of content is implied. See the W3C *XML Recommendation 1.0, Third Edition,* for full information on XML.

The body of the search response has the following format when replying to a request with the Format set to "COMPACT" or "COMPACT-DECODED":

```
<RETS1*SP ReplyCode= quoted-reply-code 1*SP
ReplyText= quoted-string *SP >
[count-tag ]
[delimiter-tag ]
[column-tag ]
*( compact-data )
[max-row-tag ]
[<RETS-STATUS [1*SP ReplyCode= quoted-end-reply-code 1*SP
ReplyText= quoted-string *SP]/>]
</RETS> CRLF
```

The body of the search response has the following format when replying to a Format request of "STANDARD-XML" data:

```
<?xml version="1.0" ?>
[doctype]
<RETS 1*SP ReplyCode= quoted-reply-code 1*SP
ReplyText= quoted-string *SP >
[* count-tag ]
*(XML-data-record)
[ max-row-tag ]
[<RETS-STATUS [1*SP ReplyCode= quoted-end-reply-code 1*SP
ReplyText= quoted-string *SP]/>]
</RETS> CRLF
```

The body of the search response has the following format when replying to a Payload request:

```
<?xml version="1.0" ?>
<RETS 1*SP name-space 1*SP xsi-value 1*SP schema-location 1*SP ReplyCode= quoted-reply-code 1*SP
ReplyText= quoted-string *SP >
[ count-tag ]
*( RESO-data-record )
[ max-row-tag ]
[<RETS-STATUS [1*SP ReplyCode= quoted-end-reply-code 1*SP
ReplyText= quoted-string *SP]/>]
</RETS> CRLF
```

doctype	::= **<!DOCTYPE RETS PUBLIC "-//RETS//DTD RETS XML Search Response 1.9.0//EN"SPC "*dtd-version*">**

dtd-version	::= <Name of the RETS DTD used to produce this document> example: **http://www.rets.org/dtd/2010/09/RETS-20100926.dtd**
RESO-data-record	**:=** <A record structure as defined in the rets-xml-search-response-1_9_0.xsd document>

When the client requests the STANDARD-XML representation, it may also specify a DTD version. The server MUST support the current version and SHOULD support the previous version. Data DTD versions are of the form

`RETS-`*`yyyymmdd`*`.dtd` where *`yyyymmdd`* is the release date of the DTD.

compact-data	::= **<DATA>** *field-delimiter* *(*field-data field-delimiter*) **</DATA>**

If a "COMPACT" or "COMPACT-DECODED" format is specified in the request then a "<DATA>" tag, a delimited list of field-data and a "</DATA>" end tag are returned to the client for each record returned. The field-delimiter is determined by the delimiter-tag.

count-tag	::= **<COUNT** *1*SP* **Records="***record-count***"** *1*SP* **/>**

When the client application specifies that a count should be returned (count-type = "1" \| "2") a count-tag MUST be sent by the server in the response. The "<COUNT>" tag MUST be on the first line following the reply-code line. The record-count value indicates the number of records on the server matching the search criteria sent in the search query.

column-tag	::= **<COLUMNS>** *field-delimiter* 1*(*field field-delimiter*) **</COLUMNS>**

If a “COMPACT” or “COMPACT-DECODED” format is specified in the request then a “<COLUMNS>” tag is also included containing a delimited list of the names of all of the fields being returned. If the StandardNames argument was set to 0 (zero) or not provided, these fields MUST be the SystemName label for every field returned. If the StandardNames argument was set to 1, these fields MUST reference the StandardName label where they exist and the SystemName label when no StandardName label exists. A server MUST NOT return the SystemName for a field that has a StandardName label.

The field-delimiter is determined by the delimiter-tag.

delimiter-tag	::= **<DELIMITER value ="** *field-delimiter* **"/>**

This parameter tells the client which character (OCTET) is used as a delimiter for both the COLUMNS list and the DATA returned. The server MUST send this parameter for "COMPACT" or "COMPACT-DECODED" formats. The "<DELIMITER>" tag MUST precede column-tag.

max-row-tag	::= **<MAXROWS/>** | **<MAXROWS></MAXROWS>**

A tag that indicates the maximum number of records allowed to be returned by the server has been exceeded, or alternatively, the Limit number passed by the client in the request has been exceeded.

> **Note: An Approved RCP is Related to this Section**
> Section 7.5 is related to the following RCP(s):
>
> RETS 1.8.0
>
> - RCP 92 RESO Payload Transport-Level Metadata Support
> - RCP 99 - Mixing StandardNames and SystemNames
>
> Content in this section has been updated or modified since the previous RETS version.
> Click the link above to review this RCP and the associated changes that are proposed for this version.

7.6 Query language

The query takes the form indicated below. This is the actual search criteria passed to the server. The server parses this query and generates a server-compatible query based on the parameters passed in the query-list.

7.6.1 Query language BNF

search-condition	::= query-clause | (search-condition or query-clause)
query-clause	::= boolean-element | (query-clause and boolean-element)
boolean-element	:: = [not] query-element
query-element	::= field-criteria | (**(** search-condition **)**)
or	::= **OR** | **|**
and	::= **AND** | **,**
not	::= **NOT** | **~**
field-criteria	::= **(** field**=**field-value **)**
field-value	::= lookup-list | string-list | range-list | period | number | string-literal | area | **.EMPTY.**
lookup-list	::= lookup-or | lookup-not | lookup-and |**.ANY.**
lookup-or	::= **|** lookup **(** ***,**lookup)
lookup-not	::= **~** lookup *(**,** lookup)
lookup-and	::= **+** lookup *(**,** lookup)
lookup	::= 1*128ALPHANUM | string-literal ;legal values are further constrained by the relevant lookup metadata
string-list	::= string *(, string)
string	::= string-eq | string-start | string-contains | string-char
string-eq	::= 1*ALPHANUM
string-start	::= 1*ALPHANUM *
string-contains	::= * 1*ALPHANUM *
string-char	::= *ALPHANUM *(**?** *ALPHANUM)
string-literal	::= **"**<PLAINTEXT except **"**> *(2**"** *<PLAINTEXT except **"**>) **"**
range-list	::= range *(**,** range)
range	::= between | greater | less}}
between	::= (period | number) **"-"** (period | number)
greater	::= (period | number | string-eq) **+**
less	::= (period | number | string-eq) **-**
period	::= (dmqldate | dmqldatetime | partial-time)
number	::= [**-**]1*DIGIT [**.** *DIGIT]
area	::= circle-by-radius | circle-by-points | rectangle | polygon
circle-by-radius	::= map-point “:“ number [length-unit]
circle-by-points	::= map-point “:” map-point
rectangle	::= map-point “, ” map-point
polygon	::= map-point 2*(“, ” map-point)

`map-point`	`::= number “ ” number`
`length-unit`	`::= “M” | “K”`
`dmqldate`	::= *`full-date`* `|` **`TODAY`**
`dmqldatetime`	::= *`RETSDATETIME`* `|` **`NOW`**
`dmqltime`	::= *`partial-time`*

An ***area*** value represents the area that contains ***map-point*** values that match the search criteria. The area may be shaped as a circle or a polygon.

A circular area ***circle-by-radius***, may be described by its centre ***map-point*** and the radius of the circle. An optional radius ***length-unit*** may be included to indicate the distance units. The default radius ***length-unit*** is miles (M). The radius length may also be measured in kilometers (K).

A circular area ***circle-by-points*** may also be described by its center and a single arbitrary point on the circle’s circumference. By convention, the first value is the centre point and the second value is a point on the circumference.

A ***polygonal*** area is described by a list of ***map-point*** values. Each *map-point* defines a vertex of the *polygon*. The edges of the *polygon* are described by two adjacent *map-point* values in the list of *map-points*. The *polygon* MUST be closed and convex, that is, each edge must form a chain, sharing a vertex with the previous edge. If the last point in the list is not the same as the first one, another side is defined by the first and last point in the list. Any two sides of the polygon MUST NOT intersect. A special case of a polygonal area is a ***rectangle***, which can be defined by just two points where the points MUST be opposite vertices of the *rectangle*.

Client and Server implementations SHOULD use Spherical Mercator for search criteria content and return results using WGS84 if possible.

Note: RETS 1.7.2: An Approved RCP is Related to this Section
Section 7.7.1 is related to the following approved RCP(s):

- RCP 71 Time Zone Data
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Note: RETS1.8.0: An Approved RCP is Related to this Section
Section 7.7.1 is related to the following approved RCP(s):

- RCP 69 LookupType Value
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

RETS1.9.0 An Approved RCP is related to this section
Section 7.6.1 is related to the following approved RCP(s):

- RCP 103 - Geospatial Search

Content in this section has been updated or modified sing the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

7.6.2 Query parameter interpretation

Query literal values are interpreted in the value space of the searched field. That is, the data type of the searched field determines the interpretation of the search literal values, which MUST be valid in that value space.

Dates and times submitted in a query MAY utilize time offsets relative to UTC using the *`dmqldatetime`* If a *`dmqldatetime`* is submitted with time offset information, the server system MUST interpret the request using the time offset information. If the time offset is not declared in the query, the server system MUST interpret the request using the default System time zone offset. This default must match the advertised time zone offset of the METADATA-SYSTEM SYSTEM TimeZoneOffset. If no time zone offset is advertised for the server system system, the default time zone offset MUST be UTC. The server system MUST interpret the **`TODAY`** token as the current date and time of the server system. For backward compatibility, the server system MUST treat clients with version less than 1.7.2 as submitting dates and times using a time zone offset of UTC/GMT. In this case, the advertised time zone offset is ignored since the client is not expected to be aware of the time zone offset. The server system MUST interpret the token **`NOW`** as the current date and time of the server system.

In processing a literal string, a server MAY substitute a *`string-char`* expression (**?**s) for the range of characters that contain any non-ALPHANUM not supported by that server.

In processing decimal numbers, where rounding is necessary, a server SHOULD round down for the bottom of ranges or values less than 0.5 and round up for the tops of ranges or values 0.5 or greater.

There are four types of field values that can be passed in the query string. They are a *`lookup-list`*, a *`range`*, a *`string`* and the special token **`.EMPTY.`**. A *`lookup-list`* is a field that may only contain predefined values, or the special token **`.ANY.`**, indicating that any value is acceptable. "Status" and "Type" are typical examples of fields with a limited range of predefined values.

The **`.ANY.`** token, if used, is to be interpreted exactly as if it contained all possible values for the given field. In particular, the use of **`.ANY.`** does not alter any limitation on the number of lookup values allowed for the field. It is merely a shorthand method of specifying all possible lookup values.

`range` fields can be searched based on a range of values. "ListPrice" and "ListDate" fall into this category. All values specified in a *`range`* are to be treated as inclusive (e.g. 2+ is the same as 2 or greater, inclusive of 2; 2-3 is the same as 2 to 3, inclusive of 2 and 3; 2- is the same as 2 or less, inclusive of 2). The types of the range endpoints MUST match the data type of the field being searched. In addition, the *`range-start`* value MUST be less than the *`range-end`* value in the value space defined by the searched field, or the result is undefined.

A *`string`* field is any other character field not falling into the other two categories. These are usually freeform text fields. An example of this kind of field is "OwnerName".

The special value **`.EMPTY.`** is to be interpreted as whatever the value of the field would be if no value had been entered. Note that this is implementation-defined: it may be the same as a search for a null value, or it may be blank or zero. A client should not expect to be able to distinguish unentered values from any other values using this search token.

Each *`field`* MUST be a SystemName, as defined in the metadata, when the StandardNames argument is set to 0 (or not given) in the request. When the StandardNames argument is set to 1, the client SHOULD reference the StandardName when StandardName labels exist but MAY reference SystemNames.

All values submitted for lookup-lists must be the Value in compact format, as defined in Section 13 .

The data types for field values may be determined by examining the metadata for the searched field. In a query using StandardNames, the RETS Data Dictionary gives the acceptable data type for search values.

Within range criteria, the datatype of the start and end range values MUST be identical. That is, no mixing of datatypes within a specific range is permitted.

If a client submits a *`lookup`* value containing non-alphanumeric characters, the client MUST use the *`string-literal`* representation of the Lookup value.

Note: An Approved RCP is Related to this Section
Section 7.6.2 is related to the following approved RCP(s):

- RCP 71 Time Zone Data
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.
- RCP 99 Mixing StandardNames and SystemNames

7.7.3 Sub-queries

This query language provides for a nesting of sub-queries. For example:

```
Query=((AREA=|1,2) | (CITY=ACTON)),(LP=200000+)
```

Example:

```
Query=(ST=|ACT,SOLD),
(LP=200000-350000),
(STR=RIVER*),
(STYLE=RANCH),
(EXT=WTRFRNT,DOCK),
(LDATE=2000-03-01),
(REM=FORECLOSE),
(TYPE=~CONDO,TWNHME),
(OWNER=P?LE)
```

Verbally, this would be interpreted as "return properties with (`ST` equal `ACT` or `SOLD`) and (`LP` between 200000 and 350000, inclusive) and (`STR` beginning with `RIVER`) and (`STYLE` equal `RANCH`) and (`EXT` equal `WTRFRNT` and `DOCK`) and (`LDATE` greater than or equal to 2000-03-01) and (`REM` containing `FORECLOSE`) and (`TYPE` not equal to `CONDO` and not equal to `TWNHME`) and (`OWNER` starting with `P` and having `LE` in the 3rd and 4th characters)."

7.7 Reply Codes

Table 7-1 Search Transaction Reply Codes

Reply Code	Meaning
0	Operation successful.
20200	Unknown Query Field The query could not be understood due to an unknown field name.
20201	No Records Found No matching records were found.
20202	Invalid Select The Select statement contains field names that are not recognized by the server.
20203	Miscellaneous Search Error The quoted-string of the body-start-line contains text that MAY be displayed to the user.
20206	Invalid Query Syntax The query could not be understood due to a syntax error.
20207	Unauthorized Query The query could not be executed because it refers to a field to which the supplied login does not grant access.
20208	Maximum Records Exceeded Operation successful, but all of the records have not been returned. This reply code indicates that the maximum records allowed to be returned by the server have been exceeded. Note: reaching/exceeding the "Limit" value in the client request is not a cause for the server to generate this error.
20209	Timeout The request timed out while executing
20210	Too many outstanding queries The user has too many outstanding queries and new queries will not be accepted at this time.
20211	Query too complex The query is too complex to be processed. For example, the query contains too many nesting levels or too many values for a lookup field.
20214	Unsupported Argument - Payload The server does not support the Payload argument.
20215	Invalid Payload The transaction is requesting an output Payload that does not match the Resource value.
20216	Invalid Argument Combination - Format and Payload The transaction uses both the Format and Payload arguments. This is not supported by the Standard.
20217	Invalid Argument Combination - Select and Payload The transaction uses both the Select and Payload arguments. This is not supported by the Standard.
20218	Invalid Argument Combination - Format, Select and Payload The transaction uses the Format, Select and Payload arguments. This is not supported by the Standard.
20219	Invalid FORMAT - The selected optional Format argument is not supported by this implementation.
20220	Requested Data Dictionary version unavailable - The selected optional Format argument data dictionary version is not supported by this implementation.
20514	Requested DTD version unavailable. The client has requested the metadata in STANDARD-XML format using a DTD version that the server cannot provide.

Note: An Approved RCP is Related to this Section

Section 7.8 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 76 GetPayloadList

RETS 1.9.0

- RCP 113 - Search Transaction Optional Format Argument - Add JSON

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Section 8 - Get Transaction

Gets an arbitrary file from the server or performs an arbitrary action, specified by URI. This is a standard HTTP GET, as described in the HTTP Specifications[2] . The file to GET is passed as part of the Request-URI.

RETS servers need not support the GET transaction to any greater extent than is necessary to implement the functionality of the Action URL (see 4.10, "Capability URL List"). If a RETS server does not intend to include an Action URL in its login responses, it need not support the GET transaction.

There are no required or optional arguments for either the request or response. Those sections have been removed.

See the General Status Codes in Section 3.9 for typical Status-Codes.

Section 9 - Change Password Transaction

The Change Password transaction provides a means for the user to change their password. The new password is appended to the username and encrypted using the Data Encryption Standard (DES), ANSI X3.92, using a hash of the old password as the key.

9.1 Required Request Arguments

```
PWD ::= PWD= <BASE64(<*DES(* Password : UserName )>)
```

This is the Base64 representation of the DES-encrypted UserName and Password. The new Password and the UserName are appended together with a colon (":") between and the resulting string is encrypted using DES in Electronic Code Book (ECB) mode. The DES key is constructed using the procedure in Section 9.6 . Base64 encoding is defined in RFC 2045 section 6.8.

9.2 Optional Request Arguments

There are no optional request arguments.

9.3 Required Response Arguments

There are no required response arguments.

9.4 Optional Response Arguments

There are no optional response arguments.

9.5 Reply Codes

Table 9-1 Change Password Reply Codes

Reply Code	Meaning
0	Operation successful.
20140	Insecure password. The password does not meet the site's rules for password security.
20141	Same as Previous Password. The new password is the same as the old one.
20142	The encrypted user name was invalid.

9.6 Encryption Key Construction

The new password is communicated to the host as a string encrypted with the Data Encryption Standard, ANSI X3.92. DES requires a 64-bit key, which is constructed as follows:

1. The old password and username are converted to uppercase and concatenated together.
2. The resulting string is hashed using MD5.
3. The key is taken as the first 64 bits of the resulting hash value. Parity bits must be corrected for encoders that check parity.

9.7 ECB Padding

The input to the DES ECB encryption process shall be padded to a multiple of 8 octets in the following manner:

Let *n* be the length in octets of the input. Pad the input by appending 8 – (*n* mod 8) octets to the end of the input, each having the value 8 – (*n* mod 8), the number of octets being added. In hexadecimal, the possible paddings are `0x01`, `0x0202`, `0x030303`, `0x04040404`, `0x0505050505`, `0x060606060606` and `0x07070707070707` and `0x0808080808080808`. All input is padded with 1 to 8 octets to produce an input string that is a multiple of 8 octets in length. The padding can be unambiguously removed after decryption.

This padding method is compatible with RFC 2315 section 10.3, note 2.

9.8 Effect of change

Servers that return a success status MUST accept the new password and reject the old password for all subsequent Login transactions and sessions. Servers that return a success status MAY require the use of the new password for all subsequent transactions in the current session by issuing a WWW-Authenticate challenge for transactions that do not contain the correct credentials.

If a client fails to receive a response to this transaction, it SHOULD retain both the old and new passwords until the effect of the Change Password transaction can be ascertained via a successful login.

Section 10 - Update Transaction

The update transaction is used to modify data on the server. The client transmits information describing the update to perform. The information is then validated by the server. Based on the metadata for a particular implementation, the information of the update may be structured as a parent-child relationship where each parent record may have several associated child records. If there are errors in the information or the associated child information, the server returns an error reply and no information is saved. That is, on an error in any of the information, parent-child or flat, the server MUST be returned to the state it was before the update transaction was attempted. If there are no errors, the record(s) as added or updated on the server will be returned. The record is returned in the same manner as a record is returned from a search.

Update requests MUST use the POST method (rather than the GET method). The request MUST use a content-type appropriate to the encoding of the request, per [16]. A content-type of `text/www-url-formencoded` is recommended, but any other method of encoding HTML form parameters may be used.

Note: An Approved RCP is Related to this Section
Section 10 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 101 Child Row Support

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

10.1 Required Request Arguments

10.1.1 Resource and ClassName

The Resource and Class arguments specify the RETS resource and class that the update is applied against.

Resource	::= *resource-name*
resource-name	::= *RETSID*

The name of the resource to be updated, as specified in the metadata. This is the `ResourceID` as defined in Section 11.2.2 .

ClassName	*class-name*
class-name	::= *RETSNAME*

The name of the class to be updated, as defined in the metadata. This is the `ClassName` as defined in section 11.3.1 .

10.1.2 Validate

Validate	*validate-flag*
validate-flag	::= **0** | **1** | **2**

If this entry is set to **0** and there are no errors in the record, the record in the server database is updated.

If this parameter is set to **1**, the server expects that the record is not complete and that it will not be stored on the server as a result of this UpdateAction. The partial record is validated by the host. Any fields that are not provided are not validated. Any fields with the metadata field "Attributes" set to "Autopop" in the metadata (see Section 11.3.4) will have their field values filled in by the server and returned to the client. The "Autopop" mode is used to automatically populate the fields of the data record. The record in the server database is not updated.

If this entry is set to **2**, the server validates all fields and returns any errors found, but does *not* store the record. The record in the server database is not updated.

10.1.3 Action

`Action`	::= *`update-action`*
`update-action`	::= *`1*24 ALPHANUM`*

The type of update to perform, as specified by the metadata. This is the `UpdateAction` as defined in Section 11.3.3.

10.1.4 Record

`Record`	*`::= field_name = field-value *(field-delimiter field_name = field-value)`*

The Record provides values for the fields that are being used in the Update Transaction. This includes those fields that are to be changed. For Action values like Clone, Change and Delete, the value for the `KeyField` (as defined in Section 11.3.3) MUST be provided in the Record, to identify the record on the server that is to be modified.

`field-name`	*`::= RETSNAME`*

The name of the field to be updated, as specified in the metadata. This is the `SystemName` as defined in Section 11.3.2 .

`field-delimiter`	*`::= OCTET`*

The octet that will separate the fields in the record. The delimiter is specified by the optional argument Delimiter (See Section 10.2). If this is not specified, an ASCII HT character (OCTET 09) is used as the default.

`field-value`	`::=` <varies depending on the field>

The text representation of the field value as defined by the metadata in Section 11.3.2 subject to the business rules. If the value is a Lookup type, the value MUST be submitted using the Value of the LookupType rather than the LongValue or ShortValue of the LookupType, formatted in the COMPACT data format.

> **Note: An Approved RCP is Related to this Section**
> Section 10.1 is related to the following approved RCP(s):
>
> RETS 1.8.0
>
> - RCP 59 Revised Update Transaction
>
> Content in this section has been updated or modified since the previous RETS version.
> Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

10.2 Optional Request Arguments

10.2.1 Delimiter

`Delimiter`	*`::= HEX HEX`*

The *Delimiter* specifies the octet that will separate fields in the record. If it is not specified, the default value of ASCII HT character (09) will be used.

10.2.2 Lock

`Lock`	*`::= lock-time`*
`lock-time`	*`::= 1*DIGIT`*

The *Lock* specifies a request to lock the requested record to prevent changes to the specific record by other clients. If the lock-time is a number greater than zero, the server SHOULD lock the requested record for the next *lock-time* seconds. If the record has already been locked by this user and the server receives another *Lock* request with the same *LockKey* value, the *lock-time* period is set to the new *lock-time* value seconds. A server MAY substitute a shorter *lock-time* period or ignore the lock request completely, based on server-specific rules. The requested *lock-time* is a hint from the client about how long it may take before the final update on the record will be requested.

If the *lock-time* argument is missing, the server will use its own discretion about locking the record.

A zero value for the *lock-time* request argument means that the client does not want to lock the record. The server will still use its own locking policy. For example, if the UpdateAction is BeginUpdate, the server MAY lock the record and the client MUST submit another Update Transaction to release the lock.

If the server supports *Lock*, the server should implement a lock expiration strategy. Network or other types of problems may prevent the release lock request from reaching the server, thus the server should be prepared to clean up an expired lock on a record.

10.2.3 LockKey

LockKey	::= *lock-key-value*
lock-key-value	::= TOKEN

If the server locks the requested record, it MAY return a value for *LockKey*. The *lock-key-value* MUST be sent back by the client in the next Update transaction on this record. This allows the server to verify that the record started with the values stored at the time of the lock.

If the client fails to submit the correct *lock-key-value*, the server MAY fail the Update request. In that case, the client has to submit another BeginUpdate action to lock the record and get the record values and a new *lock-key-value* then submit the desired Update request with the new *lock-key-value*.

10.2.4 Select

Select	::= *field-name *(, field-name)*

Specifies a list of fields that should be returned in the response. Server MUST return current (updated) values for all fields in this list.

The behavior of the Select is identical to that of Section 7.4.5. If this argument is omitted, all fields will be returned.

The field-names in the Select argument must be the System Names as defined in the Table Metadata (11.3.2).

10.2.5 WarningResponse

See Section 21.3 WarningBlock for details.

10.2.6 ChildRecord

If the Update metadata (11.3.3) describes Child_Action metadata (11.3.5) then the client can submit updated child data with the ChildRecord parameter in the same transaction.

ChildRecord	::= **ChildAction =** *child-action-id field-delimiter* **ChildRequestID =** *child-request-id field-delimiter* **Sequence =** *process-order field-delimiter* *field_name* **=** *field-value *(field-delimiter field_name* **=** *field-value)* ; **ChildAction=**123 **ChildRequestID=**3324 **Sequence=**2 ListPrice=250000 Status=1
child-action-id	::= RETSID
child-request-id	::= RETSID
process-order	::= POSITIVENUM
field-name	::= RETSNAME
field-delimiter	::= OCTET

field-value	`::=<varies depending on the field>`

The ChildRecord provides details for child rows of the class being updated. This MUST include the *child-action-id*, followed by a ChildRequestID, then a Sequence, and the fields that are to be change. Multiple ChildRecord parameters may be specified both for the same *child-action-id* and for different *child-action-ids* if the server describes multiple ChildActions for this UpdateAction. For example, in the case that a server supports three different types of ChildActions, a client can choose to update a room in one ChildRecord, remove a room in a second, and add an Open House with a third ChildRecord parameter in the same Update Transaction.

The *`child-action-id`* describes both the relationship between the parent row and this child row, and the UpdateAction to apply for this child row. This is the *child-action-id* as defined in section 11.3.5.

The *child-request-id* is a means for the client to identify a particular child row to the server. The server will reference this ID when communicating errors, warnings, and success back to the client. These ids SHOULD be unique within a request.

The *process-order* describes to the server in what order child rows are to be processed. The server MUST process child rows in ascending *process-order* order. The *process-order* values need not be contiguous. Coincident *process-order* values may lead to undetermined behaviour. For example, two child rows within the same Update Transaction with the same *process-order* . It is at the discretion of the server vendor to continue processing child rows after a data validation error has been encountered.

The *field-name* is the name of the field to be updated, as specified in the meta-data. This is the SystemName as defined in Section 11.3.2. The *field-delimiter* is the octet that separates the fields in the record. This is as defined for Delimiter in Section 10.2. The *field-value* is the text value of the field as defined by the metadata subject to the business rules.

10.2.7 ChildSelect

`ChildSelect`	`::=` **`ForeignKey =`** *`ForeignKeyID , field-name *(, field-name)`*

The ChildSelect specifies a list of fields that should be returned in the response. The server MUST return current (updated) values for all fields in this list. If this argument is omitted, no child sections will be returned. Multiple ChildSelect parameters may be specified for additional child table data. Each ChildSelect parameter MUST specify a different ForeignKeyID.

Note: An Approved RCP is Related to this Section
Section 10.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 59 Revised Update Transaction
- RCP 102 Child Row Support

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

10.3 Required Response Arguments

There are no required response arguments.

10.4 Optional Response Arguments

There are no optional response arguments.

10.5 Update Response Body Format

The body of the update response has the following format when there are no errors:

```
<RETS 1*SP ReplyCode= quoted-reply-code 1*SP
ReplyText= quoted-string *SP > CRLF
[ lock-tag ]
[ lock-key ]
[ delimiter-tag ]
column-tag
compact-data

* (child-data)
[<RETS-STATUS 1*SP ReplyCode= quoted-end-reply-code 1*SP
ReplyText= quoted-string *SP/>
</RETS> CRLF
```

The body of the update response has the following format when there are errors or warnings:

```
<RETS 1*SP ReplyCode= quoted-reply-code 1*SP
ReplyText= quoted-string *SP > CRLF
[ lock-tag ]
[ lock-key ]
[ delimiter-tag ]
column-tag
compact-data
[error-block]
[warning-block]
* (child-data)

 </RETS> CRLF
```

lock-tag	::= **Lock=***lock-time*
lock-time	::= *1*DIGIT*

The *lock-time* is the number of seconds that the record will be locked on the server. If *lock-time* is zero, the lock has been already released. Note that the *lock-time* MAY be less than requested, if the server uses any internal logic to limit the time for which a record will be locked.

A server that supports record locking MUST return the `lock-tag` in the response. If no lock was requested, a server that supports locking MUST return ***Lock*=0**. A missing `lock-tag` indicates that the server does not support record locking.

lock-key	::= **LockKey=***lock-key-value*
lock-key-value	::= *TOKEN*

If the server locks the requested record, it MAY return a *LockKey*. The *lock-key-value* MUST be sent back in the next Update Transaction to allow the server to verify that the client operates with the values the record had at the time it had been locked.

If the client fails to provide the correct *lock-key-value*, the server MAY respond with an error code to the Update request. In that case, the client has to request another BeginUpdate UpdateAction to lock the record and get the actual data, and then request the Update request with the new lock-key.

If a BeginUpdate UpdateAction was requested on a record that is already locked by the same user, *the lock-key-value*, if present, MUST be the same as when the record was originally locked. That allows the client to verify that the record has not changed since the original lock has been granted.

child-data	::= *open-child-data-tag [column-tag] *(child-row) close-child-data-tag*
open-child-data-tag	::= **<ChildData ForeignKey =** *QUOTE foreign-key-id QUOTE* **>**
foreign-key-id	::= *RETSID*
close-child-data-tag	::= **</ChildData>**

Optionally, child data rows may be included following the parent data row. Child data rows start with a tag and the foreign key. The *foreign-key-id* attribute links the relationship between the child rows in this section and the row in the primary table.

`child-row`	::= `open-child-row-tag compact-data [error-block] [warning-block] close-child-row-tag`
open-child-row-tag	::= **<ChildRow ChildRequestID =** *QUOTE child-request-id QUOTE* **>**
child-request-id	*::= RETSID*
close-child-row-tag	::= **</ChildRow>**

The *child-row* contains the updated child row along with related error and warning blocks. The *child-request-id* attribute is the ChildRequestID specified by the client in the ChildRecord request argument.

10.5.1 Error block

See Section 17.1

10.5.2 Warning block

See Section 17.2

Note: An Approved RCP is Related to this Section
Section 10.5 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 59 Revised Update Transaction
- RCP TBD Child Row Support

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

10.6 Record Locking

Clients are encouraged to use record locking to avoid conflicting changes by other parties. A typical procedure for modifying an existing record in an interactive client application would be as follows:

1. The Client requests an Update transaction with UpdateAction=BeginUpdate, the Record specifying only the key value of the requested record, and (optionally) requesting a Lock argument value with some time period to allow the user to edit the Record. The Server locks the requested record and returns the current data for the Record, along with (optionally) the Lock and LockKey arguments.
2. The Client presents the current data to the user and lets them make necessary changes. If the lock-time value returned by the Server will expire before the user is ready, the Client should request another Update transaction with UpdateAction=BeginUpdate, the Record specifying only the key value of the requested record, the LockKey with the lock-key-value sent by the server (if the server has sent one), and Lock value set to some additional period of time. The server should extend the lock on the Record and (optionally) send back Lock and LockKey arguments. If it does send LockKey, its value SHOULD be the same as that in the the original lock, to indicate to the client that nothing has changed since the original Lock request.
3. To update the Record and maintain the transaction for further update, the Client should request the Update Transaction with UpdateAction=Update, the complete Record with updated data, the LockKey with value provided by the server, and Lock value set to some additional period of time. The server should extend the lock on the Record. If the Server returns the LockKey, it should return the same lock-key-value as returned with the original lock, to indicate to the client that nothing has changed since the original Lock request.
4. To complete the Update of the Record, the Client requests an Update Transaction with the lock-key-value provided by the server and Lock=0. The Server updates the Record and releases the lock.

Clients should expect that the Lock has been released by the Server if the *`lock-time`* has expired.

A Client may send an UpdateAction=Update Transaction without first sending an UpdateAction=BeginUpdate Transaction. If a server allows for updating a unlocked record, then the Record is updated, the server sends back a response with a *`lock-time`* of zero (or missing), and that finishes the transaction. If the server does require the BeginUpdate action, it will return with error 20314. Note that the server MUST specify RequiresBegin=1 in the metadata if it requires the BeginUpdate action.

The server may choose to not implement the locking mechanism. If this is the case, the response to any BeginUpdate action will respond without the *`lock-tag`*, providing the requested data for the Record. The missing *`lock-tag`* tells the Client that locking is not in effect.

Note: An Approved RCP is Related to this Section
Section 10.6 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 59 Revised Update Transaction

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

10.7 Validation

Validation routines are indications of the checks the host system will perform against a field value before it is accepted for storage on the host. Some of these routines require data available only on the host system. However, others are relatively simple and could be performed by any RETS client to prevent invalid field values from being submitted. There are several different types of validation to be performed by the client.

A compliant client is not required to enforce the local validations provided in this section. However, if a client does not enforce the validations then the likelihood of the server rejecting the record is greatly increased.

10.7.1 Lookup

The entry is validated against a list of acceptable values. If the metadata described in Section 11.3.2 specifies the Interpretation as Lookup the only acceptable values for the field are defined in the `METADATA-LOOKUP` referenced by `LookupName`. Alternatively, if the metadata specifies a `ValidationLookup` the only acceptable values for the field are defined in the `METADATA-VALIDATION_LOOKUP` referenced by the ValidationLookup field.

10.7.2 MultiSelect Lookup

The entry is validated against a list of acceptable values. If the metadata described in Section 11.3.2 specifies the Interpretation as `LookupMulti`, the only acceptable values for the field are defined in the `METADATA-LOOKUP` referenced by `LookupName`. The maximum number of values that can be selected is defined by `MaxUpdate`.

10.7.3 Range

The entry must be between the `Minimum` and `Maximum` values specified in the metadata. The entry text length must not exceed the MaximumLength (see Section 11.3.2).

10.7.4 Test Expression

For each field being updated, evaluate all ValidationExpressions as defined in the metadata. See Section 11.4.9 for more information on Test Expressions.

10.7.5 External

The entry may be validated by searching a server resource. The Resource is defined for searching and the parameter list includes a set of suggested input fields, a set of result fields to display and a set of result fields to populate into the fields of the resource being updated. Information for external validation is provided in Section 11.4.10 .

10.8 Reply Codes

Table 10-1 Update Transaction Reply Codes

Reply Code	Meaning
0	Operation successful.
20301	Invalid parameter. Additional information is provided in the error block.
20302	Unable to save record on server.
20303	Miscellaneous Update Error.
20311	`WarningResponse` was not given for all warnings that contained a *`response-required`* value of 2.
20312	`WarningResponse` was given for a warning that contained a *`response-required`* value of 0.
20313	Incorrect LockKey. The record is locked, but the LockKey is not correct.
20314	Record Has Not Been Locked. The BeginUpdate action has to be requested before the Update. This error will be returned if the server requires the *BeginUpdate* UpdateAction and the request did not specify the *LockKey*, implying that no lock has been acquired, see Table 11-17 RequiresBegin.

20315	Record Lock Expired. The BeginUpdate action has to be requested before the Update. This error will be returned if the record is not locked and the request specified the LockKey, indicating that a lock had been previously requested.
20316	Unknown UpdateAction. The requested UpdateAction is not defined for this class.
20317	Unknown Resource or ClassName.
20318	Requested record does not exist.

The quoted-string of the body-start-line contains text that MAY be displayed to the user.

Note: An Approved RCP is Related to this Section
Section 10.8 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 59 Revised Update Transaction

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Section 11 - Metadata Format

Metadata enables a client that receives data from a compliant server to better format the data for display, and to store it efficiently for future retrieval. While use of the metadata is not necessary to retrieve data for simple display purposes, more sophisticated clients will want to use the metadata to make more intelligent use of the information retrieved. Metadata MUST be supplied by a compliant server.

11.1 Organization and Retrieval

11.1.1 Metadata Organization

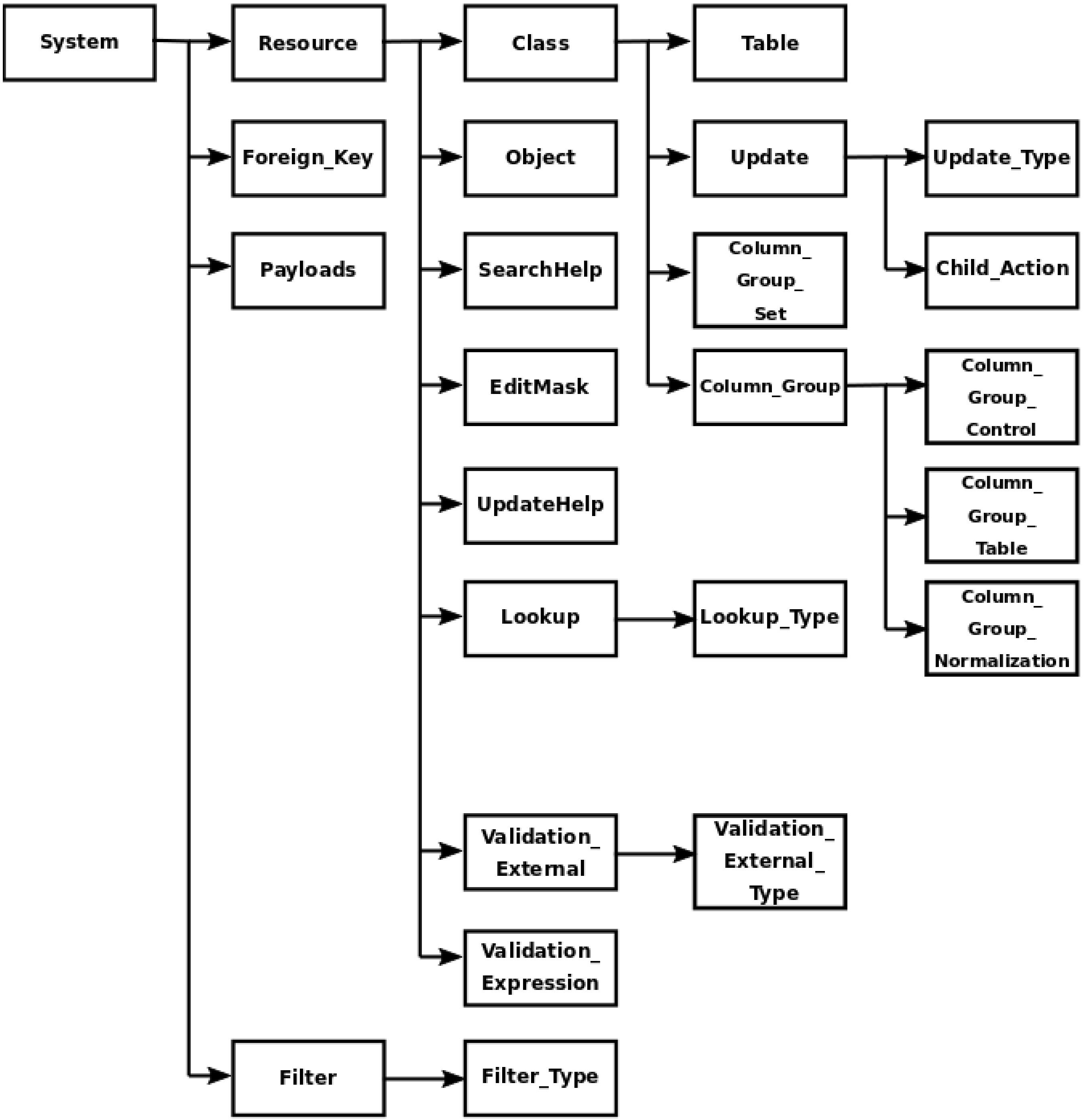
Metadata is organized by table/object, with each table having its own unique set of metadata describing the fields available in that table/object. The organization permits access to summary or detailed information about one or more resources (see Figure 11.1, "Metadata Structure").

The client retrieves the metadata by using the GetMetadata Transaction specifying the METADATA table/object(s) of interest as the Type, and the specific instance in the ID (see Section 12). The server supplies the metadata as documents using the formats described in this section. The client MUST accept fields and attributes in the metadata that are not defined in this standard, although it is not required to process those fields in any way.

The client MAY cache the metadata between sessions. If it does, it MUST record the value of the `METADATA-SYSTEM` timestamp attribute from each session in which it caches retrieved metadata, and MUST request new metadata whenever the `MetadataTimestamp` Login response value changes except when previous versions are permitted by the `MinMetadataTimestamp` value. Further, if the `MetadataID` is used in the System metadata, see Section 11.2.1, the client MAY cache separate metadata for each MetadataID. If it does, it MUST maintain both the `METADATA-SYSTEM` and the `MetadataID` and use the correct `MetadataID` and timestamp based on the Login Transaction response values.

If a client sends transactions using outdated metadata or uses metadata that does not match the Login Transaction `MetadataID` value, when present, the server operation is undefined. Server vendors MAY respond with an ErrorCode, but that may not be possible for all cases of metadata mismatch.

Figure 11.1 - Metadata Structure



NOTE:

> The names of the metadata provided in the figure may not be the correct header tag values that should be used in a GetMetadata transaction Request or Response. For the proper metadata-id value please refer to the appropriate version of the *RETS Metadata Schema*.

Note: RETS1.8.0: An Approved RCP is Related to this Section
Section 11.1.1 is related to the following approved RCP(s):

- RCP 60 Metadata Changes for Update
- RCP 87 RETS 1.7.2 Errata Document
- RCP 101 Child Rows Support
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Note: RETS1.9.0: An Approved RCP is Related to this Section
Section 11.1.1 is related to the following approved RCP(s):

- RCP 110 - Deprecate and Replace GetPayloadList
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.1.2 General Rules for Interpretation

In general, metadata keywords defined in this standard such as field names and reserved values are not case-sensitive. However, implementers are urged to adopt the strict-generation/tolerant-acceptance rule and follow the case shown in this standard.

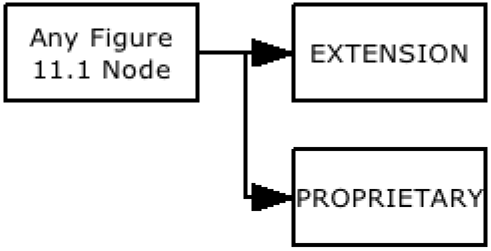
Clients requesting metadata in COMPACT format MUST ignore any metadata fields which they do not understand. In addition, the servers are permitted to send columns in any order. The order shown in the examples is not normative.

Servers may choose to extend the content of any metadata table by including additional keywords. These keywords MUST be contained under the `<PROPRIETARY>` element. The use of the `<EXTENSION>` element is reserved for the standard. See Figure 11.2 for further information.

Metadata field names for such extensions SHOULD begin with the letter "X" followed by a hyphen, followed by an implementation-defined token in order to insure compatibility with future versions of the standard.

Clients requesting metadata in XML format MUST ignore any `<EXTENSION>` or `<PROPRIETARY>` elements that they do not understand.

Figure 11.2 - Metadata Structure



NOTE:

> RETS 1.8.0 requires all server responses to be well-formed XML, and additionally requires GetMetadata responses to be valid XML. In addition, RETS requires that clients parse server responses as XML, not as simple text streams. The response formats shown here are normative with respect to content, but not normative with respect to form. That is, servers are free to produce response XML in any format that complies with the W3C XML 1.0 recommendation, so long as it is valid with respect to the appropriate DTD. XML escaping of content is implied, as is XML processing of whitespace and line endings. See the W3C *XML Recommendation 1.0, Third Edition*, for full information on XML.

Note: RETS1.8.0: An Approved RCP is Related to this Section
Section 11.1.2 is related to the following approved RCP(s):

- RCP 87 RETS 1.7.2 Errata Document
  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.1.3 Metadata Retrieval Hierarchy

The ID argument in the GetMetadata transaction reflects the metadata hierarchy as shown in Figure 11.1. For any metadata element, the ID argument is a list of the names of the parent elements for the desired element, separated by colons. For example, to retrieve the EditMask table for a given named Resource, the argument is simply the ResourceID:

`Type:`	`METADATA-EDITMASK`
`ID`	`Property`

where `Property` is the `ID` of one of the Resources listed in the Metadata-Resource table.

Since Tables are children of Classes, which are in turn children of Properties, the ID parameter contains both parents:

`Type`	`METADATA-TABLE`
`ID`	`Property : Res`

where `Res` is a class listed in the Metadata-Class table under the resource `Property`.

11.1.4 Metadata Format

Compliant RETS servers MUST supply metadata in both formats: COMPACT, described below and valid according to the RETS Metadata Content XML Schema rets-metadata-compact-1_8_0.xsd, and XML, valid according to the RETS XML Metadata Schemas rets-metadata-xmlcontent-1_7_2.xsd and rets-metadata-xmlresponse-1_7_2.xsd. See Appendix A for additional information.

The COMPACT metadata format consists of a sequence of segments with identical structure, except for System-level metadata, which has its own structure. The general structure for non-System metadata is a tab-delimited table, XML-encapsulated with the header record contained within a `<COLUMNS>` element, and each successive row contained within a `<DATA>` element.:

```
<METADATA-HEADER header-attributes>
<COLUMNS>fieldname *(fieldname)</COLUMNS>
*(<DATA>fielddata *(fielddata)</DATA>)
</METADATA-HEADER>
```

`METADATA-HEADER` is the header name for the segment, given with the description of each type of metadata, as are the *`header-attributes`* associated with each header. Each *`fieldname`* is the name of one of the metadata fields given below. Each *`fielddata`* value corresponds to the similarly-positioned *`fieldname`*, first to first, second to second and so on.

11.1.5 Metadata StandardNames

In order to support alternate or changing standard names, the server MAY support the optional StandardNames argument for the GetMetadata transaction. This argument declares the name space of the Standard Names shown in the RETS metadata. The server MUST not change other metadata when supporting this parameter. Such that a client requesting ‘StandardNames=Standard-XML: RETS- 20080829.dtd’ will receive metadata containing Standard Names from the Standard-XML RETS-20080829.dtd version of the data standard. The following table defines the representation of the StandardNames argument in BNF form.

StandardNames	::= STANDARD-XML | payload-name “:” version
payload-name	::= RETSNAME
version	::= dtd-version | 1*64PLAINTEXT

Example cases:

If the StandardNames argument is not submitted or empty then return standard names from the default STANDRD-XML format in the StandardName metadata.

If StandardNames=STANDARD-XML is submitted then return standard names from the default STANDRD-XML format in the StandardName metadata.

If StandardNames=STANDARD-XML: RETS-20080829.dtd is submitted then return standard names defined by the default STANDRD-XML format in the StandardName metadata.

If StandardNames= DataDictionary:1.0 is submitted then return standard names defined by the DataDictionary 1.0 payload format in the StandardName metadata.

Note: RETS1.8.0: An Approved RCP is Related to this Section
Section 11.1.5 is related to the following approved RCP(s):

- RCP 100 Alternate Standard Names

  Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.2 System-Level Metadata

Clients can determine the number and type of searchable and updateable entities by referencing the Resources. A server MUST advertise its resources. It MAY advertise all of its available resources or MAY restrict the advertised list by logon or other criteria. A server's advertisement of a resource does not require that the server be able to accommodate any arbitrary search for that user; the server MAY restrict access to resources that it advertises. If the server supports multimedia objects then it MUST advertise the supported types.

All resources that can be searched or updated are defined in the metadata described in this section. There are three parts to the metadata. The first part provides system information and describes the available resources, the second part describes the class specific metadata for a resource, and the third part describes the shared metadata for a resource.

11.2.1 System

The System metadata starts with a `<METADATA-SYSTEM>` tag with Version and Date attributes. This tag is followed by a `<SYSTEM>` section, which contains the system identification information and time offset. An optional <`COMMENTS`> section completes the System metadata. The System metadata has the following format:

```
<METADATA-SYSTEM Version="rets-version-type" Date="system-date" >
<SYSTEM - SystemID="code-name" - SystemDescription="long-name"
[TimeZoneOffset="time-zone-offset"][MetadataID="metadata-id"]/>
[<COMMENTS>
*( comment )
</COMMENTS> ]
</METADATA-SYSTEM>
```

system-date	::= *retsdatetime*
code-name	::= *1*128IDALPHANUM*
long-name	::= *1*128PLAINTEXT*
time-zone-offset	::= *time-offset*
metadata-id	::= *1*128IDALPHANUM*
comments	::= *TEXT*

Table 11-1 MetadataSystem Compact Header Attributes

Attribute	Content
`Version`	This is the version of the System metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.

COMPACT header tag: **`METADATA-SYSTEM`**

Table 11-2 System Compact Header Attributes

Attribute	Content
`SystemId`	An identifier for the system
`SystemDescription`	An implementation defined description of the system
`TimeZoneOffset`	The Time Zone Offset is the time offset of the server relative to UTC. The server MAY provide the TimeZoneOffset to assist in correctly calculating date and time values for requests to this server. The format is defined in Section 2.4 for the atom time-offset. Any server that provides the TimeZoneOffset value in System Metadata MUST adhere to this value when responding to requests. Client applications SHOULD use this value to calculate the correct date and time criteria for requests.
`MetadataID`	An optional identifier for caching role-based metadata. See Section 4.7.5

COMPACT header tag: **`SYSTEM`**

Table 11-3 Metadata: System Field

Field Name	Content Type	Description
`COMMENTS`	*TEXT*	Optional comments about the system. The context where the field contains characters may require that those characters are escaped by other rules like entity encoding.
`ResourceVersion`	*resource-version*	The version of the set of Resource Metadata
`ResourceDate`	*resource-date*	The date of the version of the set of Resource Metadata
`ForeignKeyVersion`	*foreignkey-version*	The version of the set of ForeignKey Metadata
`ForeignKeyDate`	*foreignkey-date*	The date of the version of the set of ForeignKey Metadata
`FilterVersion`	*filter-version*	The version of the set of Filter Metadata

`FilterDate`	*`filter-date`*	The date of the version of the set of Filter Metadata

`resource-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS`

`resource-date ::= RETSDATETIME`

`foreignkey-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS`

`foreignkey-date ::= RETSDATETIME`

`filter-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS`

`filter-date ::= RETSDATETIME`

Note: An Approved RCP is Related to this Section
Section 11.2.1 is related to the following approved RCP(s):

RETS 1.7.2:

- RCP 71 Time Zone Data

RETS 1.8.0

- RCP 70 Metadata Role Support
- RCP 98 - Additional Information Fields in METADATA-SYSTEM and Login

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.2.2 Resources

RETS does not require that any particular type of data be made available by a server. However, a server MUST use a standard well-known name under which to make its data available if a suitable well-known name is defined in the standard. Table 11-4 contains the list of well-known resource names.

Table 11-4 Well-Known Resource Names

Resource Name	Purpose
`ActiveAgent`	A resource that contains information about active agents. These are agents that are currently authorized to access the server (paid-up, not retired, etc.)
`Agent`	A resource that contains information about agents.
`History`	A resource that contains information about the accumulated changes to each listing.
`Office`	A resource that contains information about broker offices.
`OpenHouse`	A resource that contains information about open-house activities.
`Property`	A resource that contains information about listed properties. Information in this resource is described by Real Estate Transaction XML DTD in addition to appropriate metadata.
`Prospect`	A resource that contains information about sales or listing prospects.
`Tax`	A resource that contains tax assessor information.
`Tour`	A resource that contains information about tour activities.

Resource Metadata Content

Table 11-5 Resource Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Resource metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.

COMPACT header tag: **`METADATA-RESOURCE`**

Table 11-6 Metadata: Resource Description Fields

Field Name	Content Type	Description
`ResourceID`	*`RETSID`*	The name which acts as a unique ID for this resource.
`StandardName`	*`1*64ALPHANUM`*	The name of the resource. This must be a well-known name if applicable.
`VisibleName`	*`1*64PLAINTEXT`*	The user-visible name of the resource.
`Description`	*`1*64PLAINTEXT`*	A user-visible description of the resource.
`KeyField`	*`RETSNAME`*	The `SystemName` (see 11.3.2) of the field that provides a unique `ResourceKey` for each element in this resource. All classes within a resource must use the same `KeyField`.
`ClassCount`	*`POSITIVENUM`*	The number of classes in this resource. There MUST be `ClassCount` `METADATA_CLASS` descriptions for the resource. There MUST be at least one Class for each Resource.
`ClassVersion`	*`rets-version-type`*	The latest version of the Class metadata for this Resource. The version number is advisory only.
`ClassDate`	*`RETSDATETIME`*	The date on which the Class metadata for this Resource was last changed. Clients MAY rely on this date for cache management.
`ObjectVersion`	*`rets-version-type`*	The version of the Object metadata for this Resource. The version number is advisory only. A blank version indicates no Object metadata is available for this Resource.
`ObjectDate`	*`RETSDATETIME`*	The date on which the Object metadata for this Resource was last changed. Clients MAY rely on this date for cache management. A blank date indicates no Object metadata is available for this Resource.
`SearchHelpVersion`	*`rets-version-type`*	The version of the SearchHelp metadata for this Resource. The version number is advisory only. A blank version indicates no SearchHelp is available for this Resource.
`SearchHelpDate`	*`RETSDATETIME`*	The date on which the SearchHelp metadata for this Resource was last changed. Clients MAY rely on this date for cache management. A blank date indicates no SearchHelp is available for this Resource.
`EditMaskVersion`	*`rets-version-type`*	The version of the EditMask metadata for this Resource. The version number is advisory only. A blank version indicates no EditMask is available for this Resource.
`EditMaskDate`	*`RETSDATETIME`*	The date on which the EditMask metadata for this Resource was last changed. Clients MAY rely on this date for cache management. A blank date indicates no EditMask is available for this Resource.
`LookupVersion`	*`rets-version-type`*	The version of the Lookup metadata for this Resource. The version number is advisory only. A blank version indicates no Lookup is available for this Resource.
`LookupDate`	*`RETSDATETIME`*	The date on which the Lookup metadata for this Resource was last changed. Clients MAY rely on this date for cache management. A blank date indicates no Lookup is available for this Resource.
`UpdateHelpVersion`	*`rets-version-type`*	The version of the UpdateHelp metadata for this Resource. The version number is advisory only. A blank version indicates no UpdateHelp is available for this Resource.
`UpdateHelpDate`	*`RETSDATETIME`*	The date on which the UpdateHelp metadata for this Resource was last changed. Clients MAY rely on this date for cache management. A blank date indicates no UpdateHelp is available for this Resource.
`ValidationExpressionVersion`	*`rets-version-type`*	The version of the ValidationExpression metadata for this Resource. The version number is advisory only. A blank version indicates no ValidationExpression is available for this Resource.

`ValidationExpressionDate`	*`RETSDATETIME`*	The date on which the ValidationExpression metadata for this Resource was last changed. Clients MAY rely on this date for cache management. A blank date indicates no ValidationExpression is available for this Resource.
`ValidationLookupVersion`	*`rets-version-type`*	The version of the ValidationLookup metadata for this Resource. The version number is advisory only. A blank version indicates no ValidationLookup is available for this Resource.
`ValidationLookupDate`	*`RETSDATETIME`*	The date on which the ValidationLookup metadata for this Resource was last changed. Clients MAY rely on this date for cache management. A blank date indicates no ValidationLookup is available for this Resource.
`ValidationExternalVersion`	*`rets-version-type`*	The version of the ValidationExternal metadata for this Resource. The version number is advisory only. A blank version indicates no ValidationExternal is available for this Resource.
`ValidationExternalDate`	*`RETSDATETIME`*	The date on which the ValidationExternal metadata for this Resource was last changed. Clients MAY rely on this date for cache management. A blank date indicates no ValidationExternal is available for this Resource.

Note: An Approved RCP is Related to this Section
Section 11.2.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 87 RETS 1.7.2 Errata Document

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.2.3 Foreign Keys

The ForeignKeys metadata table allows a server to advertise relationships among its offered resources. A RETS client MAY use this information to provide a richer display of related information. The ForeignKeys metadata consists of tuples containing a parent resource type, a child resource type, and the foreign keys used to traverse the relation.

The nesting of foreign keys MUST be such that recursive searches are NOT REQUIRED to obtain data for well-known fields as defined in the RETS DTD. However, nesting of foreign keys is allowed except in these cases.

ForeignKeys Metadata Content

COMPACT header tag: **`METADATA-FOREIGN_KEY`~~`S`~~**

Table 11-7 ForeignKeys Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the ForeignKeys metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.

Table 11-8 Metadata Content: Foreign Keys

Metadata Field	Content Type	Description
`ForeignKeyID`	*`RETSID`*	A Unique ID that represents the foreign key combination.
`ParentResourceID`	*`RETSID`*	The `ResourceID` (Table 11-6) of the resource for which this field functions as a foreign key. The name given MUST appear in the `METADATA-RESOURCE` table..
`ParentClassID`	*`RETSID`*	The name of the resource class for which this field functions as a foreign key. This name MUST appear in the `RESOURCE-CLASS` table for the given `ParentResourceID`.

`ParentSystemName`	*RETSNAME*	The SystemName of the field in the given resource class that should be searched for the value given in the this field. This name must appear as a `SystemName` in the `METADATA-TABLE` section of the metadata for the `ParentClassID`, and the named item must have its Searchable attribute set to *`true`*.
`ChildResourceID`	*RETSID*	The `ResourceID` (Table 11-6) of the resource for which this field functions as a foreign key. The name given MUST appear in the `METADATA-RESOURCE` table.
`ChildClassID`	*RETSID*	The name of the resource class for which this field functions as a foreign key. This name MUST appear in the `RESOURCE-CLASS` table for the given `ChildResourceID`.
`ChildSystemName`	*RETSNAME*	The SystemName of the field in the given resource class that should be searched for the value given in this field. This name must appear as a SystemName in the `METADATA-TABLE` section of the metadata for the ChildClassID, and the named item must have its Searchable attribute set to *`true`*.
`ConditionalParentField`	*RETSNAME*	The SystemName of a field in the parent's `METADATA-TABLE` that should be examined to determine whether this parent-child relationship should be used. If this is blank, the relationship is unconditional. If `ConditionalParentField` is present and nonblank, then `ConditionalParentValue` MUST be present and nonblank.
`ConditionalParentValue`	*RETSNAME*	The value of the field designated by `ConditionalParentField` indicating that this relation should be used. If the type of the field named in `ConditionalParentField` is numeric, then this value is converted to numeric type before comparison. If the type of the field named in `ConditionalParentField` is character, then the shorter of the two values is padded with blanks and the comparison made for equal length. If `ConditionalParentField` is present and nonblank, then `ConditionalParentValue` MUST be present and nonblank.
`OneToManyFlag`	*BOOLEAN*	A truth value that indicated if the foreign key will return multiple rows if queried from the source to the destination.

> **Note: An Approved RCP is Related to this Section**
> Section 11.2.2 is related to the following approved RCP(s):
>
> RETS 1.8.0
>
> - RCP 60 Metadata Changes for Update
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.2.4 Filter

Filters and FilterType values describe a parent-child relation between Lookups by specifying which values in the Child lookup are legal based on the value in Parent lookup. If filtered lookups are used in the Table metadata (meaning the LookupName and FilterParentField metadata items are not empty), the server MUST guarantee that the values in the child field will not fall outside the set specified by the filter based on the value in the parent field. The client SHOULD use the filter information when checking values for the Update Transaction. The client MAY use the filter information when providing values for the Search Transaction. The LookupFilter argument in GetMetadata can also be used to limit the amount of metadata information sent by the server.

An example of using Filter and FilterType is to consider a system that has two fields called State and City that are lookup values. The field State contains lookup rows that describe the 5 states of the United States of America that this system is deployed and the City field contains lookup rows that describe 500 cities that are a subset of all the cities in the United States of America in these 5 states. Without filters, the City lookup will be 500 entries long. With filters, the lookup of City can be limited based on State to shorter lists.

Table 11-8 Filter Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Filter metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.

COMPACT header tag: **`METADATA-FILTER`**

Table 11-9 Metadata: Filter Description Fields

Field Name	Content Type	Description
`FilterID`	*RETSID*	The name which acts as a unique ID for this filter.
`ParentResource`	*RETSID*	ResourceID of the parent lookup.
`ParentLookupName`	*RETSNAME*	LookupName of the parent lookup.
`ChildResource`	*RETSID*	ResourceID of the child lookup.
`ChildLookupName`	*RETSNAME*	LookupName of the child lookup.
`NotShownByDefault`	*BOOLEAN*	If true the server will by default not include the FilterValue data of this filter in any metadata request, unless specifically asked to using the LookupFilter argument in GetMetadata.

Note: An Approved RCP is Related to this Section
Section 11.2.4 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 60 Metadata Changes for Update

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.2.5 Filter Type

Table 11-10 Filter Type Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Filter Type metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Filter`	The FilterID of the **`METADATA-FILTER`** parent entry that this child entry belongs.

COMPACT header tag: **`METADATA-FILTER_TYPE`**

Table 11-11 Metadata: Filter Description Fields

Field Name	Content Type	Description
`FilterTypeID`	*RETSID*	The name which acts as a unique ID for this filter type.
`ParentValue`	*1*128PLAINTEXT*	The LookupType Value field for the LookupType in the parent lookup.
`ChildValue`	*1*128PLAINTEXT*	The LookupType Value field for the LookupType in the child lookup.

Note: An Approved RCP is Related to this Section
Section 11.2.5 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 60 Metadata Changes for Update

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.2.6 Payloads

Table 11-63 Payload Type Metadata Compact Header Attributes

Attribute	Content

`Version`	This is the version of the Payloads metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.

COMPACT header tag: **`METADATA-PAYLOAD`**

Table 11-64 Metadata: Payload Description Fields

Field Name	Content Type	Description
`ResourceID`	*`RETSID`*	The Resource ID of the resource for this payload.
`ClassName`	*`RETSNAME`*	The Class ID of the class for this payload.
`PayloadID`	*`RETSID`*	A Unique ID that represents the Payload.
`PayloadName`	*`RETSNAME`*	Name of the payload.
`Version`	*`dtd-version | 1*64PLAINTEXT`*	A string representing the version of the data in the payload. If the payload name is "STANDARD-XML" the version MUST be in std-version format described in Section 7.5 - Search Response Body Format.
`URI`	*`URI`*	A valid location for an XML-Schema defining the payload format.
`Description`	*`1*256PLAINTEXT`*	The LookupType Value field for the LookupType in the child lookup.

Note: An Approved RCP is Related to this Section
Section 11.2.6 is related to the following approved RCP(s):

RETS 1.9.0

- RCP 110 - Deprecate and Replace GetPayloadList

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.3 Metadata Format for Class Elements

Resources and Classes that can be searched have metadata that describes fields for specific combinations of Resource and Class and are defined in this section.

11.3.1 Class

A given data resource may contain multiple classes of entries that can be searched or updated separately. The metadata for a resource supporting searchable classes MUST contain a class description for each class supported.

Table 11-12a Well-Known Class Names

Resource Name	Purpose
`ActiveAgent`	A class that contains information about active agents. These are agents that are currently authorized to access the server (paid-up, not retired, etc.)
`Agent`	A class that contains information about agents.
`History`	A class that contains information about the accumulated changes to each listing.
`Office`	A class that contains information about broker offices.
`OpenHouse`	A class that contains information about open-house activities.

`Prospect`	A class that contains information about sales or listing prospects.
`Tax`	A class that contains tax assessor information.
`Tour`	A class that contains information about tour activities.
`ResidentialProperty`	A Class that contains information about single family properties. Information in this Class is described by Real Estate Transaction XML Schema in addition to appropriate metadata.
`LotsAndLand`	A Class that contains information about lots and land properties. Information in this Class is described by Real Estate Transaction XML Schema in addition to appropriate metadata.
`MultiFamily`	A Class that contains information about multi-family properties. Information in this Class is described by Real Estate Transaction XML Schema in addition to appropriate metadata.

COMPACT header tag: **`METADATA-CLASS`**

Table 11-12 Class Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Class metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource in which this class resides.

Table 11-13 Metadata Content: Resource Class

Metadata Field	Content Type	Description
`ClassName`	*`RETSNAME`*	The name which acts as a unique ID for the class.
`StandardName`	*`1*64PLAINTEXT`*	The Well-Known Class names from Table 11-12a
`VisibleName`	*`1*64PLAINTEXT`*	The user-visible name of the class.
`Description`	*`1*128PLAINTEXT`*	A user-visible description of the class.
`TableVersion`	*`rets-version-type`*	The version of the Table metadata that describes this Class. The version number is advisory only.
`TableDate`	*`RETSDATETIME`*	The date on which the Table metadata for this Class was last changed. Clients MAY rely on this date for cache management.
`UpdateVersion`	*`rets-version-type`*	The latest version of any of the Update metadata for this Class. A blank version indicates no Update is available for this Class. The version number is advisory only.
`UpdateDate`	*`RETSDATETIME`*	The date on which any of the Update metadata for this Class was last changed. Clients MAY rely on this data for cache management. A blank date indicates no Update is available for this Class.
`ClassTimeStamp`	`RETSNAME`	The `SystemName of the field in the METADATA-TABLE` that acts as the last-change timestamp for this class.
`DeletedFlagField`	`RETSNAME`	The `SystemName of the field in the METADATA-TABLE` that indicates that the record is logically deleted. If this element is specified, then `DeletedFlagValue` MUST be specified as well.
`DeletedFlagValue`	`1*32ALPHANUM`	The value of the field designated by `DeletedFlagField` indicating that a record has been logically deleted. If the type of the field named by `DeletedFlagField` is numeric, then this value is converted to a number before comparison. If the type of the field named by `DeletedFlagField` is character, then the shorter of the two values is padded with blanks and the comparison made for equal length.
`HasKeyIndex`	*`BOOLEAN`*	When true, indicates that the Class supports the retrieval of key data for fields advertised in the Table Metadata as InKeyIndex.

`OffsetSupport`	*BOOLEAN*	When true, indicates that the server MUST honor the Offset parameter when searching this class. When false, indicates that the server does not support the Offset functionality for this class.

Note: Approved RCP are Related to this Section
Section 11.3.1 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 68 Search Has Key Index Support
- RCP 75 Offset Availability in the Metadata
- RCP 90 Deprecate CommonInterest Class Well-Known Name

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.3.2 Table

The following table lists the minimum acceptable content for server-supplied metadata used in describing a table.

COMPACT header tag: **`METADATA-TABLE`**

Table 11-14 Table Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Table metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource in which this table resides.
`Class`	The ClassName for the class in which this table resides.

Table 11-15 Metadata Content - Tables

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that never changes as long as the semantic definition of this field remains unchanged. In particular, it should be managed so as to allow the client to detect changes to the `SystemName`.
`SystemName`	*RETSNAME*	The name of the field as it is known to the native server. The system name MUST be unique within the Table.
`StandardName`	*RETSNAME*	The name of the field as it is known in the Real Estate Transaction XML DTD or XML Schema designated by the StandardNames argument or the name of the field as it is known in the RETS Standard-XML DTD.
`LongName`	*1*256TEXT*	The name of the field as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this datum when it appears on a report.
`DBName`	*1*10ALPHANUM*	A short name that can be used as a database field name. This name may not start with a number nor can it be an ANSI-SQL92 reserved word. This value can be used by a client as the name of an internal database field, so servers should attempt to provide a value for this field that is unique within the table.
`ShortName`	*1*64TEXT*	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined. It is expected that clients will use this field in human-interface elements such as pick lists.

`MaximumLength`	*POSITIVENUM*	The maximum possible character length after all Transport layer encoding. Transport layer encoding includes both HTTP and XML based encoding, but does not include RETS Lookup Value to Lookup Long Value encoding. See Appendix D for a definition and examples of how a RETS server should calculate the MaximumLength.
`DataType`	`Boolean`	A truth-value, stored using *TRUE* and *FALSE*. That is 1 for true and 0 for false.
	`Character`	An arbitrary sequence of printable characters.
	`Date`	A date in *full-date* format.
	`DateTime`	A date and time in *RETSDATETIME* format.
	`Time`	A time in *RETSTIME* format.
	`Tiny`	A signed numeric value that can be stored in no more than 8 bits.
	`Small`	A signed numeric value that can be stored in no more than 16 bits.
	`Int`	A signed numeric value that can be stored in no more than 32 bits.
	`Long`	A signed numeric value that can be stored in no more than 64 bits.
	`Decimal`	A decimal value that contains a decimal point (see Precision).
	`Point`	Geospatial coordinates represented as a space-delimited pair of decimal numbers, showing first the longitude then the latitude of a point on the Earth’s surface.
`Precision`	*OPTNONNEGATIVENUM*	The number of digits to the right of the decimal point when formatted. Applies to Decimal fields only.
`Searchable`	*BOOLEAN*	When true, indicates that the field is searchable.
`Interpretation`	`Number`	An arbitrary number.
	`Currency`	A number representing a currency value.
	`Lookup`	A value that should be looked up in the Lookup Table. This is a single selection type lookup (e.g. STATUS). This interpretation is also valid for Boolean data types, in which case the `LookupType` specified by the `LookupName` entry MUST contain exactly two elements, one with a `Value` of 0, and the other with a `Value` of 1.
	`LookupMulti`	A value that should be looked up in the Lookup Table. This is a multiple-selection type lookup (e.g. FEATURES) where the character strings representing each selection are separated by commas.The character strings MAY be quoted text following the rules for Value of section 11.4.3 Lookup Type.
	`Location`	A Point value representing the longitude and latitude of a point on the Earth’s surface.
	`URI`	An arbitrary URI or URL that is fully qualified and that an application will be able to successfully link to.
`Alignment`	`Left`	The value MAY be displayed left aligned.
	`Right`	The value MAY be displayed right aligned.
	`Center`	The value MAY be centered in its field when displayed.

	Justify	The value MAY be justified within its field when displayed.
UseSeparator	*BOOLEAN*	When true, indicates that the numeric value MAY be displayed with a thousands separator.
EditMaskID	_RETSNAME *("," RETSNAME)_	For each *RETSNAME,* the name of the METADATA-EDITMASK EditMaskID containing the edit mask expression for this field (see Section 11.4.5). Multiple masks are permitted and are separated by commas.
LookupName	RETSNAME	The name of the METADATA-LOOKUP containing the lookup data for this field (see Section 11.4.2). Required if Interpretation is Lookup, LookupMulti, LookupBitstring or LookupBitmask.
MaxSelect	Numeric	This field is required if Interpretation is LookupMulti, LookupBitstring or LookupBitmask. This value indicates the maximum number of entries that may be selected in the lookup.
Units	(Feet | Meters | SqFt | SqMeters | Acres | Hectares)	Unit of measure.
Index	*BOOLEAN*	When true, indicates that this field is part of an index. The client MAY use this information to help the user create faster queries.
Minimum	Numeric	The minimum value that may be stored in a field (applies to numeric fields only).
Maximum	Numeric	The maximum value that may be stored in a field (applies to numeric fields only).
Default	*serial*	The order that fields should appear in a default one-line search result. Fields that should not appear in the default one-line format should have a value of 0, Fields that should never be visible to the user should have a value of --1.
Required	Numeric	A non-zero value indicates the field is required when searching. This value should be sequential starting with one. If multiple fields share the same value, then one of the fields with the same value is required. (e.g. City = 1 & ZipCode = 1 implies that the user is required to include either City or ZipCode in their query).
SearchHelpID	RETSNAME	The name of the entry in the METADATA-SEARCH_HELP table (see Section 11.4.4).
Unique	*BOOLEAN*	When true, indicates that this field is a unique identifier for the record that it is part of.
ModTimeStamp	*BOOLEAN*	When true, indicates that changes to this field update the class's ClassTimeStamp field.
ForeignKeyName	*RETSID*	When nonblank, indicates that this field is normally populated via a foreign key. The value is the ForeignKeyID from the METADATA-FOREIGN_KEYS table.
ForeignField	*RETSNAME*	The SystemName from the child record accessed via the specified foreign key.
KeyQuery[deprecated]	*BOOLEAN*	When true, indicates that this field may be included in a query that uses the Key optional argument.[deprecated]
KeySelect[deprecated]	*BOOLEAN*	When true, indicates that this field may be included in the Select list of a query that uses the Key optional argument.[deprecated]
InKeyIndex	*BOOLEAN*	When true, indicates that this field may be included in the Select argument of a Search to suppress normal Limit behavior following the rule described in Section 7.4.5
FilterParentField	*RETSNAME*	Specifies that values allowed in this field are limited by a Lookup filter, using the contents of the field named here as ParentValue. FilterParentField may only be specified with fields that have a LookupName, where the named Lookup has a non-empty FilterID.

`DefaultSearchOrder`	`Numeric`	The order that fields should appear in a default search screen that is excuted in order to give the user a list of existing records to select from. Fields that should not appear in the default search screen should have a value of '0'. Fields that should never be visible to the user should have a value of **'-1'** .
`Case`		Applicable when the field has a data type of Character. A value that indicates that the server will store the data with the specified case. This allows a client to automatically convert data in these fields to the correct case.
	`UPPER`	The data is stored on the server as upper case. A client should convert values in this field to upper case for both searches and updates. Servers MUST perform a case insensitive search.
	`LOWER`	The data is stored on the server as lower case. A client should convert values in this field to lower case for both searches and updates. Servers MUST perform a case insensitive search.
	`EXACT`	The data is stored on the server as entered by the user. The server MUST perform a case sensitive search.
	`MIXED`	The data is stored on the server as entered by the user. The server MUST perform a case insensitive search.

Note: Approved change proposals are related to this Section
Section 11.3.2 is related to the following approved RCP(s):

RETS 1.7.2

- RCP 71 Time Zone Data

RETS 1.8.0

- RCP 60 Metadata Changes for Update
- RCP 68 Search Has Key Index Support
- RCP 77 Maximum Field Length
- RCP 78 Specification Errata Changes
- RCP 87 RETS 1.7.2 Errata Document
- RCP 100 Alternate Standard Names

RETS1.9.0

- RCP 103 - Geospatial Search

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.3.3 Update

A given data resource may contain multiple classes of entries that can be updated separately. The metadata for a resource supporting updateable classes MUST contain a Class Table description for each class supported.

COMPACT header tag: **`METADATA-UPDATE`**

Table 11-16 Update Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Update metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource to which this metadata table applies.
`Class`	The ClassName for the class to which this metadata table applies.

Table 11-17 Metadata Content - Update

Metadata Field	Content Type	Description

`MetadataEntryID`	*RETSID*	A value that never changes so long as the semantic definition of this entry remains unchanged.
`UpdateAction`	*1*24ALPHANUM*	This identifies the nature of the update, such as "add" or "modify". Some update types, such as changes to a property record (e.g. "Sell", "Back on Market"), will imply a set of business rules specific to the server. However, where possible, the following standard type names should be used:
	Add	Add a new record
	Clone	Create a new record by copying an old record
	Change	Change an existing record
	Delete	Delete an existing record
	BeginUpdate	MAY be requested before any other Update request to get the specified record's actual data and to put a lock on the record. The server MAY lock the requested record until another Update for that record is requested.
	CancelUpdate	MUST be used after **BeginUpdate** if no other update is requested on the locked record. It is not an error to request **CancelUpdate** on a record that is not locked
	ShowLocks	Request to show which records are currently locked by this user. The server MUST respond with a column-tag showing the `KeyField` and `Lock` in the `COMPACT-DATA` format containing one line for each locked record. The `Lock` indicates that number of seconds before the record lock will expire.
`Description`	*1*64PLAINTEXT*	A user visible description of the Update Type.
`KeyField`	*RETSNAME*	The SystemName (see Section 11.3.2) of the field that must be used to retrieve an existing record for the update.
`UpdateTypeVersion`	*1*2DIGITS . 1*2DIGITS . 1 *5DIGITS*	The latest version of this Update Type metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. The version number is advisory only.
`UpdateTypeDate`	*RETSDATETIME*	The date on which any of the content of this Update Type was last changed. Clients MAY rely on this date for cache management.
`RequiresBegin`	*BOOLEAN*	If this value is `TRUE` (1), the **BeginUpdate** action MUST be called before this update action.
`IsChildActionOnly`	`BOOLEAN`	If this value is TRUE (1), the **UpdateAction** MUST be performed as a child (ChildRecord) of a parent's **UpdateAction**.

The optional **IsChildActionOnly** flag is used to indicate that this **UpdateAction** MUST be performed as a child of a parent's **UpdateAction**. If not specified or omitted, the default value of **FALSE** (0) is used. If **TRUE** then rows must exist in Child_Action metadata describing valid child actions and the client MUST submit this data in a ChildRecord parameter of a parent's **UpdateAction**.

Note: An Approved RCP is Related to this Section
Section 11.3.3 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 101 - Child Rows Support

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.3.4 Update Type

11.3.4 Update Type

A given Resource may contain multiple classes of entries that can be updated separately. Each of these classes may have different types of updates that can be performed. There might be different test expressions or sequences. This section describes how each of those are specified.

COMPACT header tag: **`METDATA-UPDATE_TYPE`**

Table 11-18 UpdateType Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Update Type metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the Resource that this metadata table applies.
`Class`	The ClassName for the Class to which this metadata table applies.
`Update`	The UpdateAction for the Update that this metadata table applies.

Table 11-19 Metadata Content - Update Type

Metadata Field	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that never changes as long as the semantic definition of this entry remains unchanged.
`SystemName`	*RETSNAME*	This is the SystemName of the field as defined in Section 11.3.2 .
`Sequence`	*1*5DIGIT*	Sequence number of the field, representing the order of entry
`Attributes`	1*(**1** | **2** | **3** | **4** | **5** | **6** | **7** [,])	Multiple entries are separated by commas.
	1	Display Only - Field may not be changed.
	2	Required - Field may not be left blank.
	3	Autopop - Field is populated by the server.
	4	Interactive-Validate - When changed, the client can validate the field only by contacting the server. All fields listed as "AdditionalField" MUST also be passed.
	5	Clear On Cloning - Field SHOULD be cleared when the containing record is cloned.
	6	Autopop Required - Field is mandatory when calling the Update transaction for Auto-population (validate-flag=1).
	7	Hidden - Field may be used in ValidationExpression, but is to remain hidden from the user.
`Default`	*<PLAINTEXT>*	Default value of field (i.e. value if not specified by user)
`ValidationExpressionID`	*(RETSNAME *(", " RETSNAME)*	<multiple entries are separated by commas> The names of the ValidationExpressions to use. See section 11.4.9
`UpdateHelpID`	*RETSNAME*	The name of the entry in the `METADATA-UPDATE_HELP` table (see Section 11.4.6).
`ValidationLookupName` deprecated	*RETSNAME*	deprecated - The name of the ValidationLookup to use. See section 11.4.7
`ValidationExternalName`	*RETSNAME*	The name of the ValidationExternal to use. See section 11.4.10
`MaxUpdate`	*1*5DIGIT*	For LookupMulti fields, the maximum number of values that may be specified for the field. This value has no meaning for fields with any other interpretation.
`SearchResultOrder`	`Numeric`	The order that fields should appear in a default one-liner search result that is executed in order to give the user a list of existing records to select from for updating. Fields that should not appear in the default one-line format should have a value of "0". Fields that should never be visible to the user should have a value of "-1".

`SearchQueryOrder`	`Numeric`	The order that fields should appear in a default search screen that is executed in order to give the user a list of existing records to select from for updating. Fields that should not appear in the default search screen should have a value of "0". Fields that should never be visible to the user should have a value of "-1".

Note: An Approved RCP is Related to this Section
Section 11.3.4 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 59 Revised Update Transaction
- RCP 60 Metadata Changes for Update

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.3.5 Child Action

11.3.5 Child Action

A given update action may have multiple permitted child actions. Child Actions describe actions that can be performed on related classes during an update. The child class is designated by specifying the RETSID of the Foreign Key in the ForeignKeyID column and the child’s update action in the Update column. All fields in this metadata table MUST contain values.

COMPACT header tag: **`METADATA-CHILD_ACTION`**

Table 11-63 Child Action Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Child Action metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the Resource that this metadata table applies.
`Class`	The ClassName for the Class to which this metadata table applies.
`Update`	The UpdateAction for the Update that this metadata table applies.

Table 11-64 Metadata Content - Child Action

Metadata Field	Content Type	Description
`ChildActionID`	*RETSID*	A value that never changes as long as the semantic definition of this entry remains unchanged. It identifies this child action for the containing parent record update action.
`ForeignKeyID`	*RETSNAME*	The ForeignKeyID (Table 11-8) that describes the relationship between the parent and child. The ForeignKey metadata MUST have ParentResourceID and ParentClassID that match the Resource and Class header attributes of this Child Action. The ChildResourceID and ChildClassID match the child's resource and class of the child. The OneToManyFlag MUST be true.
`Update`	*1*24ALPHANUM*	The UpdateAction of the child class.

Note: An Approved RCP is Related to this Section
Section 11.3.5 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 101 Child Rows Support

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.4 Metadata Format for Shared Elements

Shared elements are metadata items that can apply to one or more Resource or Resource/Class types. They include Lookup values and help files.

11.4.1 Object

Object type names allow the operator of a particular server to advertise its supported multimedia types. These types are standard MIME types as registered with IANA. RETS does not require that a server make available any particular type of multimedia object. However, a server MUST use a standard well-known name under which to make its multimedia objects available, if a suitable well-known name is defined in the standard. Multimedia names are defined in Table 11-17.

COMPACT header tag: **`METDATA-OBJECT`**

Table 11-20 Well-known Object Types

Object Name	Purpose
`Photo`	A representation image related to the element defined by the Resource KeyField.
`Plat`	An image of the property boundaries related to the element defined by the Resource KeyField
`Video`	A moving image with or without sound related to the element defined by the Resource KeyField.
`Audio`	A sound clip related to the element defined by the Resource KeyField.
`Thumbnail`	A lower-resolution image related to the element defined by the Resource KeyField.
`Map`	A location image related to the element defined by the Resource KeyField.
`VRImage`	A multiple-view, possibly-interactive image related to the element defined by the Resource KeyField.

Table 11-21 Object Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Object metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata.This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the Resource to which this metadata table applies.

Table 11-22 Metadata Content: Resource Object

Metadata Field	Content Type	Description
`MetadataEntryID`	*`RETSID`*	A value that never changes as long as the semantic definition of this field remains unchanged.
`ObjectType`	*`1*24ALPHANUM`*	The classification of the Object. If one of the well-known Object types in Table 11-20 applies, then it MUST be used.

`MIMEType`	A comma separated list of MIME type/subtype per RFC 2045	The mime-type/subtypes of the Object type. This is the collection of Object media encodings available for the Objects on this system. Objects may have one or more mime-type of those listed in this field. This list is the mime-types that can be passed by the client in the "Accept" parameter in the GetObject transaction. All Objects can return a mime-type of text/xml as an error code/error reply when a fault occurs in the GetObject transaction. See Section 5.1.
`VisibleName`	*`1*64PLAINTEXT`*	The user-visible name of the Object type.
`Description`	*`1*128PLAINTEXT`*	A user-visible description of the Object type.
`ObjectTimeStamp`	*`RETSNAME`*	The `SystemName of the field in a METADATA-TABLE` that acts as the timestamp for Objects of this type. This `SystemName` MUST be one that appears in every class that has Objects of this type.
`ObjectCount`	*`RETSNAME`*	The `SystemName` of the field in a `METADATA-TABLE` that acts as the count for Objects of this type. This `SystemName` MUST be one that appears in every class that has Objects of this type.
`LocationAvailability`	*`BOOLEAN`*	When true, indicates that the server MUST honor the Location=1 parameter for all Objects o f this type. When false, indicates that the server does not support the Location=1 functionality.
`PostSupport`	*`BOOLEAN`*	When true, indicates that the server will honor the PostObject Transaction for this Object. When false, indicates that the server does not support the PostObject Transaction functionality for this Object.
`ObjectData`	*`RETSID`* **:** *`RETSNAME`*	The `ResourceID` and `ClassName` identifying a `METADATA-TABLE` that provides additional data about Objects described by this metadata. If an Object contains no additional data, this field MUST be empty.
`MaxFileSize`	*`POSITIVENUM`*	Indicates the maximum file size, in bytes, that is accepted by the server for Objects. The server MAY refuse any Object files in the PostObject Transaction that are larger than this size. A server MUST return an error when the file is rejected because it is too large. A server MAY return an http error code if an Object file bigger than this size is received. A server MAY return a RETS ReplyCode if an Object file bigger than this size is received. If a server does not have a maximum file size, this field MUST be empty.

Note: An Approved RCP is Related to this Section
Section 11.4.1 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 63 Object Data and Upload
- RCP 74 Location Availability in Object Metadata
- RCP 87 RETS 1.7.2 Errata Document

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.4.2 Lookup

This section describes the lookup tables that are referenced by the LookupName in the Table section. There MUST be a corresponding lookup table for every "LookupName".

COMPACT header tag: **`METADATA-LOOKUP`**

Table 11-24 Lookup Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Lookup metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource in which this table resides.

Table 11-25 Metadata Content: Lookup

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that never changes as long as the semantic definition of this entry remains unchanged.
`LookupName`	*RETSNAME*	The name of Lookup Table. There MUST be an entry for each LookupName value used in the Table metadata.
`VisibleName`	*1*64PLAINTEXT*	A description of the table that is human-readable.
`LookupTypeVersion`	*1*2DIGITS . 1*2DIGITS . 1*5DIGITS*	The latest version of this Lookup Table metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. The version number is advisory only.
`LookupTypeDate`	*RETSDATETIME*	The date on which any of the content of this Lookup was last changed. Clients MAY rely on this date for cache management.
`FilterID`	*RETSID*	The FilterID of an existing filter. If present, the range of valid LookupType values n this lookup is limited by the value of a parent lookup.
`NotShownByDefault`	*BOOLEAN*	If true, the server will, by default, not include the LookupType data of this lookup in any metadata request unless specifically asked to, using the LookupFilter argument in the GetMetadata Transaction. This field MUST be set to false (or empty) unless a FilterID is non-empty.

Note: An Approved RCP is Related to this Section
Section 11.4.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 60 Metadata Changes for Update
- RCP 78 Specification Errata Changes
- RCP 87 RETS 1.7.2 Errata Document

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.4.3 Lookup Type

This section describes the content of a lookup table that is referenced by the LookupName in the Table section. There MUST be a corresponding lookup table for every "Lookup" and "LookupMulti".

COMPACT header tag: **`METADATA-LOOKUP_TYPE`**

Table 11-26 Lookup Type Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Lookup Type metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The `ResourceID` for the resource in which this table resides.
`Lookup`	The `LookupName` for the class in which this table resides.

Table 11-27 Metadata Content: Lookup Type

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that never changes so long as the semantic definition of this entry remains unchanged. In particular, it should be managed so as to allow the client to detect changes to the `Value` .
`LongValue`	*1*128PLAINTEXT*	The value of the field as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; expected uses include displays on reports and other presentation contexts. This is the value that is returned for a COMPACT-DECODED or STANDARD-XML format request.
`ShortValue`	*1*128PLAINTEXT*	An abbreviated field value that is also localizable and human-readable. Use of this field is implementation-defined; expected uses include picklist values and other human interface elements.

`Value`	*1*128PLAINTEXT*	The value to be sent to the server when performing a search. This is the value that is returned for a COMPACT format request.
`StandardValue`	*1*128PLAINTEXT*	The value of the field as it is known in the RESO Data Dictionary. This value SHOULD be used for a COMPACT-DECODED or STANDARD-XML format response when StandardName=1 is used in the RETS Query. If this value is empty, then the LongValue MAY be used in the response. If the system does not support the Data Dictionary, the StandardValue will be the empty string. The following paragraphs describe the behaviour where the system supports the Data Dictionary. If the LOOKUP is not in the Data Dictionary, the StandardValue will be the empty string. If the LOOKUP is in the Data Dictionary, if the LOOKUP_TYPE is not in the Data Dictionary, the StandardValue will be the empty string. If the LOOKUP and the LOOKUP_TYPE are in the Data Dictionary, the StandardValue will match that of the Data Dictionary. A special case exists where the LOOKUP as defined in the Data Dictionary has a 'closed' enumeration. That is, all values of the enumeration are defined in the Data Dictionary and no additional values are permitted. In this case, if the LOOKUP is in the RESO Data Dictionary, the StandardValue MUST match a value in the Data Dictionary and CANNOT be empty. Please consult the appropriate Data Dictionary version to determine the set of 'closed' enumerations.

In a search response, the values of a LookupMulti field may include comma and other special characters. Values of a LookupMulti MAY be quoted as string-literal, as defined in Section 7.7.1. Values of a LookupMulti MUST be quoted as string-literal when they contain commas or other special characters.

Note: An Approved RCP is related to this section

Section 11.4.3 is related to the following approved RCP(s):

RETS 1.7.2

- RCP 66 Deprecate Lookup Types LookupBitmask and LookupBitstring
- RCP 72 LookupType String Length

RETS 1.8.0

- RCP 69 LookupType Value
- RCP 72 LookupType String Length
- RCP 82 LookupMulti Quoting Rule

RETS 1.9.0

- RCP 104 StandardValue for Enumerations

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.4.4 Search Help

This section describes the Search Help text tables that are referenced in the Table section. There MUST be a corresponding table entry for each Search HelpTextID referenced in the `METADATA-TABLE`.

COMPACT header tag: **`METADATA-SEARCH_HELP`**

Table 11-28 Search Help Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Search Help metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The `ResourceID` for the resource to which this metadata table applies.

Table 11-29 Metadata Content: Search Help

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that never changes so long as the semantic definition of this entry remains unchanged.
`SearchHelpID`	*RETSNAME*	A unique ID for the help text. This ID is referenced as the SearchHelpID in section 11.3.2
`Value`	*1*1024TEXT*	The value to be displayed to the user.

11.4.5 Edit Mask

This section describes the Edit Mask table that is referenced in the Table section. There MUST be a corresponding table entry for each Search `EditMaskID` referenced in the `METADATA-TABLE`.
A Regular Expression is used to define the edit mask. Table 11-28 describes te structures that make up RETS regular expressions.

COMPACT header tag: **`METADATA-EDITMASK`**

Table 11-33 EditMask Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Edit Mask metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The `ResourceID` for the resource to which this metadata table applies.

Table 11-34 Metadata Content: Edit Mask

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that remains unchanged so long as the semantic definition of this field remains unchanged.
`EditMaskID`	*RETSNAME*	A unique ID for the Edit Mask. This ID is referenced as the EditMaskID in section 11.3.2
`Value`	*1*256TEXT*	The Regular Expression to be used.

RETS Regular Expression Specification

RETS regular expressions are a subset of POSIX 1003.2 extended regular expressions [12], supporting the metacharacters in Table 11-28.
Table 11-35 RETS Regular Expression Metacharacters

Metacharacter	Function
. (period)	Matches any single character
*****	Matches zero or more of the preceding pattern
+	Matches one or more of the preceding pattern
?	Matches zero or one of the preceding pattern
|	Alternation: used between two subpatterns, matches either the one to its left or the one to its right.
( ) parentheses	Grouping: causes the enclosed pattern to be treated as atomic. Parentheses may not be nested; that is, only one level of grouping is required.
{*min*[**,***max*]**}** (braces)	Quantifier: matches at least *min* and at most *max* of the preceding pattern, where *min* and *max* are both nonnegative integer values. If *max* is omitted, matches exactly *min* of the preceding pattern.
[ ] brackets	Character class: matches any of the characters contained in the brackets. Except for the circumflex, described below, and the closing bracket, characters within a character class are never treated as metacharacters.
^ (circumflex)	Used as the first character of a character class, reverses the sense of the character class; for example, [^0] matches any character except a "0".

-	Operates only within brackets. Except as the first or last character, denotes a range of characters on the default host collating sequence. For example, [0-9] matches any digit. When - is the first or the last character, it is treated as a member of the character class.
\	Escape: treats the following character as an ordinary character rather than a metacharacter. For example, * matches a single asterisk. The \ character itself must be escaped. The escape character is not needed within character classes.

The following is a simple example:

```
[0-9]+[a-fA-F][1-8][A]?[0-9]{2}[A-C]{1,3}
```

One or more digits, followed by an upper or lower case letter A - F, followed by a digit 1 – 8, optionally followed by one letter A, followed by two digits 0 – 9, followed by between one and three of the letters A – C.
A phone number example:

```
[0-9]{3}[0-9]{4}
```

11.4.6 Update Help

This section describes the Update Help Text tables that are referenced in the Update Type section of the document. There MUST be a corresponding table entry for each Update Help Text ID referenced in any of the `METADATA-UPDATE_TYPEs`.

COMPACT header tag: **`METADATA-UPDATE_HELP`**

Table 11-36 Update Help Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Update Help metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource to which this metadata segment belongs.

Table 11-37 Metadata Content: Update Help

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that remains unchanged so long as the semantic definition of this entry remains unchanged.
`UpdateHelpID`	*RETSNAME*	A unique ID for the help text. This ID is referenced as the UpdateHelpID in section 11.4.6 .
`Value`	*1*1024TEXT*	The value to be displayed to the user.

11.4.7 Validation Expression

This section describes the ValidationExpression table that is referenced in Section 11.3.4 . There MUST be a corresponding table entry for each ValidationExpressionID referenced in the set of `METADATA-UPDATE_TYPE` for a Resource.

The table contains expressions that are to be evaluated when a field value is entered by the user. Expressions in the list MUST be evaluated in the order in which they appear in the list.

Note that the *`KeyField`* for an update (see table 11-17) has a specific purpose. If there is a validation expressions associated with that field, it must be evaluated even if the field itself is not part of the update request (in ADD update). If the expression for this field evaluates as REJECT, the whole record is rejected and the client SHOULD NOT send the Update request.

COMPACT header tag: **`METADATA-VALIDATION_EXPRESSION`**

Table 11-42 Validation Expression Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Validation Expression metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.

`Resource`	The ResourceID for the resource to which this metadata table applies.

Table 11-43 Metadata Content: Validation Expression

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that remains unchanged so long as the semantic definition of this entry remains unchanged.
`ValidationExpressionID`	*RETSNAME*	A unique ID for the ValidationExpression. This ID is referenced as the ValidationExpression in Section 11.3.4 .
`ValidationExpressionType`	*1*32ALPHANUM*	A validation expression type from Table 11-44.
`Value`	*1*512TEXT*	The test expression to be evaluated.
`Message`	**1024TEXT*	Message to be shown to the user if a field is rejected, or a warning is issued as a result of this validation expression.
`IsCaseSensitive`	*BOOLEAN*	If true, the string comparisons in the expressions are case sensitive.

11.4.7.1 Validation Expression Types and Data Types

There are several types of validation expressions, each introduced by a keyword preceding the expression:
Table 11-44 Validation Expression Types

Keyword	Type	Purpose
ACCEPT	*BOOLEAN*	If the expression is true, the field value is considered accepted without further testing. Immediately following SET expressions MUST be executed. If the expression is false, following validation expressions MUST be executed. If the expression is ERROR (evaluation failed) in client, the client SHOULD act as if the field was accepted, allowing the server to make the final decision.
REJECT	*BOOLEAN*	If the expression is true, the field value is considered rejected without further testing. Subsequent SET expressions MUST NOT be evaluated. If the expression is false, following validation expressions MUST be executed. If the expression is ERROR, evaluation failed in client, the client SHOULD act as if the field was accepted, allowing the server to make the final decision.
WARNING	*BOOLEAN*	If the expression is true, the client should show a warning message to the user, and if the warning is okayed by the user, include a Warning-Response in the UPDATE request. If the user does not okay the warning, the field is considered rejected and following SET validation expressions MUST NOT be evaluated. If the expression is false, the following validation expressions MUST be evaluated.
SET	Assignment	The expression MUST begin with a field name and an equal sign ("="). The following expression is evaluated and the result stored in the designated field.
SET_DEFAULT	The appropriate data type of the assigned field	This expression MUST be executed ONLY when a NEW record is created. Supersedes the default value as indicated in the Update Metadata.
SET_REQUIRED	*BOOLEAN*	Expressions of this type are designed to evaluate an expression and set the field that the rule is applied on to Required if the expression returns true and to Non Required if the expression returns false.
SET_READ_ONLY	*BOOLEAN*	Expressions of this type are designed to evaluate an expression and set the field that the rule is being applied on to Read Only if the expression returns true and to Updateable if the expression returns false. The client application is expected to lock the value of the field the rule is being executed on to the value at the time the **`SET_REQUIRED`** expression is evaluated.
RESTRICT_PICKLIST	List of*CHAR*	Expressions of this type are designed to return one or more LOOKUP values that are to be removed from the LOOKUP list that is defined for the field the rule is being executed on. This is always the entire set of values to remove from the lookup. In other words, if this returns a blank list or .EMPTY., the entire set of LOOKUP values is to be displayed. The value of this expression MUST be a <List>, rather than <Exp>, as defined in 11.4.9.1. All members of the list MUST be of the same type as the type of the field the rule is being executed on.

SET_PICKLIST	List of*CHAR*	Expressions of this type are designed to return one or more LOOKUP values that are to be used in the LOOKUP list that is defined for the field the rule is being executed on. The value of this expression MUST be a <List>, rather than <Exp>, as defined in 11.4.9.1. Every member of the list MUST exist in the Lookup list as defined in the metadata for the field the rule is being executed on.
SET_DISPLAY	*BOOLEAN*	Expressions of this type are designed to allow a client to make fields visible or invisible based on the evaluation of an expression. The result of this expression has no effect on whether a field is READ ONLY or not.

Table 11-45 Validation Expression Data Types

Each validation expression has one of these data types:

CHAR	Any String of ASCII characters
INT	Any integer (tiny, small, int, long)
FLOAT	decimal number with fraction part
TIME	time, date, datetime.
BOOLEAN	true or false, as defined in Section 2.4
LIST	list of several values of the same type. The type of the values may be any of those above
EMPTY	missing data (similar to NULL in database systems)
ERROR	this is the type of an expression which cannot be parsed

Note: An Approved RCP is Related to this Section
Section 11.4.9 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 65 Session information tokens
- RCP 61 Validation Expression Replacement

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.4.7.2 Validation Expression BNF Representation

Literal entry is denoted by quotations around the term and the term and quotations will be in blue. The quotations are not part of the literal.

Exp	::= *OrExp*
OrExp	:: = *AndExp* *(".OR." *AndExp*)
AndExp	:: = *NotExp* *(".AND." *NotExp*)
NotExp	:: = ".NOT." *NotExp* | *EqExp*
EqExp	:: = *CmpExp* | *CmpExp* "=" *CmpExp* | *CmpExp* "!=" *CmpExp*
CmpExp	:: = *CntExp* | *CntExp* "<=" *CntExp* | *CntExp* "=>" *CntExp* | *CntExp* "<" *CntExp* | *CntExp* ">" *CntExp*
CntExp	:: = *SumExp* | *SumExp* ".CONTAINS." *SumExp* | *SumExp* ".IN." *List*
SumExp	:: = *ProdExp* *(("+" | "-" | *Concat*) *ProdExp*)

ProdExp	:: = AtomExp *(("*" | "/" | ".MOD.") AtomExp)
AtomExp	:: = LPAREN Exp RPAREN | Value | FuncExp
FuncExp	:: = Func LPAREN Param *(Param) RPAREN
Func	:: = ALPHA *(ALPHANUM)
Param	:: = Exp
Value	:: = SpecValue | CharValue | IntValue |FloatValue | TimeValue | FieldName
Concat	::= "||"
FieldName	::= "["0*1 **"LAST"** RETSNAME "]"
SpecValue	::= DOT RETSNAME DOT ; spec value is further constrained - see table 11-46
CharValue	::= SQUOTE *PLAINTEXT SQUOTE | QUOTE *PLAINTEXT QUOTE
TimeValue	::= "#" RETSDATETIME "#"
IntValue	::= 0*1("+" | "-") 1*(DIGIT)
FloatValue	::= IntValue "." * (DIGIT)
LPAREN	::= %x28 ; ASCII left parenthesis character (
RPAREN	::= %x29 ; ASCII right parenthesis character)
SQUOTE	::= %x27 ; ASCII single quote character '
QUOTE	::= %x22 ; ASCII double quote character "
DOT	::= %x2e ; ASCII period character .
List	::= LPAREN Exp *("," Exp) RPAREN | LPAREN RPAREN

The value of a Validation Expression MUST conform to the `Exp` syntax in the grammar above, except for RESTRICT_PICKLIST and SET_PICKLIST expressions, whose value MUST conform to the `List` syntax. Any expression with keyword starting with “X-” MAY have a `List` value as well.

The text in `CharValue` must not include the (single or double) quote used to delimit the value.

`TimeValue` must be expressed in the form described in section 2.4, Atoms and Primitives. This is the ISO8601 format constrained for use by RETS and is enclosed in hashmarks(#) (ex. #2007-09-11T14:30:00#).

A `FieldName` is a bracketed name of a field belonging to the same class as the field to which this expression is attached, and has a type of that field specified by the metadata. If used with the LAST keyword, its value is the value of the field as it was in the database before the current updates took place. If used without LAST, the updated value of the field MUST be used."

A `TimeValue` has TIME type.
A `CharValue` has CHAR type.
A `IntValue` has INT type.
A `FloatValue` has FLOAT type.

The `SpecValue` uses well-known names defined in the table Validation Expression Special Operand Tokens (table 11-39. See the table for valid values. These values may change between versions of the standard.

11.4.7.3 Validation Expression Special Operand Tokens

Table 11-46 Validation Expression Special Operand Tokens (SpecValue)

Token	Data Type	Description
`.EMPTY.`	EMPTY	A value that matches an empty or all-blank field. Supplies an empty (zero-length) field when used in a SET expression.
`.TRUE.`	BOOLEAN	Boolean value of TRUE (1)
`.FALSE.`	BOOLEAN	Boolean value of FALSE (0)
`.TODAY.`	TIME	The current date.
`.NOW.`	TIME	The current time.
`.ENTRY.`	type of the current field	The current field text.
`.OLDVALUE.`	type of the current field	The value that was in the field as returned from the host in the search operation. If the field is new, `.OLDVALUE.` is **`.EMPTY.`**
`.USERID.`	CHAR	The value of the user-id field returned in the Login transaction (Section 4.9).
`.USERCLASS.`	CHAR	The value of the user-class field returned in the Login transaction (Section 4.9).
`.USERLEVEL.`	CHAR	The value of the user-level field returned in the Login transaction (Section 4.9).
`.AGENTCODE.`	CHAR	The value of the agent-code field returned in the Login transaction (Section 4.9).
`.BROKERCODE.`	CHAR	The value of the broker-code field returned in the Login transaction (Section 4.9).
`.BROKERBRANCH.`	CHAR	The value of the broker-branch field returned in the Login transaction (Section 4.9).
`.UPDATEACTION.`	CHAR	Name of the UpdateAction for which this validation is performed.
`.any.`	(see description)	If the name of the SpecValue (stripped of the first and last dot) is equal to a name of one of the info-token-keys returned as part of the Login response, then the type and value of this SpecValue is defined by that info-token-key. If no such info-token-key exists, the value is ERROR.

Note: An Approved RCP is Related to this Section
Section 11.4.9 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 65 Session information tokens

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.4.7.4 Validation Expression Functions and Operators

A Validation Expression may use Functions. We define <FuncExp> as a function with parameters. The following functions are defined:
Table 11-47 Validation Expression Functions

Function	Parameter Types	Type
BOOL	BOOLEAN or CHAR	BOOLEAN
CHAR	Any, except for FLOAT	CHAR
CHARF	FLOAT, INT	CHAR
TIME	TIME or CHAR	TIME
DATE	TIME or CHAR	TIME
INT	INT or FLOAT or BOOL or CHAR	INT
FLOAT	INT or FLOAT or BOOL or CHAR	FLOAT

SUBSTR	CHAR,INT,INT	CHAR
STRLEN	CHAR	INT
LOWER	CHAR	CHAR
UPPER	CHAR	CHAR
IIF	BOOLEAN,any,any	any
YEAR	TIME	INT
MONTH	TIME	INT
DAY	TIME	INT
WEEKDAY	TIME	INT

The BOOL, CHAR, TIME, DATE, INT and FLOAT functions are used just to change a type of expression. The DATE and TIME functions are synonyms. Note that any of these functions may fail (return an ERROR value) if the parameter can not be transformed to the appropriate type.

In conversion from BOOLEAN to INT or FLOAT, .TRUE. is converted to 1 and .FALSE. is converted to 0. Casting FLOAT to INTEGER discards the fractional part.

When converting to CHAR, BOOL values are represented as “0” and “1”, TIME values are represented using format defined in RFC 1123 with digital timezone, INT values are represented with no leading zeroes.

When converting from CHAR to BOOL, values “0”,”1”,”YES”,”NO”,”TRUE” and “FALSE” (no matter what the case) MUST be understood.

When converting from CHAR to TIME, any RFC 1123 –compliant format MUST be understood. A leading and/or trailing # MUST be removed before conversion.

When converting from CHAR to INT or FLOAT, usual formats MUST be understood. Scientific format (with exponent) MUST NOT be understood. FLOAT numbers with empty integral part (.5, -.4) MUST be understood as long as there is at least one digit after the decimal point.

The CHARF function converts a Float number, and in the second parameter specifies how many decimal digits MUST appear after the point.

The SUBSTR function returns a substring of its first parameter. Second parameter is a starting position of the substring, third parameter is the ending position of the substring. Positions are 1-based.

The STRLEN function returns the length if its parameter.

The UPPER function returns its parameter upper-cased.

The LOWER function returns its parameter lower-cased.

The IIF function return the value of its second parameter if the first parameter evaluates to true, or the value of its third parameter otherwise. Types of second and third parameter must be same, and it is the type of the result.

The YEAR,MONTH,DAY and WEEKDAY parse the date part of TIME value. They return values ranging from 1 to the appropriate maximum. WEEKDAY returns 1 for Sunday, 2 for Monday etc.

Other functions may be defined later (HOUR and MINUTE are first candidates). If a server uses a function the client does not recognize, the client MUST evaluate it as ERROR.

A Validation Expression may have Operations applied to the parameters. The Operators may be applied on certain types that determine the result type. The input types and resulting output type is defined in the Validation Expression Operators Table below.
Table 11-48 Validation Expression Operators

Operator	Left Operand	Right Operand	Result	Meaning
`.MOD.`	INT	INT	INT	Arithmetic MODULO operation
/, *	INT	INT	INT	Integer division and multiplication
/, *	INT	FLOAT	FLOAT	Division and multiplication
/, *	FLOAT	INT	FLOAT	Division and multiplication
/, *	FLOAT	FLOAT	FLOAT	Division and multiplication
+,-	INT	INT	INT	Integer addition and subtraction.
+,-	INT	FLOAT	FLOAT	Addition and subtraction.
+,-	FLOAT	INT	FLOAT	Addition and subtraction.

+,-	FLOAT	FLOAT	FLOAT	Addition and subtraction.
+	FLOAT	TIME	TIME	Time shift.
+	TIME	FLOAT	TIME	Time shift.
-	TIME	FLOAT	TIME	Time shift.
-	TIME	TIME	FLOAT	Time shift.
||	CHAR	CHAR	CHAR	String concatenation.
`.CONTAINS.`	CHAR	CHAR	BOOLEAN	String containment. The operation is TRUE if the left operand contains the right operand as a substring anywhere within it.
`.IN.`	Any	List of operands, all of the same type as the left operand	BOOLEAN	List inclusion. The operation is TRUE if the left operand is equal to any member of the list.
`.AND.`	BOOLEAN	BOOLEAN	BOOLEAN	A Boolean operator that takes two Boolean operands, and whose value is TRUE if and only if both of its operands are TRUE.
`.OR.`	BOOLEAN	BOOLEAN	BOOLEAN	A Boolean operator that takes two Boolean operands, and whose value is TRUE if either of its operands is TRUE.
`.NOT.`	BOOLEAN	BOOLEAN	BOOLEAN	A Boolean operator that takes a single Boolean operand and returns its inverse.
=, **!=**	Any	Same as left	BOOLEAN	Equality.
<,>,<=,>=	INT,FLOAT	INT,FLOAT	BOOLEAN	Numeric comparison.
<,>,<=,>=	TIME	TIME	BOOLEAN	Date and time comparison
<,>,<=,>=	BOOLEAN	BOOLEAN	BOOLEAN	Boolean comparison (TRUE > FALSE).

Arithmetic operations between dates use number of days as the FLOAT parameter (or result); e.g. 0.25 represents a time span of 6 hours.

String operations are case sensitive or not, based on the IsCaseSensitive field in the expression's metadata.

Any operation with an ERROR argument MUST evaluate to ERROR. An EMPTY value may be compared (=,!=) against any value.

Appropriate casting functions (BOOL, CHAR, TIME, INT, FLOAT) MUST be applied to the parameters. If a function or an operator is applied to a datatype different than shown in the above tables, the expression MUST evaluate to ERROR.

11.4.8 Validation External

This section describes the Validation External tables that are referenced in the Update Type section of the document. There MUST be a corresponding Validation External table for each one referenced in any of the `METADATA-UPDATE_TYPEs` for the Resource.

COMPACT header tag: **`METADATA-VALIDATION_EXTERNAL`**

Table 11-49 Validation External Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Validation External metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource to which this metadata table applies.

Table 11-50 Metadata Content: Validation External

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that remains unchanged so long as the semantic definition of this entry remains unchanged.
`ValidationExternalName`	*RETSNAME*	The unique name of this Validation External. Each Name in the Update Type ValidationExternalName field MUST have a definition.
`SearchResource`	*RETSNAME*	The `ResourceID` of the Resource to be searched from 11.2.2 .
`SearchClass`	*RETSNAME*	The `ClassName` within the Resource to be searched from 11.3.1 .

`Version`	*1*2DIGITS"."1*2DIGITS"."1*5DIGITS*	The latest version of this Validation External metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. The version number is advisory only.
`Date`	*RETSDATETIME*	The date on which any of the content of this Validation External was last changed. Clients MAY rely on this date for cache management.

11.4.9 Validation External Type

This section describes the content of the Validation External Type tables that are referenced in the Table section of the document. There MUST be a corresponding Validation External Type table for each one referenced in the `METADATA-UPDATE_TYPEs` for the Resource.

The Validation External Type provides lists of search, display, and results fields. The Validation External may be used for several cases: 1) The database involved is too large or dynamic to be provided as a standard lookup (e.g. Tax). 2) There are business rules that can only be enforced on the server (e.g. expiration dates). 3) The content of a field populates fields from another database (e.g. `Sale_agent_name` , `Sale_office_name` , `Sale_office_id` from `Sale_agent_id`).

COMPACT header tag: **`METADATA-VALIDATION_EXTERNAL_TYPE`**

Table 11-51 Validation External Type Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Validation External Type metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource to which this metadata table applies.
`ValidationExternalName`	The `ValidationExternalName` to which this entry type applies.

Table 11-52 Metadata Content: Validation External Type

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that remains unchanged so long as the semantic definition of this entry remains unchanged.
`SearchField`	*1*512PLAINTEXT*	A comma separated list of valid fields using `SystemName` from Section 11.3.2 .
`DisplayField`	*1*512PLAINTEXT*	A comma separated list of valid fields using `SystemName` from Section 11.3.2 .
`ResultFields`	*1*1024PLAINTEXT*	A comma separated list of valid field pairs joined by = (equal) the first is a target field in the table being updated and the second is a source field in the table being searched. The fields use a `SystemName` from Section 11.3.2 .

11.5 Metadata Format for Presentation Elements

Metadata Presentation elements may be used to build applications that use hints provided by the Server to display, group and organize the data.

- 11.5.1 Column Group Set
- 11.5.2 Column Group
- 11.5.3 Column Group Control
- 11.5.4 Column Group Table
- 11.5.5 Column Group Normalization

11.5.1 Column Group Set

This metadata defines a tree structure which should be used to render the data in any GUI system that is designed in order to satisfy the display requirements of an MLS.

Table 11-53 Column Group Set Metadata Compact Header Attributes

Attribute	Content

`Version`	This is the version of the Column Group Set metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for *RETSDATETIME*.
`Resource`	The ResourceID for the resource that this column group set belongs. The content is *RETSID*
`Class`	The ClassID for the class that this column group set belongs. The content is *RETSID*

COMPACT header tag: **METADATA-COLUMN_GROUP_SET**

Table 11-54 Metadata Content: Column Group Set

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that never changes as long as the semantic definition of this entry remains unchanged. In particular, it should be managed so as to allow the client to detect changes to the ColumnGroupSetName.
`ColumnGroupSetName`	*RETSID*	The name that uniquely identifies this Column Group Set within the Class.
`ColumnGroupSetParent`	*RETSID*	The ColumnGroupSetName of the Parent Column Group Set. If not specified, this Column Group Set is the top node in the tree.
`Sequence`	*1*5DIGIT*	The sequence that this Column Group Set is to be displayed in within its parent group.
`LongName`	*RETSNAME*	The name of the Column Group Set as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this Column Group Set when it appears on a report.
`ShortName`	*RETSNAME*	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined; it is expected that clients will use this field in human-interface elements such as lookups.
`Description`	*1*256PLAINTEXT*	A brief description of the purpose for this Column Group Set.
`ColumnGroupName`	*RETSID*	The name of the Column Group that is to be displayed in this Column Group Set. If not specified, this Column Group Set is to be treated as a node in the tree that displays no data. The ColumnGroupName must exist in the Column Group metadata for this Class.
`PresentationStyle`	*1*32PLAINTEXT*	One of the following values:
	Edit	Basic Edit Block displayed in PresentationColumns number of columns.
	Matrix	Expected to be displayed using Normalization Grid.
	List	Show one record per row.
	Edit List	Show one record per row and allow the records to be added, edited and deleted.
	GIS Map Search	Special Case: Can only have 2 columns in Column Group. First column is Latitude and Second column is Longitude. These columns are expected to be filled in with results from GIS Map Search.
	URL	Indicates that this is to simply go to the specified URL, a ColumnGroup name MUST not be specified and a URL MUST be specified for this PresentationStyle.
`URL`	*1*256TEXT*	Indicates a URL that is to be accessed using this entry instead of a standard Column Group. You may not specify a ColumnGroupName and a URL. The URL may be formed with place-holders surrounded by the '[' and ']' characters so that a substitution for any valid SystemName within the class being displayed, Info Tokens from the Login Response or Validation Expression Special Operand Tokens as specified in Table 11-20. To differentiate between SystemNames and tokens, an additional character '.' is used to surround the tokens. Example: [http://www.example.com/agent?Agent=[.AGENTCODE.]] &Listing=[ListingID]

`ForeignKeyID`	*RETSID*	The identifier of the Foreign Key that is to be displayed in this ColumnGroupSet. If specified, the ForeignKeyID MUST exist in the METADATA-FOREIGNKEY metadata and the Parent MUST be the Property and Class of this ColumnGroupSet. When this is specified, it means that a multi-row block is expected to be displayed to the user within which he can Add, Edit or Delete records of the Child Resource and Class that is specified. Furthermore, the ChildSystemName field should always be filled from data found in the ParentSystemName field to provide for a proper Master/Detail relationship.

Notes: It is important to note that only one of ColumnGroupName, ForeignKeyId or URL may contain data. These three fields are mutually exclusive.

Note: An Approved RCP is Related to this Section
Section 11.4.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 60 Metadata Changes for Update

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.5.2 Column Group

This metadata defines grouping element which should be used to group columns together in any GUI system that is designed in order to satisfy the display requirements of an MLS.

Table 11-55 Column Group Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Column Group metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource that this column group belongs. The content is RETSID
`Class`	The ClassID for the class that this column group belongs. The content is RETSID

COMPACT header tag: **`METADATA-COLUMN_GROUP`**

Table 11-56 Metadata Content: Column Group

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that never changes as long as the semantic definition of this entry remains unchanged. In particular, it should be managed so as to allow the client to detect changes to the ColumnGroupName.
`ColumnGroupName`	*RETSID*	The name that uniquely identifies this Column Group within the Class.
`ControlSystemName`	*RETSID*	The SystemName of the Table Metadata that identifies the data element that is used to control the display of this Column Group.
`LongName`	*RETSNAME*	The name of the Column Group as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this Column Group when it appears on a report.
`ShortName`	*RETSNAME*	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined; it is expected that clients will use this field in human-interface elements such as lookups.
`Description`	*1*256PLAINTEXT*	A brief description of the purpose for this Column Group.

Note: An Approved RCP is Related to this Section
Section 11.4.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 60 Metadata Changes for Update

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.5.3 Column Group Control

This metadata defines the valid ranges of values that the specified SystemName may have that control the display of the Column Group. If the SystemName contains any of the values that fall within the ranges specified in this table, the Column Group may be displayed. If it does not, the Column Group should not be displayed. The data is returned as a list of high and low values that determine whether the Column Group should be displayed.

Table 11-57 Column Group Control Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Column Group Control metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource that this column group control belongs. The content is *`RETSID`*
`Class`	The ClassID for the class that this column group control belongs. The content is *`RETSID`*
`ColumnGroup`	The ColumnGroup MetadataEntryID for the class that this column group control belongs. The content is *`RETSID`*

COMPACT header tag: **`METADATA-COLUMN_GROUP_CONTROL`**

Table 11-58 Metadata Content: Column Group Control

Field Name	Content Type	Description
`MetadataEntryID`	*`RETSID`*	A value that never changes as long as the semantic definition of this entry remains unchanged. In particular, it should be managed so as to allow the client to detect changes to an individual pair of High/Low values.
`LowValue`	*`1*64ALPHANUM`*	The minimum value that the ControlSystemName field of the ColumnGroup is allowed to have in order to display the ColumnGroup. It is expected that the actual data type returned is interpreted as per the data type of the ControlSystemName of the ColumnGroup.
`HighValue`	*`1*64ALPHANUM`*	The maximum value that the ControlSystemName field of the ColumnGroup is allowed to have in order to display the ColumnGroup. It is expected that the actual data type returned is interpreted as per the data type of the ControlSystemName of the ColumnGroup. If the restricting data is not a range, then HighValue may be left blank.

Note: An Approved RCP is Related to this Section
Section 11.4.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 60 Metadata Changes for Update

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.5.4 Column Group Table

This metadata defines the set of SystemNames that are to be displayed within a Column and the order in which they are to be displayed.

Table 11-59 Column Group Table Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Column Group metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.

`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource that this column group belongs. The content is RETSID
`Class`	The ClassID for the class that this column group belongs. The content is RETSID
`ColumnGroup`	The ColumnGroup MetadataEntryID for the column group that this column group table belongs. The content is RETSID

COMPACT header tag: **`METADATA-COLUMN_GROUP_TABLE`**

Table 11-60 Metadata Content: Column Group Table

Field Name	Content Type	Description
`MetadataEntryID`	*RETSID*	A value that never changes as long as the semantic definition of this entry remains unchanged.
`SystemName`	*RETSID*	The SystemName of the field that is to be displayed in the ColumnGroup. This MUST be a valid SystemName for this Class. A SystemName MUST be unique within the ColumnGroup. This MUST not be specified if a ColumnGroupSetName is specified. Both `SystemName` and `ColumnGroupSetName` may be blank. In this case, the Client may use this as a spacer.
`ColumnGroupSetName`	*RETSID*	The name of a ColumnGroupSet to display in place of a single field. It is expected that this is a ColumnGroupSet that does not display a large number of columns. This MUST not be specified if a SystemName is specified. The ColumnGroupSet MUST not contain a ColumGroup that also specifies a ColumnGroupSetName in the COLUMN_GROUP_TABLE metadata. Both `SystemName` and `ColumnGroupSetName` may be blank. In this case, the Client may use this as a spacer.
`LongName`	*1*128PLAINTEXT*	The name of the Column Group Table (data field) as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this Column Group when it appears on a report.
`ShortName`	*1*128PLAINTEXT*	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined; it is expected that clients will use this field in human-interface elements such as lookups.
`DisplayOrder`	*1*5DIGIT*	The order within the ColumnGroup that this SystemName is to be displayed in. DisplayOrder values MAY contain gaps and may have the same value as other columns. If multiple columns have the same value, the client SHOULD display the columns in Alphabetical order.
`DisplayLength`	*1*5DIGIT*	The number of characters to allow when displaying data for this column.
`DisplayHeight`	*1*5DIGIT*	The number of rows to display the data in. A value greater than one in this column implies a multi-line data entry field of DisplayLength width. If users enter data into this field that is longer than will fit within this text box, it is expected that the field will scroll to allow further data entry.
`ImmediateRefresh`	*BOOLEAN*	A truth value which indicates whether a change to this field by the user should cause an automatic GUI refresh. This is primarily intended for use

Note: An Approved RCP is Related to this Section
Section 11.4.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 60 Metadata Changes for Update

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

11.5.5 Column Group Normalization

This metadata defines a grid that can be used by a client to display related fields in a manner more appropriate for data entry.

Table 11-61 Column Group Normalization Metadata Compact Header Attributes

Attribute	Content
`Version`	This is the version of the Column Group metadata. The convention used is a "<*major*>.<*minor*>.<*release*>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.

`Date`	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
`Resource`	The ResourceID for the resource that this column group belongs. The content is RETSID
`Class`	The ClassID for the class that this column group normalization belongs. The content is RETSID
`ColumnGroup`	This value MUST be a ColumnGroupName found in the ColumnGroup metadata for this Class. It is the ColumnGroup for which the grid applies. The content is RETSID

COMPACT header tag: **`METADATA-COLUMN_GROUP_NORMALIZATION`**

Table 11-62 Metadata Content: Column Group Normalization

Field Name	Content Type	Description
`MetadataEntryID`	*`RETSID`*	A value that never changes as long as the semantic definition of this entry remains unchanged.
`TypeIdentifier`	*`RETSID`*	Y Axis – Row Label – The Label that is to be displayed on the left side of the screen that identifies the Type of data that the user is entering.
`Sequence`	*`1*5DIGIT`*	Y Axis – Row Sequence – The Sequence number that is to be displayed on the left side of the screen after the TypeIdentifier. This itemizes the Type of data that the user is entering.
`ColumnLabel`	*`RETSID`*	X Axis – Column Label – This is the label that is to appear at the top of the screen for data within this column. It is expected that all data in this grid with the same ColumnLabel be displayed in the same column on the screen.
`SystemName`	*`RETSID`*	The SystemName of the field that is to be displayed in this position in the Grid for the ColumnGroup. This MUST be a valid SystemName for this Class and MUST be within the ColumnGroupTable of the ColumnGroup. Fields that appear within a ColumnGroup, but not within the Normalization for the ColumnGroup, are to be treated as separate data entry fields that are not part of the grid. The SystemName MUST be unique within the ColumnGroup.

Note: An Approved RCP is Related to this Section
Section 11.4.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 60 Metadata Changes for Update

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Section 12 - GetMetadata Transaction

The GetMetadata transaction is used to retrieve structured information known as metadata related to the system entities. Metadata requested and returned from this transaction are requested and returned as MIME media types.

12.1 Required Request Header Fields

There are no additional required client header fields.

12.2 Required Request Arguments

`Type`	`::=` <A grouping of related metadata elements (see Section 11)>

The type of metadata being requested. The Type MUST begin with METADATA and MAY be one of the defined metadata types (see Section 11).

`ID`	`::=` *metadata-id*`[: metadata-id]`
metadata-id	`::=` *1*ALPHANUM | **

Metadata is organized hierarchically. Each level specifies in its first field an identifier for the metadata contained within that level (e.g. for the Resource level: ResourceID-Agent, Property, etc. for the Lookup level: LookupName-Status, Area, etc.). This identifier can be used to restrict requests to the Type metadata contained within specific instances of higher levels. If the last metadata-id is 0 (zero), then the request is for all Type metadata contained within that level; if the last metadata-id is "*", then the request is for all Type metadata contained within that level and all metadata Types contained within the requested Type. This means that for a metadata-id of `METADATA-SYSTEM`, for example, the server is expected to return *all* metadata.

`NOTE:` The metadata-id for `METADATA-SYSTEM` and `METADATA-RESOURCE` must be 0 or `*`.

12.3 Optional Request Arguments

12.3.1 Format

`Format`	`=` **`COMPACT`** | **`STANDARD-XML`** | **`STANDARD-XML:`***version*
version	`::=` <RETS metadata public identifier>

"COMPACT" means a table descriptor, field list `<COLUMNS>` followed by a delimited set of the data fields. See Section 11 for more information on the COMPACT formats. "STANDARD-XML" means an XML presentation of the data in the format defined by the RETS Metadata XML DTD. Servers MUST support all formats. If the format is not specified, the STANDARD-XML presentation will be returned.

When the client requests the STANDARD-XML representation, it MAY also specify the public identifier of the DTD that it expects. The server MUST support the current version and SHOULD support the prior version.

12.3.2 LookupFilter

Some servers may use lookups that list hundreds of values (like community names or street names). Such lookups will make the full metadata very large. Therefore clients may opt to get metadata with limited sets of values.

LookupFilterArgument	::= **LookupFilter =** *Filter* *(, *Filter*) | **LookupFilter = -1** | **LookupFilter =** *

Filter	::= *FilterID* **=** *ParentValue*
FilterID	::= <A filter value from Table 11-9>
ParentValue	::= * | **-1** | 1*128ALPHANUM

If the LookupFilterArgument is present, the server that implements this extension MUST limit lookup data in the response. Specifically, LookupType data for a Lookup whose ParentValue FilterID is included in the LookupFilterArgument MUST be limited to the child values for the requested *ParentValue*. Also, FilterType data for a filter included in the LookupFilterArgument MUST be limited to those with requested *ParentValue*.

The asterisk ***** in place of *Filter* specifies that all values should be listed. The value **-1** indicates that no values should be listed.

If the LookupFilter argument is specified as **-1** , the server MUST NOT send any LookupType data for lookups that have non-empty FilterID, nor any FilterType data at all.

If the LookupFilter argument is specified as *****, the server MUST send all requested LookupTypes without limitations.

The LookupFilterArgument is meant to be used with requests for LookupType metadata, Filter metadata, or any metadata with ID ending with asterisk *****.

Some examples will assist in understanding the use of this feature.

A request for all the counties in the State of California:
LookupFilter=CountyByState=CA

A request for all the counties in the States of California and Tennessee:
LookupFilter=CountyByState=CA,CountyByState=TN

A request for all the lookups without filtration (Equivalent to omitting the **LookupFilter** argument):
LookupFilter=*

A request to omit any lookup that can be filtered
LookupFilter=-1

A request for all counties, cities in Marine County, and omit the streets by city lookup
LookupFilter=CountyByState=,CityByCounty=Marine,StreetByCity=**-1**

Note: An Approved RCP is Related to this Section
Section 7.4.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 60 Metadata Changes for Update

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

12.4 Required Response Header Fields

In addition to the other Required Server Header Fields specified in Section 3.3 the following response header fields are required.

Content-Type	The media type of the underlying data. The server MUST return this field in all replies. This field MUST be set to the type of media returned.

Content-Type	::= **Content-Type :** *type* **/** *subtype*

Example: `Content-Type: text/xml`

12.5 Required Response Arguments

There are no required response arguments.

12.6 Optional Response Arguments

There are no optional response arguments.

12.7 Metadata Response Body Format

The body of the metadata response has the following format when replying to a request with the format set to "COMPACT":

```
<RETS 1*SP ReplyCode=quoted-reply-code 1*SP
ReplyText=quoted-string SP > CRLF
[metadata-segment]
[rets-status-tag]
</RETS> CRLF
```

metadata-segment ::= <A metadata segment as defined in Section 11 .>

The body of the metadata response has the following format when replying to a format request of "STANDARD-XML" data:

```
<?xml version="1.0" ?>
[doctype]
<RETS 1*SP ReplyCode=quoted-reply-code 1*SP ReplyText=quoted-string SP >
[XML-metadata-segment]
[rets-status-tag ]
</RETS> CRLF
```

doctype ::= **`<!DOCTYPE RETS PUBLIC "-//RETS//DTD Metadata Content 1.8.1//EN">`**

XML-metadata-segment ::= A metadata segment as defined by the RETS Metadata XML DTD.

NOTE:

> RETS 1.8.1 requires all server responses to be well-formed XML, and additionally requires GetMetadata responses to be valid XML. In addition, RETS requires that clients parse server responses as XML, not as simple text streams. The response formats shown here are normative with respect to content, but not normative with respect to form. That is, servers are free to produce response XML in any format that complies with the W3C XML 1.0 recommendation, so long as it is valid with respect to the appropriate DTD. XML escaping of content is implied, as is XML processing of whitespace and line endings. See the W3C *XML Recommendation 1.0, Third Edition,* for full information on XML.

12.8 Reply Codes

Table 12-1 GetMetadata Reply Codes

Reply Code	Meaning
20500	Invalid Resource The request could not be understood due to an unknown resource.
20501	Invalid Type The request could not be understood due to an unknown metadata type.
20502	Invalid Identifier The identifier is not known inside the specified resource.
20503	No Metadata Found No matching metadata of the type requested was found.
20506	Unsupported Mimetype The server cannot return the metadata in any of the requested MIME types.
20507	Unauthorized Retrieval The metadata could not be retrieved because it requests metadata to which the supplied login does not grant access (e.g. Update Type data).
20508	Resource Unavailable The requested resource is currently unavailable.
20509	Metadata Unavailable The requested metadata is currently unavailable.
20510	Request Too Large Metadata could not be retrieved because a system limit was exceeded.
20511	Timeout The request timed out while executing.

20512	Too many outstanding requests The user has too many outstanding requests and new requests will not be accepted at this time.
20513	Miscellaneous error The server encountered an internal error.
20514	Requested DTD version unavailable. The client has requested the metadata in STANDARD-XML format using a DTD version that the server cannot provide.

Section 13 - PostObject Transaction

The PostObject transaction is used to upload structured information related to known system entities. This transaction allows the client to send one file as a MIME type along with more information. The server's response is similar to that of the Update transaction.

If the server supports the PostObject transaction, it sets the PostSupport field in the Object metadata to 1.

Before a client tries to upload an object, it SHOULD check for the presence of the ObjectData class in the metadata. If such a class exists, the client SHOULD check all its validation expressions, supplying the characteristics of the uploaded file as values of the known fields of the ObjectData table. See Table 5-1 ObjectData Content.

The PostObject request MUST be sent using POST method. The request arguments for the transaction are sent using HTTP headers.

> **Note: An Approved RCP is Related to this Section**
> Section 13 was added by the following approved RCP(s):
>
> RETS 1.8.0
>
> - RCP 63 Object Data and Upload
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

13.1 Required Request Header Fields

In addition to the Required client request header fields specified in section 3.4, the header of any single-file message MUST contain the following fields:

Field	Content Type	Description
X-UpdateAction	::= *1*24ALPHANUM*	
	Add	Add a new Object
	Replace	Change an existing Object
	Delete	Delete an existing Object
Content-type	<mime-type as defined in Section 5.1>	The mime-type of an Object
Content-length	::= **DIGIT*	The length of the posted Object in bytes.
Type	<Object type as defined in Table 11-20 Well-known Object Types>	
X-Resource	::= *RETSID*	The ResourceID for the Resource that this Object belongs as defined in Table 11-6 Metadata: Resource Description Fields

In addition to the header fields above, additional headers will be added based on the `UpdateAction` that is provided. These headers are defined below and the rules for including these headers is described following the definition.

Field	Content Type	Description
X-ResourceID	<value from the KeyField>	A value from the KeyField of the Resource for which the Object is to be uploaded.
X-ObjectID	::= *1*5DIGIT*	
X-OrderHint	::= *1*5DIGIT*	See Section 13.2 for further information.
X-UID	::= *TOKEN*	A string identifying the single Object being posted

The *X-UID* is the unique identifier of an existing Object, as reported by the server in a previous GetObject, PostObject, or in the related ObjectData class.

If the server does not support *`X-UID`* for the requested type of Objects and the client submits an *`X-UID`* instead of the *`ID`*, the server MUST respond with an error. The preferred error code is 20403: No Object Found. Servers that do not implement the PostObject Transaction MAY respond with a 20402: Invalid Identifier. If the requested type of Object has an ObjectData class linked in the metadata, the server MUST support this argument.

For Objects, the *`X-ResourceID`* is a value (e.g., MLS number, AgentID) from the `KeyField` of the Resource for which the Object is to be retrieved.

The *`X-ObjectID`* is the particular Object to be posted. Objects are assumed to be stored sequentially on the host beginning with an *`X-ObjectID`* of "1". This parameter can be used to specify the photo number, e.g. a value of "3" would indicate photo number 3.

Depending on the UpdateAction value, the *`X-ResourceID`*, *`X-ObjectID`*, *`X-UID`* or *`X-OrderHint`* should be omitted from the request based on the rules described below.

If any of *`X-ResourceID`*, *`X-ObjectID`* or *`X-OrderHint`* are used along with *`X-UID`*, the server MUST return the error 20804, Invalid or inconsistent request parameters.

If X-UpdateAction=***Add*** and the *`X-ResourceID`* and either *`X-ObjectID`* or optional header *`X-OrderHint`* number is used, the uploaded file will be added to the list of objects. The server will adjust *`X-ObjectID`* to ensure that the uploaded object assumes appropriate position in the list of existing objects. If *`X-ObjectID`* is used, the server MUST increase the *`X-ObjectID`* by one for existing objects that have an *`X-ObjectID`* that is of the same value or greater than the *`X-ObjectID`* of the inserted object. If the number of existing objects is less then the *`X-ObjectID`*, the uploaded object becomes the last one in the list. If *`X-OrderHint`* is used, the server recalculates the *`X-ObjectID`* of all objects so that they stay in sync with the order of the *`X-OrderHint`* values. If a client sends values for both *`X-ObjectID`* and *`X-OrderHint`*, the server MUST return error, preferably 20804 (Inconsistent parameters).

Required Header Field
`X-ResourceID`
`X-ObjectID` | *`X-OrderHint`*

If the X-UpdateAction=***Add*** and the *`X-UID`* is provided, the server may return an error.

If X-UpdateAction=***Add*** and none of *`X-ObjectID`*, *`X-OrderHint`* or *`X-UID`* is provided, the uploaded file becomes the last in the list of existing objects. *`X-ResourceID`* is required in this case. The server may set its *`X-OrderHint`* to any number higher that all existing *`X-OrderHint`* for this *`X-ResourceID`*.

Required Header Field
`X-ResourceID`

If X-UpdateAction=***Replace***, either the *`X-ResourceID`* and *`X-ObjectID`*, or the *`X-UID`* MUST be used. The uploaded file replaces the original Object. Any ObjectData fields not sent with the PostObject request (and not affected by the uploaded file) keep their previous values. If a client sends values for both *`X-ObjectID`* and *`X-UID`*, the server must return error, preferably 20804 (Inconsistent parameters).

Required Header Field
`X-ResourceID`
`X-ObjectID`

OR

Required Header Field
`X-UID`

If X-UpdateAction=***Delete***, either the *`X-ResourceID`* and *`X-ObjectID`*, or the *`X-UID`* MUST be used, and they MUST identify an existing Object. The body of the request SHOULD be empty and MUST be ignored by the server. It is expected that the server will re-adjust the *`X-ObjectID`* values such that there is no gap introduced by a Delete. If a client sends values for both *`X-ObjectID`* and *`X-UID`*, the server must return error, preferably 20804 (Inconsistent parameters).

Required Header Field
`X-ResourceID`
`X-ObjectID`

OR

Required Header Field
`X-UID`

The Delete request may also be sent with *`X-ResourceID`* without *`X-ObjectID`*, in which case all objects for the given *`X-ResourceID`* are to be deleted.

Required Header Field
`X-ResourceID`

Note: RETS 1.9.0: An approved RCP is related to this section
Section 13.1 is related to the following approved RCP(s):

- RCP 108 HTTP User Space Headers

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

13.2 Optional Request Header Fields

The client MAY specify additional headers. If a header name is identical to the System Name of a field in a corresponding ObjectData class (see 5.12), the server MAY use the header value to update that field. It is the server's decision whether such a field will be set to the client provided value, or calculated based on the uploaded file or other data. However, the server SHOULD calculate the values (rather than using the client-provided values) whenever it’s able to do so. Specifically, the field with standard name FileSize SHOULD always reflect the length of the file as it is stored with the server.

The client SHOULD check the ObjectData class to see what headers may be needed for the server.

13.2.1 OrderHint

`X-OrderHint`	*::= 1*5DIGIT*

`X-OrderHint` is a number suggesting where in the sequence of all objects belonging to the same *`X-ResourceID`* an uploaded object should be placed. Unlike *`X-ObjectID`* numbers, which must be an uninterrupted sequence of integral numbers starting with 1, the *`X-OrderHint`* may be any number. After an update, the server must modify *`X-ObjectID`* values for a given *`X-ResourceID`*, to ensure that the *`ObjectID`* order is the same as the *`X-OrderHint`* order. *`X-ObjectID`* numbers MUST follow the ordering of *`X-OrderHint`* numbers in the sense that if the *`X-OrderHint`* for object A is lower than the *`X-OrderHint`* for object B, then the *`X-ObjectID`* for object A MUST be lower that *`X-ObjectID`* for object B. In the case where multiple objects are set to share the same *`X-OrderHint`* value, the resulting *`X-ObjectID`* ordering is non-predictive.

The *`X-OrderHint`* MUST NOT be used in the PostObject request if it is not exposed in the ObjectData class linked to this object metadata (see 5.12 and Table 5-1). If it is used, the server should ignore the parameter.

13.2.2 WarningResponse

See Section 21.1 WarningResponse for details.

Note: RETS 1.9.0: An approved RCP is related to this section
Section 13.2 is related to the following approved RCP(s):

- RCP 108 HTTP User Space Headers

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

13.3 Request Body

The body of the request is the file being uploaded. If X-UpdateAction=Delete, the body MAY be empty, and SHOULD be ignored by the server.

13.4 PostObject Response Body Format

The response from the server is similar to that of the Update transaction (10.5):

```
<RETS 1*SP ReplyCode= quoted-reply-code 1*SP ReplyText= quoted-string *SP> CRLF
[ delimiter-tag ]
column-tag
compact-data
[activation-tag]
[error-block]
[warning-block]
[<RETS-STATUS 1*SP ReplyCode= quoted-end-reply-code 1*SP ReplyText= quoted-string *SP/>
</RETS> CRLF
```

In the compact-data, the server MUST send the values of Resource, Type, ResourceID, ObjectID and UID, if they were used in the request. The UID, if it exists in the related ObjectData class, MUST be sent even if it was not requested. The UID MAY also be sent if no ObjectData class is linked to this Object metadata, but the server is able to honor GetObject requests with UID.

Unless the UpdateAction requested was Delete, all other fields from the ObjectData table that were requested or changed MUST also be sent. Other fields from the ObjectData table MAY be sent as well.

activation-tag	::= TIMESTAMP [; TEXT]

If the object is not immediately accessible, the server MUST send a datetime when it is supposed to be activated. An explanation why the object is delayed MAY be appended.

error-block and *warning-block* are described in section 17.1 and 17.2.

13.5 Reply Codes

Table 13.1 Standard Reply Codes

Reply Code	Meaning
0	Upload successful.
20800	Unknown resource
20801	Invalid object type
20802	Invalid identifier
20803	Invalid update action
20804	Invalid or inconsistent request parameters
20805	No object found (for Delete)
20806	Unsupported MIME type
20807	Unauthorized
20808	Some objects not deleted (in case of Delete without ObjectID or UID, if some objects could not be deleted, while some were)
20809	Refused: object does not meet business rules
20810	FileSize too large Note that some servers MAY respond with HTTP status “413 – Request entity too large” if the uploaded file is larger than any acceptable limit.
20811	Timeout
20812	Too many outstanding requests
20813	Miscellaneous error

Section 14 - **DEPRECATED** GetPayloadList Transaction

This document will use the term payload to describe output formats that use either RESO defined XML schema or RESO defined DTD documents to provide the structure description for a well-formed XML document containing matching records from the server.

The GetPayloadList transaction is used to retrieve a list of available payloads for Search transaction output formats supported by the server.

The only available response format is COMPACT.

> **Note: An Approved RCP is Related to this Section**
> Section 14 was added by the following approved RCP(s):
>
> RETS 1.8.0
>
> - RCP 76 GetPayloadList
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

> **Note: An Approved RCP is Related to this Section**
> Section 14 was added by the following approved RCP(s):
>
> RETS 1.9.0
>
> - RCP 110 - Deprecate and Replace GetPayloadList
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

14.1 Required Request Arguments

There are no required request arguments.

14.2 **DEPRECATED** Optional Request Arguments

14.2.1 ID

The ID argument controls the payload list response, indicating that only payloads for the specified ID hierarchy should be returned.

ID	::= *metadata-id*[: metadata-id]
metadata-id	::= *1*ALPHANUM* | *

Metadata is organized hierarchically. Each level specifies in its first field an identifier for the metadata contained within that level. The GetPayloadList transaction only provides information for the Resource level of metadata. That is, a request may ask for all payloads for all resources or all payloads for a specific Resource level. (e.g. for the Resource level: ResourceID-Agent, Property, etc.). This identifier can be used to restrict requests to the payload list contained within specific instances of higher levels. If the last metadata-id is "*", then the request is for all Type metadata contained within that level and all metadata.

The ID values are those returned in the GetMetadata transaction reflects the metadata hierarchy as shown in Figure 11.1. For any metadata element, the ID argument is a list of the names of the parent elements for the desired element, separated by colons. For example, to retrieve the payload list for a given named Resource, the argument is simply the ResourceID.

Example: `ID=Property`

To retrieve the payload list for a specific class within a resource:

Example: `ID=Property:RES`

Servers MUST treat a request without an ID argument as requesting all available payloads.

14.3 Required Response Arguments

There are no required response arguments.

14.4 Optional Response Arguments

There are no optional response arguments.

14.5 **DEPRECATED** Payload Response Body Format

The body of the GetPayloadList response has the following format:

```
<RETS 1*SP ReplyCode=quoted-reply-code 1*SP
ReplyText=quoted-string SP > CRLF
<RETSPayloadList 1*SP Resource=quoted-string 1*SP
Class=quoted-string 1*SP
Version=quoted-string 1*SP
Date=quoted-string SP > CRLF
column-tag
[*compact-data]
[rets-status-tag]
</RETSPayloadList> CRLF
</RETS> CRLF
```

compact-data	::= **<DATA> *(** ***field-data*** **) </DATA>** ***CRLF***

A "<DATA>" tag, followed by a delimited list of field-data and a "</DATA>" end tag are returned to the client for each record returned. The field-delimiter is determined by the delimiter-tag.

column-tag	::= **<COLUMNS>** PayloadNameResourceClassDescriptionURIMetadataEntryIDVersion **</COLUMNS>** ***CRLF***

A fixed format row that indicates the information represented in each delimited column of the *compact-data* response rows.

Column Name	Content Type	Description
PayloadName	RETSNAME	Name of the payload
Resource	RETSID	The ResourceID of a resource that supports the payload
Class	RETSNAME	The ClassName of a class that supports the payload
Description	1*256PLAINTEXT	Description of the payload
URI	URI	Valid location of DTD or XML Schema defining the payload format
MetadataEntryID	RETSID	A value that does not change as long as the semantic meaning of the Payload record does not change.
Version	dtd-version | 1*64PLAINTEXT	A string representing the version of the payload. If the payload name is "STANDARD-XML" the version MUST be in dtd-version format as described in section 7.5 Search Response Body Format.

NOTE:

> RETS 1.8.0 requires all server responses to be well-formed XML, and additionally requires GetPayloadList responses to be valid XML. In addition, RETS requires that clients parse server responses as XML, not as simple text streams. The response formats shown here are normative with respect to content, but not normative with respect to form. That is, servers are free to produce response XML in any format that complies with the W3C XML 1.0 recommendation, so long as it is valid with respect to the appropriate DTD. XML escaping of content is implied, as is XML processing of whitespace and line endings. See the W3C *XML Recommendation 1.0, Third Edition,* for full information on XML.

Note: An Approved RCP is Related to this Section

Section 14 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 100 Alternate Standard Names

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

14.6 **DEPRECATED** Reply Codes

Table 14-1 GetPayloadList Reply Codes

Reply Code	Meaning
0	Successful.
20500	Invalid Resource. The request could not be understood due to an unknown resource.
20503	No Metadata Found. No matching metadata of the type requested was found.
20508	Resource Unavailable. The requested resource is currently unavailable.
20511	Timeout. The request timed out while executing.
20513	Miscellaneous Error. The server encountered an internal error.

Section 15 - Compact Data Format

Clients may choose to access data from a server in a compact data format that does not use full XML representation. When a client requests information from a compliant server in "COMPACT" or "COMPACT-DECODED" format, it will typically need to interpret the result by using the metadata that the server makes available.

15.1 Overall format

Compact format records are sequences of fields separated by delimiter. A tab character (an octet with a value of 09) is the default delimiter unless another is specified as part of the transaction. The delimiter MUST be some character other than the comma "," character. This character is reserved for separating values in any field with an interpretation of LookupMulti where more than one value may be applied to that field. The sequence of fields MUST be described by a `<COLUMNS>` tag in the body of the message that carries the compressed records. No field described in the `<COLUMNS>` tag may be omitted from the `<DATA>`; if the value of a particular field for some record is undefined or is suppressed for authorization reasons, the value MUST be represented by two delimiters with no intervening space. No field omitted in the `COLUMNS` tag may be added in any `DATA` tag. The number of fields in the `<COLUMNS>` tag MUST match the number of fields in the `<DATA>` tags.

Each compact record~~s~~ is enclosed within a `<DATA>` start tag and a `</DATA>` end tag.

Fields with an interpretation of Lookup or LookupMulti contains the LookupType Value from Table 11-23 when the format is COMPACT and the LookupType LongValue from Table 11-23 when the format is COMPACT-DECODED.

15.2 Decoded Format

COMPACT-DECODED format requires sending field data in an expanded form. For example, if a field representing data for City is given the interpretation of Lookup in the Metadata, there will be a corresponding LookupType table that contains at least two values, Value and LongValue. It may also contain a ShortValue, but that is not relevant to the example. For this example, the Value is 101 and the LongValue is Anytown. In the COMPACT format, the returned data for this field is 101. This is referred to as the coded value. In the COMPACT-DECODED format case, the returned data for this field is Anytown. This is referred to as the decoded value. A server MUST perform the expansion from the Value to the LongValue for fields with an interpretation of Lookup or LookupMulti.

15.3 Multivalued Fields

If the field is multivalued, values MUST be separated by commas and an optional space between each value. The final value does not have the comma or space before the field delimiter.

15.4 Transmission standards

A client or server transmitting a compact record MUST encode the data according to Table 15-1.

Table 15-1 Compact Data Field Format Representation

Type	Encoding Format
Numeric	An optional negative sign, followed by zero or more digits, followed by an optional period, followed optionally by zero or more digits. The interpretation determines if an optional character may be included. A valid number MUST contain at least one digit if it includes a decimal point or sign. The value may contain leading zeros before the decimal point. The value may contain trailing zeros after the decimal point and fraction, if any. Data types Tiny, Small, Int and Long (Table 11-12) may be signed but may not have a decimal point or fraction. Values with the interpretation LookupBitmask must not be signed, nor may they have nonzero digits after the decimal point.
Character	The plain character sequence, except for LookupMulti, which contains multiple sequences of characters separated by commas. Values with the interpretation LookupBitstring must contain only the characters "0" and "1".
Date	A date in *full-date* format.
Time	A date in *RETSTIME* format.
Date-Time	A date in *RETSDATETIME* format.
MultiSelect	A string consisting of one or more substrings, comma-delimited, each of which corresponds to an entry in the field's associated MetadataLookup table.
Boolean	A single character, either 1 for true or 0 for false.

All fields described in the <COLUMNS> tag MUST be included in the <DATA> tag. If the value of a particular field is undefined or empty, the value MUST be represented by two delimiters with no intervening space. If the value is suppressed, for authorization reasons, the value MUST be represented by two delimiters with no intervening space unless the request included the RestrictedIndicator, in which case the value of the RestrictedIndicator MUST be used.

Note: An Approved RCP is Related to this Section
Section 15.4 is related to the following approved RCP(s):

RETS 1.7.2:

- RCP 71 Time Zone Data

RETS 1.8.0:

- RCP 78 Specification Errata Changes
- RCP 87 RETS 1.7.2 Errata Document

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Section 16 - Session Protocol

A RETS session follows a well-defined sequence of transactions defined in this document to become established and when terminating. In particular, the authorization sequence MUST be followed in order to begin using other transactions within the protocol.

While HTTP is a stateless protocol, that is the current request does not depend on the previous request, the addition of the session protocol allows critical information to be communicated to the client reliably.

HTTP does include additional methods that are not used within this document. HTTP status codes are used to provide HTTP level information including session time out, while RETS status codes provide information on the response at the RETS protocol level.

The protocol contains four phases: connection establishment, authorization, session and termination.

16.1 Connection Establishment

A client initiates communication with a server by beginning a TCP connection on any mutually agreed TCP port, with the default being 6103 for unencrypted connections, and port 12109 for TLS-encrypted connections (24). When the TCP connection has entered the Established state, the session proceeds to the start of the Authorization phase.

> **Note: RETS 1.9.0: An approved RCP is related to this section**
> Section 16.1 is related to the following approved RCP(s):
>
> - RCP 109 - Update TLS specification references
>
> Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

16.2 Authorization

Authorization begins when the client sends the server a Login transaction. The Login transaction contains the basic information that the server requires in order to start an authorization decision: the user ID and optionally, some information about the client software.

A server responds to the Login request by sending back a "401 Unauthorized" status code and a WWW-Authenticate header. This is part of an authentication challenge to the client. Part of the WWW-Authenticate header may contain a checksum (nonce) of a concatenation of the following:

1. The client-IP.
2. The server-supplied timestamp.
3. The server's private-key.

Server implementers should note that because of intervening proxy servers, the client IP address may change from connection to connection.

The client concatenates the nonce to the checksum of the Request-URI; then performs an MD5 digest using a concatenation of the username, realm and password as the secret. This result is then returned to the server as part of an Authorization header. The server MUST then compute the equivalent function using its own stored copy of the user's password. If the two match and the nonce is the same, the user is considered authenticated, and the login can proceed with the server informing the client of the available capabilities. The login has been accomplished without actually sending the password. A server MAY provide an anonymous login. A client wishing an anonymous login sends an empty Authentication field in its Login transaction, after which the authorization proceeds as before.

16.3 Session

Once the Authorization phase has been completed, both endpoints enter the Session phase. During the Session phase, clients may issue any combination of requests for which they are authorized. The first of these MUST be to issue a GET requests for the "Action" URL, if any, included in the Login response (Section 4.10). After this, clients may issue other transactions. Note that the transport level security requirements will determine the protocol (HTTP/HTTPS), the port (6103/12109) and the version of the protocol for a specific implementation. Vendors using transport level security SHOULD use the most current version of the transport standard and the supporting libraries to minimize security risks.

Clients MAY issue multiple transactions without waiting for responses. However, servers are not required to process these requests in parallel, nor are servers required to complete the requests in the order in which they were issued. If a client issues a request before receiving a response to some earlier request, the client MUST be prepared to receive the responses in any order. The only way for a client to guarantee sequential execution of requests on every server is to wait for a response to any outstanding request before issuing a new request.

> **Note: RETS 1.9.0: An approved RCP is related to this section**
> Section 16.3 is related to the following approved RCP(s):

- RCP 109 - Update TLS specification references

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

16.4 Termination

A client SHOULD initiate termination of the session by sending a Logoff transaction. If a server receives a Logoff transaction while other operations are pending, it SHOULD abort those pending operations. However, a server MUST NOT rely on receiving a Logoff transaction in order to terminate a session, due to the possibility of communications problems preventing the transmission of the Logoff transaction by the client.

Servers SHOULD provide a timeout mechanism, and if they do, MUST inform the client of the timeout interval during the Login transaction (Section 4.7).

Section 17 - Update Response Blocks

The Update Transaction and the PostObject Transaction may result in warnings or errors. Client applications may return an explanation for the warning from the end user.

The format of the `<ERRORDATA>` and `<WARNINGDATA>` tag content is identical to COMPACT format.

17.1 Error Block

error-block	::= **<ERRORBLOCK>** *CRLF* 1*(**<ERRORDATA>** *field-delimiter field field-delimiter error-num* *field-delimiter error-offset field-delimiter* *error-text field-delimiter* **</ERRORDATA>**) **</ERRORBLOCK>**

An Error Block is returned when there is a problem with one or more of the fields. The error block contains information about the fields that have errors. It contains the field name, an error number, some additional text about the error (*error-text*), and where in the field data the error occurred (*error-offset*).

error-num	::= 1* *16 DIGIT*

This is the host error number. This number along with the *error-text* MAY be displayed to the user when looking at the corresponding field in the client application.

error-offset	::= *1*5DIGIT*

This is the offset into the field data that was sent by the client application to the server. It indicates at what character in the field data the problem was encountered. This number is set to zero ("0") if the offset of the error is unknown.

error-text	::= ** 1024*ALPHANUM

This is the error text generated by the host to assist the user in determining the problem with the field data. This text is associated with the *error-num*.

The error return format follows the `COMPACT` data format in all particulars. This affects primarily the quoting of special characters and the selection of the delimiter that separates the field values. In effect, the error return is a `COMPACT` data block without the usual `COLUMNS` element.

Note: An Approved RCP is Related to this Section
Section 17.1 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 59 Revised Update Transaction

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

17.2 Warning Block

warning-block	::= **<WARNINGBLOCK>** 1*(**<WARNINGDATA>** *field-delimiter field field-delimiter warning-num* *field-delimiter warning-offset field-delimiter* *warning-text field-delimiter response-required* *field-delimiter* **</WARNINGDATA>**) **</WARNINGBLOCK>**

A Warning Block is returned when there is a problem with one or more of the fields that would not prevent the record from being saved in the database. It contains a field name, a warning number, some additional text about the warning (*`warning-text`*), where in the field data the warning occurred (*`warning-offset`*) and an indicator whether an end-user response to this warning is requested or required. The delimiter is the same as the one defined for the *`error-block`*.

`field`	::= *RETSNAME*

The SystemName of the field to which the warning applies.

`warning-num`	::= *`1* 16`* *DIGIT*

The host warning number. This number, along with the *`warning-text`*, MAY be displayed to an end-user in association with the corresponding field in the client application.

`warning-text`	::= +**1024+TEXT*
`warning-offset`	::= *1*5DIGIT*

The offset into the field data that was sent by the client application to the server. It indicates at what character in the field data the problem was encountered. This number is set to zero if the offset of the error is unknown or if an offset is inapplicable.

`response-required`	::= **`0	1	2`**

The *`response-required`* value indicates whether an end-user response is requested or required:

Warning Number	Meaning	Explanation
`0`	No response is permitted.	The server is informing the user that a warning occurred. No user action is necessary.
`1`	A response is requested.	Business rules on the server request that the user explain why the warning occurred.
`2`	A response is mandatory.	Business rules require a reason. A response is required before the record is accepted.

If the *`response-required`* field indicates that a response is mandatory, the client MUST send the end-user response for the specific warning-num in the WarningResponse request argument in order for this record to be saved to the database.

Note: An Approved RCP is Related to this Section
Section 17.2 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 59 Revised Update Transaction

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

17.3 Warning Response

A previous Update or PostObject Transaction may have resulted in one or more warnings that must be resolved before the record can be stored in the server database. The WarningResponse permits the user to resolve those issues.

`WarningResponse`	::= *`warning-response *(field-delimiter warning-response)`*

`warning-response`	::= *`field`* **:** *warning-num* **=** *user-response*
`warning-num`	::= *`1*`* *`16 DIGIT`*

The *`warning-num`* value is the host warning number that was returned in the prior Update or PostObject Response body.

`user-response`	::= ** `1024 TEXT`* excluding delimiter

The *`user-response`* value is the text of the warning response in response to the specified warning. If a *`warning-num`* sent in the prior Update or PostObject Response body had a *`response-required`* value of 2, then the *`user-response`* value MUST NOT be null.

Note: An Approved RCP is Related to this Section
Section 17.3 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 59 Revised Update Transaction

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Section 18 - Authors

Paul Stusiak
Falcon Technologies Corporation
635 Ivy Ave.
Coquitlam, BC V3J 2H8
pstusiak@falcontechnologies.com

Sergio Del Rio
Templates for Business, Inc.
1155 West Pender Street, Suite 302
Vancouver, BC V6E 2P4
sergio.del.rio@t4bi.com

Libor Viktorin
CoreLogic Inc.
lviktorin@corelogic.com

Matthew McGuire
CoreLogic Inc.
MpMcGuire@corelogic.com

Archived - Previous to RETS 1.7.2

Leo Bijnagte
Vista Information Systems
100 Washington Square, Suite 1000
Minneapolis, MN 55401

Dan Musso
WyldFyre Technologies, Inc.
900 East Hamilton Ave.
Suite 500
Campbell, CA 95008

Bruce Toback
OPT, Inc.
11801 N. Tatum Blvd.
Suite 142
Phoenix, AZ 85028

Section 18 - 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®.

This document has benefited greatly from the comments of all those participating in the National Association of REALTORS®-Standards Work Group.

In addition to the authors, valuable discussion instrumental in creating this document has come from:

Richard Mendenhall
National Association of REALTORS®

Dale Stinton
National Association of REALTORS®

Mark Lesswing
National Association of REALTORS®

Larry Colson
Moore Data Management Services

Tom Curtis
Metro MLS

Kevin Knoepp
GTE Enterprise Solutions

Tom McLean
Resolution Software Consulting, Inc.

Tony Salvati
Grant Thornton

Errol Samuelson
RealSelect, Inc.

Allan Shapiro
Wantao Zhou
Interealty Corporation

Stuart Schuessler
Libor Viktorin
Mathew McGuire
Steve Clarke
MarketLinx Corporation

Michael DelGaudio
MRIS, Inc.

Maggie Diaz
Brita Brodin
Laure Chipman
WyldFyre, Inc.

Joshua Vosper
Rapattoni Corporation

Laila Sharshar
NewportWorks, Inc.

Eric Schlosser
Hewlett-Packard Company

Frank Tadman
MLSListings Inc.

Sergio Del Rio
Templates for Business Inc.

Jaison Freed
FBS Data Systems, Inc.

Ryan Bonham
Transparent Technologies Inc.

Gina Accawi
Falcon Technologies Corp.

Section 19 - References

Number	Reference
1	Braden, R., "Requirements for Internet Hosts — Communication Layers" STD 3, RFC 1123, IETF 1989.
2	R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, June 2014 J. Reschke, "Initial Hypertext Transfer Protocol (HTTP) Authentication Scheme Registrations", RFC 7236, June 2014 J. Reschke, "Initial Hypertext Transfer Protocol (HTTP) Method Registrations", RFC 7237, June 2014
3	Rivest, R., "The MD5 Message Authentication Algorithm", RFC 1321, April 1992
4	Crocker, D., "Standard for ARPA Internet Text Messages", RFC 2822, IETF 2001
5	US-ASCII. Coded Character Set - 7-Bit American Standard Code for Information Interchange. Standard ANSI X3.4-1986, ANSI, 1986.
6	Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E., and L. Stewart, "An Extension to HTTP: Digest Access Authentication", RFC 2617, January 1997.
7	International Organization for Standards, "Data Elements and Interchange Formats - Information Interchange - Representation of Dates and Times", ISO 8601, June 1988.
8	Borenstein, N., Freed, F., "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
9	American National Standard for Data Encryption Algorithm (DEA). Standard ANSI X3.92, ANSI, 1981.
10	Data Encryption Standard, FIPS46-2, December 30, 1993.
11	DES Modes of Operation, FIPS81, December 2, 1980
12	IEEE/ANSI Std. 1003.2-1992, Information Technology – Portable Operating System Interface (POSIX®) Part 2
13	Berners-Lee et al., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, IETF 1998
14	Kaliski, "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, IETF 1998
15	A. Barth, "HTTP State Management Mechanism", RFC 6265, April 2011
16	W3C, "HTML 4.01 Specification", W3C Recommendation 24 December 1999 (http://www.w3.org/TR/html401/)
17	W3C, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C Recommendation 4 February 2004 (http://www.w3.org/TR/2004/REC-xml-20040204/)
18	Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000
19	International Standards Organization, "ISO 8601:2004(E) Date elements and interchange formats - Information interchange - Representations of dates and times"
20	W3C, "Date and Time Formats", W3C Note 15 September 1997 [online] (http://www.w3.org/TR/NOTE-datetime)
21	Klyne, G. and Newman, C., "Date and Time on the Internet: Timestamps", RFC 3339, IETF 2002
22	Crocker, D. and Overell, P., "Augmented BNF for Syntax Specification: ABNF", RFC 2234, IETF 1997
23	M. Belshe, "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, May 2015
24	Dierks, T. "The Transport Layer Security (TLS) Protocol." *The Internet Engineering Task Force*. Internet Society. RFC 5246, IETF 2008 **NOTE**: This RFC may be obsoleted by the IETF. Implementations where this standard applies (HTTP-over-TLS) MUST use the most recent Standards Track version of the IETF standard.

Section 20 - Appendices

Appendix A - XML Schema References

Table A-1 DTD References

Real Estate Transaction Standard Data Content DTD	
Description	The document returned by a search specifying STANDARD-XML format. This DTD describes the document only, not the entire response. It may be used when transmitting listing or membership data through a channel other than a RETS server (for example, FTP).
Public Identifier	`-//RETS//DTD RETS Data Content 1.8.1//EN`
System Identifier	`http://www.rets.org/dtd/2008/08/REData-20080829.dtd`
Real Estate Transaction Standard STANDARD-XML Search Response DTD	

Description	The response returned by a search specifying STANDARD-XML format. This DTD simply encapsulates the REData DTD (above) in a standard RETS response element.
Public Identifier	`-//RETS//DTD RETS XML Search Response 1.8.1//EN`
System Identifier	`http://www.rets.org/dtd/2008/08/RETS-20080829.dtd`
Real Estate Transaction Standard COMPACT Search Response DTD	
Description	The response returned by a search specifying COMPACT or COMPACT-DECODED format.
Public Identifier	`-//RETS//DTD RETS COMPACT Search Response 1.8.1//EN`
System Identifier	`http://www.rets.org/dtd/2008/08/rets-compact-search-1_7_2.dtd`
RETS Metadata Content DTD	
Description	This DTD describes the STANDARD-XML metadata format. It may be used when transmitting metadata through a channel other than a RETS server.
Public Identifier	`-//RETS//DTD Metadata Content 1.8.1//EN`
System Identifier	`http://www.rets.org/dtd/2008/08/rets-metadata-content-1_7_2.dtd`
RETS Metadata STANDARD-XML GetMetadata Response DTD	
Description	The document returned by a GetMetadata transaction specifying a format of STANDARD-XML. This encapsulates the RETS Metadata Content DTD in a standard RETS response element.
Public Identifier	`-//RETS//DTD Metadata 1.8.1//EN`
System Identifier	`http://www.rets.org/dtd/2008/08/rets-metadata-1_7_2.dtd`
RETS Metadata COMPACT GetMetadata Response DTD	
Description	The document returned by a GetMetadata transaction specifying a format of COMPACT.
Public Identifier	`-//RETS//DTD Compact Metadata 1.8.1//EN`
System Identifier	`http://www.rets.org/dtd/2008/08/rets-compact-metadata-1_7_2.dtd`
RETS Login Response DTD	
Description	The document returned by a Login transaction.
Public Identifier	`-//RETS//DTD Login Response 1.7.2//EN`
System Identifier	`http://www.rets.org/dtd/2008/08/rets-login-1_7_2.dtd`
RETS Update Response DTD	
Description	The document returned by an Update transaction.
Public Identifier	`-//RETS//DTD Update 1.8.1//EN`
System Identifier	`http://www.rets.org/dtd/2008/08/rets-update-1_7_2.dtd`

`Note:` Certain System Identifier values have been split across multiple lines to prevent hypenation characters being added to the document that are not part of the identifier. Each System Identifier is a well-formed URI.

Appendix B - Sample Compact Metadata Response

This appendix contains examples for COMPACT metadata responses. It is NON-NORMATIVE: these examples illustrate one way of formatting COMPACT metadata, and one set of values. Section 11 describes the content and formatting rules in detail.

B.1 System

```
<METADATA-SYSTEM Version="1.00.000" Date="2002-03-20T12:03:38Z">
<SYSTEM SystemID= "NTREIS" SystemDescription= "North Texas Real Estate Information System" />
<COMMENTS>
This is a comment line
</COMMENTS>
</METADATA-SYSTEM>
```

B.2 Resource

```
<METADATA-RESOURCE Version="1.00.000"
Date="2002-03-20T12:03:38Z" >
<COLUMNS>ResourceIDStandardNameVisibleNameDescription
ClassCountKeyFieldClassVersionClassDateObjectVersion
ObjectDateSearchHelpVersionSearchHelpDateEditMaskVersion
EditMaskDate LookupVersionLookupDateUpdateHelpVersion
UpdateHelpDate ValidationExpressionVersion
ValidationExpressionDateValidationLookupVersion
ValidationLookupDateValidationExternalVersion
ValidationExternalDate</COLUMNS>
<DATA>AgentAgent AgentAgent Table1 Agentid1.00.000
2002-03-20T12:03:38Z</DATA>
<DATA>PropertyPropertyPropertyProperty Tables5
LN1.00.000 2002-03-20T12:03:38Z1.00.000
2002-03-20T12:03:38Z1.00.000
2002-03-20T12:03:38Z 1.00.000
2002-03-20T12:03:38Z 1.00.000
2002-03-20T12:03:38Z 1.00.000
2002-03-20T12:03:38Z 1.00.000
2002-03-20T12:03:38Z 1.00.000
2002-03-20T12:03:38Z 1.00.000
2002-03-20T12:03:38Z </DATA>
<DATA>TaxTaxTaxMultimedia objects20 PID1.00.000
2002-03-20T12:03:38Z</DATA>
</METADATA-RESOURCE>
```

B.3 Foreign Keys

```
<METADATA-FOREIGN_KEYS Version="1.00.000000"
Date="Wed, 23 Jan 2002 12:37:38 GMT">
<COLUMNS>PARENT_RESOURCE_IDPARENT_CLASS_IDPARENT_SYSTEMNAME
CHILD_RESOURCE_IDCHILD_CLASS_IDCHILD_SYSTEMNAME</COLUMNS>
<DATA>PropertyRESMLSNUMTAXTAXMLSNUM</DATA>
<DATA>PropertyRESMLSNUMHistoryHistoryMLSNUM</DATA>
<DATA>PropertyRESMLSNUMOpenHouseOpenHouseMLSNUM</DATA>
<DATA>PropertyRESListingAgentIDAgentAgentAgentID</DATA>
<DATA>PropertyRESCOListingAgentIDAgentAgentAgentID</DATA>
<DATA>PropertyRESSellingAgentIDAgentAgentAgentID</DATA>
<DATA>PropertyRESCOSellingAgentIDvAgentAgentAgentID</DATA>
<DATA>PropertyRESListingOfficeIDOfficeOfficeOfficeID</DATA>
<DATA>PropertyRESSellingOfficeIDOfficeOfficeOfficeID</DATA>
</METADATA-FOREIGNKEYS>
```

Note: An Approved RCP is Related to this Section
Section B.3 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 87 RETS 1.7.2 Errata Document

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

B.4 Class

GetMetadata request:

```
Type: METADATA-CLASS

ID: 0
```

Compact reply:

```
<METADATA-CLASS Resource="Agent" Version="1.00.000"
Date="2002-03-20T12:03:38Z" />
<COLUMNS>ClassNameVisibleNameStandardNameDescription
TableVersionTableDateUpdateVersion UpdateDate OffsetSupport </COLUMNS>
<DATA>AgentAgentAgentAll Agents1.00.000
2002-03-20T12:03:38Z</DATA>
</METADATA-CLASS>
```

B.5 Table

GetMetadata request:

```
Type: METADATA-TABLE

ID: Property: RES
```

Compact reply:

```
<METADATA-TABLE Resource="Property" Class="RES" Version="1.00.000"
Date= "2002-03-20T12:03:38Z" >
<COLUMNS>SystemNameStandardNameLongNameDBNameShortName
MaximumlengthDataTypePrecisionSearchableInterpretation
AlignmentUseSeparatorEditMaskIDLookupNameMaxSelectUnits
IndexMinimumMaximumDefaultRequiredSearchHelpID
MetadataEntryIDModTimeStampForeignKeyForeignFieldKeyQuery
KeySelect</COLUMNS>
<DATA>LNListIDListing IDLNListID8Int01
NumberLeft011</DATA>
<DATA>PTYPPropTypeProperty TypePTProp Type
2Int01NumberLeft0</DATA>
<DATA>LPListPriceList PriceLPLst Pr8Int01
CurrencyRight1142</DATA>
<DATA>OWNOwnerOwner NameOWNOwn Name20Character
00Left0</DATA>
<DATA>VEWViewViewVEWView10Long01LookupBitmaskLeft
0VEW1</DATA>
<DATA>EFExtFeatFeaturesEFExt Feat10Character01
LookupMultiLeft0EFT2</DATA>
<DATA>SDSchDistSchool DistrictSDSchDist10Character
01LookupLeft0SD</DATA>
<DATA>ARMLSAreaMLS AreaARArea4Int01LookupLeft
0AR3031</DATA>
</METADATA-TABLE>
```

B.6 Update

GetMetadata request:

```
Type:METADATA_UPDATE

ID: Property: RES
```

Compact reply:

```
<METADATA-UPDATE Resource="Property" Class="RES" Version="1.00.000"
Date= "2002-03-20T12:03:38Z" >
<COLUMNS> UpdateNameDescriptionKeyFieldVersionDate
MetadataEntryID</COLUMNS>
<DATA>AddAdd a new Residential Listing1.00.000
2002-03-20T12:03:38Z</DATA>
<DATA>ChangeChange a Residential ListingListNumber1.00.000
2002-03-20T12:03:38Z</DATA>
<DATA>BOMPut a Residential Listing Back on Market ListNumber
1.00.0002002-03-20T12:03:38Z</DATA>
</METADATA-UPDATE>
```

B.7 Update Type

GetMetadata request:

```
Type: METADATA-UPDATE_TYPE

ID: Property: RES: Add
```

Compact reply:

```
<METADATA-UPDATE_TYPE Resource="Property" Class="RES" Update="Add"
Version="1.00.000" Date="2002-03-20T12:03:38Z" >
<COLUMNS>SystemNameSequenceAttributesDefault
ValidationExpressionIDUpdateHelpIDValidationLookupName
ValidationExternalNameMetadataEntryIDMaxUpdate</COLUMNS>
<DATA>STNUM12StNumHelp</DATA>
<DATA>STNAME22StreetName</DATA>
<DATA>LD32ListDateDateHelp</DATA>
<DATA>LISTOFF42,3</DATA>
</METADATA-UPDATE_TYPE>
```

B.8 Object

GetMetadata request:

```
Class:METADATA-OBJECT

ID:0
```

Compact reply:

```
<METADATA-OBJECT Resource="Property" Version="1.00.000"
Date="2002-03-20T12:03:38Z" >
<COLUMNS>ObjectTypeMIMETypeVisibleNameDescription
MetadataEntryIDObjectTimeStampObjectCountLocation<</COLUMNS>
<DATA>Photoimage/jpegFull PhotosHigh Resolution Property Photos
12015-05-22T12:03:38Z124</DATA>
<DATA>Thumbnailimage/jpegSmall PhotosLow Resolution Property Photos
12015-05-24T15:03:38Z12</DATA>
</METADATA-OBJECT>
```

Note: An Approved RCP is Related to this Section
Section B.8 is related to the following approved RCP(s):

RETS 1.8.0

- RCP 87 RETS 1.7.2 Errata Document

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

B.9 Lookup

GetMetadata request:

```
Type: METADATA-LOOKUP

ID: 0
```

Compact reply:

```
<METADATA-LOOKUP Resource="Property" Version="1.00.000"
Date="2002-03-20T12:03:38Z" >
<COLUMNS>LookupNameVisibleNameVersionDateMetadataEntryID</COLUMNS>
<DATA>1Status1.00.0002002-03-20T12:03:38Z</DATA>
<DATA>2Phone Type1.00.0002002-03-20T12:03:38Z</DATA>
</METADATA-LOOKUP>
<METADATA-LOOKUP Resource="Agent" Version="1.00.000"
Date="2002-03-20T12:03:38Z">
<COLUMNS>LookupNameVisibleNameVersionDateMetadataEntryID</COLUMNS>
<DATA>1Status1.00.0002002-03-20T12:03:38Z</DATA>
</METADATA-LOOKUP>
```

B.10 Lookup Type

GetMetadata request:

```
Type: METADATA-LOOKUP_TYPE

ID: *
```

Compact reply:

```
<METADATA-LOOKUP_TYPE Resource="Property" Lookup="AR" Version="1.00.000"
Date="2002-03-20T12:03:38Z">
><COLUMNS>LongValueShortValueValueMetadataEntryID</COLUMNS>
<DATA>Capitol HillCap Hill1</DATA>
<DATA>Juanita HillJuanita2</DATA>
<DATA>Maple ValleyMpl Valley3</DATA>
<DATA>Downtown RedmondDntn Rdmd<4></DATA>
</METADATA-LOOKUP_TYPE>
<METADATA-LOOKUP_TYPE Resource="Agent" Lookup="STAT" Version="1.00.000"
Date= "2002-03-20T12:03:38Z">
<COLUMNS>LongValueShortValueValueMetadataEntryID</COLUMNS>
<DATA>Active ACT1</DATA>
<DATA>SuspendedSUS2</DATA>
<DATA>InactvieINA3</DATA>
</METADATA-LOOKUP_TYPE>
```

B.11 Search Help

GetMetadata request:

```
Type: METADATA-SEARCH_HELP

ID: Property
```

Compact reply:

```
<METADATA-SEARCH_HELP Resource="Property" Version="1.00.000"
Date="2002-03-20T12:03:38Z" >
<COLUMNS>SearchHelpIDValueMetadataEntryID</COLUMNS>
<DATA>1Enter the number in the following format dxd</DATA>
<DATA>2Enter the number in the following format d.dd</DATA>
</METADATA-SEARCH_HELP>
```

B.12 Edit Mask

GetMetadata request:

```
Type: METADATA-EDITMASK

ID: Property
```

Compact reply:

```
<METADATA-EDITMASK Resource="Property" Version="1.00.000"
Date= "2002-03-20T12:03:38Z">
<COLUMNS>EditMaskIDValueMetadataEntryID</COLUMNS>
<DATA>1[0-9]{1,2}[x][0-9]{1,2} </DATA>
<DATA>2[0-9]{3}[0-9]{2}[0-9]{4} </DATA>
</METADATA-EDITMASK>
```

B.13 Update Help

GetMetadata request:

```
Type: UPDATE_HELP

ID: Property
```

Compact reply:

```
<METADATA-UPDATE_HELP Resource="Property" Version="1.00.000"
Date="2002-03-20T12:03:38Z" >
<COLUMNS>UpdateHelpIDValueMetadataEntryID</COLUMNS>
<DATA>1Enter the number in the following format dxd</DATA>
<DATA>2Enter the number in the following format d.dd</DATA>
</METADATA-UPDATE_HELP>
```

B.14 Validation Expression

GetMetadata request:

```
Type: METADATA-VALIDATION_EXPRESSION

ID: Property
```

Compact reply:

```
<METADATA-VALIDATION_EXPRESSION Resource="Property" Version="1.00.000"
Date= "2002-03-20T12:03:38Z" >
<COLUMNS>ValidationExpressionIDValidationExpressionTypeValue
MetadataEntryID</COLUMNS>
<DATA>Office1ACCEPT>
LAG=.AGENTCODE. .OR. (LO=.BROKERCODE. .AND. .ENTRY.=OFFICE)</DATA>
<DATA>Agent1ACCEPT(LAG=.AGENTCODE.) .OR. (SAG=.AGENTCODE.)</DATA>
<DATA>ListDateACCEPT LD>.TODAY. - 3 .AND. LD<.TODAY. + 3</DATA>
</METADATA-VALIDATION_EXPRESSION>
```

B.15 Validation External

GetMetadata request:

```
Type: METADATA-VALIDATION_EXTERNAL

ID: Property
```

Compact reply:

```
<METADATA-VALIDATION_EXTERNAL Resource="Property" Version="1.00.000"
Date= "2002-03-20T12:03:38Z" >
<COLUMNS>ValidationExternalNameSearchResourceSearchClassVersionDate
MetadataEntryID</COLUMNS>
<DATA>1Office Office1.00.0002002-03-20T12:03:38Z </DATA>
<DATA>2TaxHENN1.00.0002002-03-20T12:03:38Z </DATA>
</METADATA-VALIDATION_EXTERNAL>
```

B.16 Validation External Type

GetMetadata request:

```
Type: METADATA-VALIDATION_EXTERNAL_TYPE

ID: Property: VET1
```

Compact reply:

```
<METADATA-VALIDATION_EXTERNAL_TYPE Resource="Property"
ValidationExternalName="VET1" Version="1.00.000"
Date="2002-03-20T12:03:38Z" >
<COLUMNS>SearchFieldDisplayFieldResultsFieldsMetadataEntryID
</COLUMNS>
<DATA>AgentID, AgentCodeAgentName, OfficeNameSaleAgentID=AgentID,
SaleAgentName=AgentName, SaleOfficeID=OfficeID,
SaleOfficeName=OfficeName</DATA>
</METADATA-VALIDATION_EXTERNAL_TYPE>
```

B.17 Column Group Set

GetMetadata request:

```
Type: METADATA-COLUMN_GROUP_SET

ID Format: Resource : Class
```

ID Example: Property : RES

Compact reply:

```
<METADATA-COLUMN_GROUP_SET Version="1.00.000" Date= "Sat, 20 Mar 2002 12:03:38 GMT" Resource="Property" Class="RES" >
<COLUMNS>MetadataEntryIdColumnGroupSetNameColumnGroupSetParent
SequenceLongNameShortNameDescriptionColumnGroupNamePresentationStylePresentationColumnsURL</COLUMNS>
<DATA>10000123456Residential1Residential ListingResidentialThe top node of the Residential Listing Data Entry Hierarchy</DATA>
<DATA>10000123457WaterFrontResidential1WaterFront InformationWaterFrontDetails about water front for propertyWaterFrontEdit1</DATA>
<DATA>10000123457BedroomsResidential2Bedroom InformationBedroomsDetails about bedrooms for propertyBedroomsMatrix</DATA>
<DATA>10000123457AgentInfoResidential3Agent InformationAgentAgent Website
www.mywebsite.com/agent?Agent=.AGENTCODE.?Listing=.ListingID.</DATA>
</METADATA-COLUMN_GROUP_SET>
```

B.18 Column Group

GetMetadata request:

```
Type: METADATA-COLUMN_GROUP

ID Format: Resource : Class
```

ID Example: Property : RES

Compact reply:

```
<METADATA-COLUMN_GROUP Version="1.00.000" Date= "Sat, 20 Mar 2002 12:03:38 GMT"
Resource="Property" Class="RES" >
<COLUMNS>MetadataEntryIdColumnGroupNameControlSystemName
LongNameShortNameDescriptionVersionDate</COLUMNS>
<DATA>10001123456WaterFrontWaterFrontFlagWater Front InformationWater FrontListing data that contains water front information for the
Listing1.00.000Thu, 3 Feb 2005 20:35:15 GMT</DATA>
</METADATA-COLUMN_GROUP>
```

B.19 Column Group

GetMetadata request:

```
Type: METADATA-COLUMN_GROUP_CONTROL

ID Format: Resource : Class : ColumnGroup
```

ID Example: Property : RES : WaterFront

Compact reply:

```
<METADATA-COLUMN_GROUP_CONTROL Version="1.00.000" Date= "Sat, 20 Mar 2002 12:03:38 GMT" Resource="Property" Class="RES"
ColumnGroup="WaterFront" >
<COLUMNS>MetadataEntryIdLowValueHighValue</COLUMNS>
<DATA>1001112345611</DATA>
</METADATA-COLUMN_GROUP_CONTROL>
```

B.21 Column Group Table

GetMetadata request:

```
Type: METADATA-COLUMN_GROUP_TABLE
ID Format: Resource : Class : ColumnGroup
```

ID Example: Property : RES : WaterFront

Compact reply:

```
<METADATA-COLUMN_GROUP_TABLE Version="1.00.000" Date= "Sat, 20 Mar 2002 12:03:38 GMT" Resource="Property" Class="RES"
ColumnGroup="WaterFront" >
<COLUMNS>MetadataEntryIdSystemNameDisplayOrder</COLUMNS>
<DATA>10111123450WaterFront1</DATA>
<DATA>10111123451WaterAccess2</DATA>
<DATA>10111123452WaterFrontage3</DATA>
<DATA>10111123453WaterView4</DATA>
</METADATA-COLUMN_GROUP_TABLE>
```

B20 Column Group Normalization

GetMetadata request:

```
Type: METADATA-COLUMN_GROUP_NORMALIZATION
ID Format: Resource : Class : ColumnGroup
```

ID Example: Property : RES : WaterFront

Compact reply:

```
<METADATA-COLUMN_GROUP_NORMALIZATION Version="1.00.000" Date= "Sat, 20 Mar 2002 12:03:38 GMT" Resource="Property"
Class="RES" ColumnGroup="WaterFront" >
<COLUMNS>MetadataEntryIdTypeIdentifierSequenceColumnLabelSystemName</COLUMNS>
<DATA>10211123450Bedroom1LengthBedroom1Length</DATA>
<DATA>10211123451Bedroom1WidthBedroom1Width</DATA>
<DATA>10211123452Bedroom1AreaBedroom1Area</DATA>
<DATA>10211123453Bedroom2LengthBedroom2Length</DATA>
<DATA>10211123454Bedroom2WidthBedroom2Width</DATA>
<DATA>10211123455Bedroom2AreaBedroom2Area</DATA>
<DATA>10211123456Bedroom3LengthBedroom3Length</DATA>
<DATA>10211123457Bedroom3WidthBedroom3Width</DATA>
<DATA>10211123458Bedroom3AreaBedroom3Area</DATA>
<DATA>10211123459LivingRoomLengthLivingRoomLength</DATA>
<DATA>10211123460LivingRoomWidthLivingRoomWidth</DATA>
<DATA>10211123461LivingRoomAreaLivingRoomArea</DATA>
<DATA>10211123462DiningRoomLengthDiningRoomLength</DATA>
<DATA>10211123463DiningRoomWidthDiningRoomWidth</DATA>
<DATA>10211123464DiningRoomAreaDiningRoomArea</DATA>
<DATA>10211123465KitchenLengthKitchenLength</DATA>
<DATA>10211123466KitchenWidthKitchenWidth</DATA>
<DATA>10211123467KitchenAreaKitchenArea</DATA>
</METADATA-COLUMN_GROUP_NORMALIZATION>
```

Example Screen Display based on Above Compact Reply:

Type	Sequence	Length	Width	Area
Bedroom	**1**	**Bedroom1Length**	**Bedroom1Width**	**Bedroom1Area**
Bedroom	**2**	**Bedroom2Length**	**Bedroom2Width**	**Bedroom2Area**
Bedroom	**3**	**Bedroom3Length**	**Bedroom3Width**	**Bedroom3Area**
LivingRoom		**LivingRoomLength**	**LivingRoomWidth**	**LivingRoomArea**
DiningRoom		**DiningRoomLength**	**DiningRoomWidth**	**DiningRoomArea**
Kitchen		**KitchenLength**	**KitchenWidth**	**KitchenArea**

The user's data entry area would be the greyed out section and the data that they enter would be assigned to the SystemName in that grid position.

Note: An Approved RCP is Related to this Section
Section B.18 is related to the following approved RCP(s):

RETS 1.7.2

- RCP 71 Time Zone Data

RETS 1.8.0

- RCP 87 RETS 1.7.2 Errata Document

Content in this section has been updated or modified since the previous RETS version. Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

B.21 Payload

GetMetadata request:

```
Type: METADATA_PAYLOAD

ID: Property
```

Compact reply:

```
<METADATA-PAYLOAD Version="1.00.000"
Date="2017-03-16T12:03:38Z" >
<COLUMNS>RESOURCE_IDCLASS_IDPAYLOAD_IDPAYLOADNAMEVERSIONURIDESCRIPTION</COLUMNS>
<DATA>PropertyRESRESListing1Residential Listing Short1.5http://example.com/rs1.xsdOne line residential
listing</DATA>
<DATA>PropertyRESRESListing2Residential Listing Short-json1.5http://example.com/rs2.jsonJson one line
resi</DATA>
<DATA>PropertyLNDLand1LotsAndLand1.5http://example.com/lnd1.xsdLots and Land</DATA>
<DATA>AgentAgentListingAgent1Short Agent1.5http://example.com/agt1Public agent information for
syndication</DATA>

</METADATA-PAYLOAD>
```

Appendix C - Summary of RETS Reply Codes

Table C-1 Consolidated RETS Reply Codes

Reply Code	Meaning
0	Operation successful
10000	System error The server has detected an error with the request that prevents it from identifying the type of request, or that prevents the server from routing the request for processing. This return code MUST NOT be used when a more specific return code can be determined.
20003	Zero Balance The user has zero balance left in their account.
20004 thru 20011	RESERVED
20012	Broker Code Required The user belongs to multiple broker codes and one must be supplied as part of the login. The broker list is sent back to the client as part of the login response (see section 4.6).
20013	Broker Code Invalid The Broker Code sent by the client is not valid or not valid for the user
20014 thru 20019	RESERVED
20022	Additional login not permitted There is already a user logged in with this user name, and this server does not permit multiple logins.
20036	Miscellaneous server login error The quoted-string of the body-start-line contains text that SHOULD be displayed to the user
20037	Client authentication failed. The server requires the use of a client password (section 4.1.2), and the client either did not supply the correct client password or did not properly compute its challenge response value.
20041	User-agent authentication required. The server requires the use of user-agent authentication (section 4.1.2), and the client did not supply the user-agent header values.
20050	Server Temporarily Disabled The server is temporarily offline. The user should try again later
20140	Insecure password. The password does not meet the site's rules for password security.
20141	Same as Previous Password. The new password is the same as the old one.
20142	The encrypted user name was invalid.

20200	Unknown Query Field The query could not be understood due to an unknown field name.
20201	No Records Found No matching records were found.
20202	Invalid Select The Select statement contains field names that are not recognized by the server.
20203	Miscellaneous Search Error The quoted-string of the body-start-line contains text that MAY be displayed to the user.
20206	Invalid Query Syntax The query could not be understood due to a syntax error.
20207	Unauthorized Query The query could not be executed because it refers to a field to which the supplied login does not grant access.
20208	Maximum Records Exceeded Operation successful, but all of the records have not been returned. This reply code indicates that the maximum records allowed to be returned by the server have been exceeded. Note: reaching/exceeding the "Limit" value in the client request is not a cause for the server to generate this error.
20209	Timeout The request timed out while executing
20210	Too many outstanding queries The user has too many outstanding queries and new queries will not be accepted at this time.
20211	Query too complex The query is too complex to be processed. For example, the query contains too many nesting levels or too many values for a lookup field.
20212 [deprecated]	Invalid key request [deprecated] The transaction does not meet the server's requirements for the use of the `Key` option.
20213 [deprecated]	Invalid `Key` [deprecated] The transaction uses a key that is incorrect or is no longer valid. Servers are not required to detect all possible invalid key values.
20301	Invalid parameter. Additional information is provided in the error block.
20302	Unable to save record on server.
20303	Miscellaneous Update Error.
20311	`Warning Response` was not given for all warnings that contained a {{*response-required*}}value of 2.
20312	`Warning Response` was given for a warning that contained a *`response-required`* value of 0.
20400	Invalid Resource The request could not be understood due to an unknown resource.
20401	Invalid Type The request could not be understood due to an unknown object type for the resource.
20402	Invalid Identifier The identifier does not match the KeyField of any data in the resource.
20403	No Object Found No matching object was found to satisfy the request.
20406	Unsupported MIME type The server cannot return the object in any of the requested MIME types.
20407	Unauthorized Retrieval The object could not be retrieved because it requests an object to which the supplied login does not grant access.
20408	Resource Unavailable The requested resource is currently unavailable.

20409	Object Unavailable The requested object is currently unavailable.
20410	Request Too Large No further objects will be retrieved because a system limit was exceeded.
20411	Timeout The request timed out while executing
20412	Too many outstanding requests The user has too many outstanding requests and new requests will not be accepted at this time.
20413	Miscellaneous error The server encountered an internal error.
20414	URL Location Not Supported The server does not support retrieving Objects by URL.
20415	Objects in the response body, Location=0 Not Supported The server does not support retrieving Objects in the response body.
20500	Invalid Resource The request could not be understood due to an unknown resource.
20501	Invalid Type The request could not be understood due to an unknown metadata type.
20502	Invalid Identifier The identifier is not known inside the specified resource.
20503	No Metadata Found No matching metadata of the type requested was found.
20506	Unsupported MIMEType The server cannot return the metadata in any of the requested MIME types.
20507	Unauthorized Retrieval The metadata could not be retrieved because it requests metadata to which the supplied login does not grant access (e.g. Update Type data).
20508	Resource Unavailable The requested resource is currently unavailable.
20509	Metadata Unavailable The requested metadata is currently unavailable.
20510	Request Too Large Metadata could not be retrieved because a system limit was exceeded.
20511	Timeout The request timed out while executing.
20512	Too many outstanding requests The user has too many outstanding requests and new requests will not be accepted at this time.
20513	Miscellaneous error The server encountered an internal error.
20514	Requested DTD version unavailable. The client has requested the metadata in STANDARD-XML format using a DTD version that the server cannot provide.
20701	Not logged in The server did not detect an active login for the session in which the Logout transaction was submitted.
20702	Miscellaneous error. The transaction could not be completed. The ReplyText gives additional information.
20800	Unknown resource
20801	Invalid object type
20802	Invalid identifier

20803	Invalid update action
20804	Invalid (inconsistent) request parameters
20805	No object found (for Delete)
20806	Unsupported MIME type
20807	Unauthorized
20808	Some objects not deleted (in case of Delete without ObjectID or UID, if some objects could not be deleted, while some were)
20809	Refused: object does not meet business rules
20810	FileSize too large Note that some servers MAY respond with HTTP status “413 – Request entity too large” if the uploaded file is larger than any acceptable limit.
20811	Timeout
20812	Too many outstanding requests
20813	Miscellaneous error

Appendix D - Maximum Field Length and Display Information

This appendix defines formulas used to determine the maximum character length of field data based on the data type of the field and provides examples. These formulas describe how the RETS server should calculate the MaximumLength for reliable use by RETS client applications.

It is NON-NORMATIVE: these examples illustrate one case of calculating and formatting field values and their metadata and one set of values. Section 11.3.2 describes the rules in detail.

D.1 Datatype Boolean

Any field with the Boolean data type should represent the MaximumLength as '1'. The definition of the Boolean data type requires a single character representation of the data.

Interpretation	Precision	Separator	Units	Max Select	Extreme Example	Maximum Length	Display
null	n/a	n/a	n/a	n/a	0	1	`False`
null	n/a	n/a	n/a	n/a	1	1	`True`
Lookup	n/a	n/a	n/a	n/a	12345678	8	from lookup LookupName, longvalue, lookup shortvalue, the value from the corresponding lookup values, from value, 0 or 1

D.2 Datatype Character

Fields designated as the Character data type with an interpretation of Lookup or LookupMulti should calculate the MaximumLength according to the following:

(MaxSelect * (MaxValueLength + 3)) -1

Add the maximum character length of the longest Lookup Value in the Lookup metadata defined by the Lookup Name of the field to the enclosing quotations and delimiter characters ('3') then multiply this with the MaxSelect for the field. Finally, since there is no final delimiter in the list of delimiter separated values, subtract '1' from the total to determine the MaximumLength.

Example: The Field 'Appliance' has 10 Lookup items and has a MaxSelect of 4 lookup items. The longest Lookup Value length is 6 for the Lookup Value 'FRIDGE', used for the Lookup Long Value 'Refrigerator'

(4 * (6 + 3)) - 1 = 35.

Interpretation	Precision	Separator	Units	Max Select	Extreme Example	Maximum Length	Display
null	n/a	n/a	n/a	n/a	random_string	13	`random_string`
Lookup	n/a	n/a	n/a	n/a	random_string	13	`from lookup LookupName, True`

D.3 Datatype Decimal

Fields designated as Decimal only include numbers represented using a Decimal point. The Precision of the field determines the maximum number of decimal characters following the decimal point of the number. However, the maximum decimal precision of the data includes the decimal spaces before the decimal as well. For a decimal number, the MaximumLength should match the maximum decimal precision of the value plus one to represent the decimal point character itself.

An example: A signed 16 bit floating point number with a Precision of 3 has a maximum number of +32.787/-32.768 which is 7 characters, including sign. RETS does not define Decimal numbers by binary size, so the server should advertise the MaximumLength as appropriate to the data exposed via the RETS server interface.

The Currency interpretation follows the same rules as a Decimal number with 2 decimal points of Precision.

Interpretation	Precision	Separator	Units	Max Select	Extreme Example	Maximum Length	Display
Numeric	2	,	null	n/a	-12342.21	9	`-12,342.21`
Numeric	1	,	Feet	n/a	123.1	5	`123.1 feet`
Currency	2	,	n/a	n/a	1246.227	7	`$1246.22`

Note: An Approved RCP is Related to this Section
Appendix D is related to the following approved RCP(s):

RETS 1.8.0

- RCP 77 Maximum Field Length

Content in this section has been updated or modified since the previous RETS version.
Click the link above to review this RCP and the associated changes that were proposed and adopted in this version.

Appendix E - Approved RCPs

- Version 1.7.2
    - RETS Change Proposal 64 - Omnibus Adopted Schemas Revisions and Errata
    - RETS Change Proposal 66 - Deprecate Lookup Types LookupBitmask and LookupBitstring
    - RETS Change Proposal 71 - Time Zone Data
    - RETS Change Proposal 72 - LookupType String Length
- Version 1.8.0
    - RCP 59 - Revised Update Transaction
    - RCP 60 - Metadata Changes for Update
    - RCP 61 - Validation Expression Replacement
    - RCP 63 - Object Data and Upload
    - RCP 65 - Session information tokens
    - RCP 68 - Search Has Key Index Support
    - RCP 69 - LookupType Value
    - RCP 70 - Metadata Role Support
    - RCP 74 - Location Availability in Object Metadata
    - RCP 75 - Offset Availability in the Metadata
    - RCP 76 - GetPayloadList
    - RCP 77 - Maximum Field Length
    - RCP 78 - Specification Errata Changes
    - RCP 79 - Add Preferred Flag to GetObject Responses
    - RCP 80 - Optional Query
    - RCP 82 - LookupMulti Quoting Rule
    - RCP 87 - RETS 1.7.2 Errata Document
    - RCP 90 - Deprecate CommonInterest Class Well-Known Name
    - RCP 91 - StandardNames Version Information in Login Transaction
    - RCP 92 - RESO Payload Transport-Level Metadata Support
    - RCP 93 - Add Content-Sub-Description to GetObject
    - RCP 94 - Improved Error Handling in GetObject
    - RCP 98 - Additional Information Fields in METADATA-SYSTEM and Login
    - RCP 99 - Mixing StandardNames and SystemNames
    - RCP 100 - Alternate Standard Names
    - RCP 101 - Child Rows Support
    - RCP 102 - GetObject URL as Default Location
- Version 1.9.0
    - RCP 103 - Geospatial Search
    - RCP 104 - StandardValue for Enumerations
    - RCP 105 - Update Transaction Response Format Correction
    - RCP 106 - Client cookie support for RFC 6265
    - RCP 107 - IETF HTTP RFC Updates to references in the RETS specification
    - RCP 108 - Migrate RETS specific HTTP Headers to HTTP User Space headers (X-*)
    - RCP 109 - Update TLS specification references to current Internet/Industry Standards
    - RCP 110 - Deprecate and Replace GetPayloadList
    - RCP 112 - RETS Metadata Version Header
    - RCP 113 - Search Transaction Optional Format Argument - Add JSON

Version 1.7.2

Change proposals that were added to the RETS 1.7.2 version:

- RETS Change Proposal 64 - Omnibus Adopted Schemas Revisions and Errata
- RETS Change Proposal 66 - Deprecate Lookup Types LookupBitmask and LookupBitstring
- RETS Change Proposal 71 - Time Zone Data
- RETS Change Proposal 72 - LookupType String Length

RETS Change Proposal 64 - Omnibus Adopted Schemas Revisions and Errata

Information

Change Proposal Number: 64
Change Proposal Title: Omnibus Adopted Schemas Revisions and Errata
Originating Workgroup: Schema
Date: March 31, 2008
Version: 1.0.0
RETS Version: Data Schemas
Status: Adopted
Submitted Date: March 30, 2008
Voting Date:
Contact Information
Author: Paul Stusiak
Organization: Falcon Technologies Corp.
Telephone:
E-mail: pstusiak@falcontechnologies.com

Synopsis

The change proposal modifies the adopted schemas as of the April 2008 trimester meeting to resolve some issues discovered in implementations of the schemas and to better describe the intent of the schemas through renaming of certain elements.

Rationale

The change proposal provides clarifications and corrections to the existing schemas.

Proposal

The following schemas are to be modified:
Offices.xsd,
Members.xsd,
Teams.xsd,
Person.xsd,
ContactMethods.xsd,
Address.xsd.
Financial.xsd,
Listings.xsd.
The modifications are as follows.
Offices.xsd and Members.xsd - change the name of the element from MLSLicensing to ProfessionalLicensing and the type from RELicense to ProfessionalLicenseType to reflect the more general use of licensing to include appraisers and auction.
Offices.xsd and Members.xsd - make consistent use of the cardinality maxOccurs on each element to remove ambiguity by explicitly showing the cases of maxOccurs="1". maxOccurs="1" is the default, therefore the change in this case creates no functional difference.
Members.xsd - change the ModificationTimestamp element to be required to match the other top level schema.
Members.xsd - add the missing retsid values for the enumeration MLSMembershipStatusEnum.
Members.xsd - add MembersMediaItems element to provide media for members.
Members.xsd – add a container for EligibleRoles and a string for a list of BillingCodes.
Offices.xsd - change the type definition nrds:OfficeType to nrds:OfficeCategory in NRDSCommons to resolve an ambiguous reference error in certain parser-generators that may be used with the schema to validate and/or generate instances or code stubs. This also makes the archetype more consistent with the general pattern that complexTypes take the ending 'Type' while elements that describe the classification of the concept take the ending 'Category'.
Offices.xsd - rename Media to OfficeMediaItems to be consistent.
Teams.xsd - add TeamMediaItems element to provide media for teams.
Person.xsd - resolve a retsid conflict with retsid=100016 being repeated twice in the document. Change the second instance of the id on element ContactMethods to 101612.
Address.xsd - move the unit type out of the sequence to stand alone at the same level as City and ProvinceOrState at the request of the Syndication workgroup. Change the type definitions to use the pattern of 'Type'.
Address.xsd - correct the missing retsids for the AddressCategoryEnumeration.
ContactMethods.xsd - correct the missing retsids for the enumerations.Financial.xsd - resolve a retsid conflict with retsid=101280 being repeated twice in the document. Change the second instance of the id on enumeration Other - Federal to 101604.
Financial.xsd - resolve a retsid conflict with retsid=100161 - 100163 being repeated twice in the document. Change the first instance of the id on

elements of ClosingFinancingType to 101604.
Listings.xsd - resolve a retsid conflict with retsid=100384 being repeated twice in the document. Change the second instance of the id on element ClosingTerms to 101597.
Listings.xsd - resolve a retsid conflict with retsid=100359 being repeated twice in the document. Change the first instance of the id on element Financing of type ClosingFinancingType to 101598.
The namespace of each affected schema will be updated to 2008-04.

Impact

For each of the changes that involves a retsid or documentation change, there should be no impact. For changes that rename existing fields or restructure the sequence of fields, the impact will be minor to medium if the schema has been used to generate code stubs. If the schema has not been used to general code, the impact should be minor.

Compatibility

The changes may break compatibility when mapping between existing implementations of the schema and applications that depend on those schemas. Given the small number of implementations of the schema, the changes should be broadly compatible.

Document History

Date	Version	Author	Description
2008-03-29	1.0	Paul Stusiak	Initial Release
2008-04-10	1.1	Paul Stusiak	Added namespace version update

RETS Change Proposal 66 - Deprecate Lookup Types LookupBitmask and LookupBitstring

Note: This RCP Affects the Following Sections
- Section 11.4.3 Lookup Type

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Information

Change Proposal Number: 66
Change Proposal Title: Deprecate Lookup Types LookupBitmask and LookupBitString
Originating Workgroup: RETS 1.7.1 Document
Date: June 9th, 2008
Version: 1.0.0.
RETS Version: 1.8
Status: Adopted
Submitted Date: June 9th, 2008
Voting Date: [{review board authorized voting date for the change proposal}]
Contact Information
Author: Mark Lesswing
Organization: National Association of REALTORS(R)
Telephone:
E-mail: mlesswing@realtors.org
Author: Paul Stusiak
Organization: Falcon Technologies Corporation
Telephone:
E-mail: pstusiak@falcontechnologies.com

Synopsis

The lookup types LookupBitmask and LookupBitstring be removed from the standard and be marked deprecated because they represent implementation details and are not generally used, making the documentation of LookupType unnecessarily complex.

Rationale

Confusion has existed around the LookupType metadata which has three different types of lookup representation. These three different types exist to provide some direct mapping between the standard and certain relational databases. In general use, most if not all implementations have settled on a single form, the direct mapping, where each lookup triplet (LongValue, ShortValue and Value) is represented by a single lookup Value. Having unused lookup formats does not improve the interoperability of the standard and requires additional code on the part of client vendors to support this limited feature, making a barrier to entry to the standard.

Proposal

The proposal will modify Section 11.4.3, Table 11-10, removing the definitions for LookupBitmask and LookupBitstring. The existing text,
The value to be sent to the server when performing a search. This field must be numeric for LookupBitmask and LookupBitstring types. For LookupBitmask fields, 2(value-1) is used to compute this component as part of the applicable choices. For LookupBitstring fields, this is the position with in the field, 1-based, at which the value contains a"1".
will be replaced by
The value to be sent to the server when performing a search.

Impact

This change will remove LookupBitstring and LookupBitmask from the standard and from the compliance tester. Existing systems that provide this feature may still have this feature, but it will be an extension to the standard. Client applications will not work with such systems without additional coding and documentation to describe how the form is used.
Given that the premise of the change proposal is that this is not a widely used form, the actual impact will be limited or no impact.

Compatibility

This change will be compatible with RETS 1.8 and higher.h3.**Document History**

Date	Version	Author	Description
June 9, 2008	1.0.0	Paul Stusiak	Initial Release

RETS Change Proposal 71 - Time Zone Data

Original document: RCP 71 Time Zone Data

Note: This RCP Affects the Following Sections:

- Section 2.4 – Atoms and Primitive Entities
- Section 3.6 – Server Response Header Fields
- Section 4.4.2 – SavedMetadataTimestamp Argument
- Section 4.7.3 – Metadata Version Information
- Section 4.8.2 – Access Control Information
- Section 5.11 – Multipart Responses
- Section 7.7.1 – Query Language BNF
- Section 7.7.2 – Query Language Interpretation
- Section 11.2.1 – System Metadata
- Section 11.3.2 – Table
- Section 13.3(13.4) – Transmission Standards
- Appendix B – Sample COMPACT Metadata Responses

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Information

Author: Matthew McGuire
Organization: MarketLinx Inc.
Telephone Number: (865) 470-1500
Address: 1400 Centerpoint Blvd. Suite 100, Knoxville, TN 37932
Email: mmcguire@marketlinx.com
Status: Adopted
Date: July 2, 2007
Version: 1.7d6
Incorporated in Version: 1.7.2

1 Synopsis

This proposal refines the formatting and use of Date and Time data with regards to the application of Time Zones. To that effect this proposal will define how Time Zone data can be communicated by the server using metadata and data representation. This proposal will also attempt to clarify any date or time specific inconsistencies in the specification that may have an affect on the interpretation or use of time zone information.

2 Rationale

The specification currently provides various time and date-time data points to communicate time information about the server and response content. However the specification does not define a method or practice for handling data in default or multiple time-zones. Presently the specification also requires all Search requests to be formed in GMT and translated by the server to the appropriate time zone. However this does not address the time zone of data returned by the server and therefore only obscures the actual time zone for the data returned to the client. RETS 1.7d6 Section 15 – Server Information attempts to address this outside of the existing metadata model. Although the attempt minimizes impact on the existing metadata formats, it is unclear how the Server Information response integrates with the existing metadata model or how the client application would be able to use this information effectively for either metadata or content data. Additionally the Server Information response does not address the presentation of time zone data in search response bodies. This proposal attempts to resolve these shortcomings by redefining the time and date-time formats where necessary and adds additional metadata to the specification in order to provide a precise representation of data on the server.

3 Proposal

3.1 Specification Changes

The following sections detail each area of the existing specification that needs to be changed or clarified and provides reasoning related to each change. Each are of the change will be listed according to the section of the specification using the specification numbering in italics. For example changes to the METADATA-SYSTEM response format would look like the following: "Section 11.2.1".

Section 2.4 – Atoms and Primitive Entities

This section defines a series of tokens for use within the Augmented BNF content of the specification. Presently only DATE exists to represent a time format in the BNF. Additionally the DATE token is defined as "Date using the format defined in RFC 1123" which is incongruent with the remaining specification where ISO8601 is defined in most cases such as the Search Query BNF. To resolve this, the following segment of BNF replaces the BNF for DATE in section 2.4 and removes the reference to RFC 1123. For simplicity throughout changes to the specification the definitions for DATE, TIME, and DATETIME are redefined to use the appropriate definitions below.

date-fullyear::= 4DIGIT
date-month::= 2DIGIT ; 01-12
date-mday::= 2DIGIT ; restriction based on month/year
time-hour::= 2DIGIT ; 00-23
time-minute::= 2DIGIT ; 00-59
time-second::= 2DIGIT ; restriction based on leap second rules
time-secfrac::= "." 1*DIGIT
time-noffset::= ("+" | "-") time-hour ":" time-minute
time-offset::= "Z" | time-noffset
partial-time::= time-hour ":" time-minute ":" time-second [time-secfrac]
full-date::= date-fullyear "~~" date-month "~~" date-mday
full-time::= partial-time [time-offset] time-offset
date-time::= full-date "T" full-time
ISODATE ::= full-date
ISOTIME ::= full-time
ISODATETIME::= date-time
ISOTIMEZONE::= time-offset
DATE::= RFC 1123 date format which MUST be GMT

Note that DATE is defined for use throughout the specification and all reference to DATE must be reviewed and modified to be ISODATE, ISOTIME, or ISODATETIME as appropriate. However to allow for backwards compatibility the DATE definition which is used throughout the Metadata hierarchy is not changed other than to require that this date represent time in GMT. This additional requirement assures that metadata timestamps and other metadata related content is consistently represented across the server. This is also consistent with the usage of the date format in HTTP and has been chosen to reduce confusion. The ISODATE, ISOTIME, and ISODATETIME definitions are provided to enable time zone aware dates and times in content response bodies and search criteria. The ISOTIMEZONE definition simply abstracts the time-offset definition for naming consistency. This definition will be used to communicate the default and current time zone offset of data communicated by the server.

Section 3.6 – Server Response Header Fields

The existing specification explicitly requires the use of the DATE HTTP header. This date format must conform to the format defined in RFC 2616. As a result dates within HTTP headers must use the DATE definition in Section 2.4. Although there are no changes to the language of this section this note is provided for clarity.

Section 4.4.2 – SavedMetadataTimestamp Argument

This argument is defined in order to communicate the metadata timestamp of cached by the client application. As a result this date format should be consistent with the existing metadata timestamps and is represented in RFC **WHA2616 T** format. The additional requirement that this date represent GMT time still applies. This note was provided for clarity.

Section 4.7.3 – Metadata Version Information

The metadata versioning mechanism defines two timestamps using the DATE primitive. These two dates must represent GMT time as described in the BNF in section 2.4. The remaining use of the DATE definition will not be detailed as all use of this definition will conform to the description here. This note was provided for clarity.

Section 4.8.2 – Access Control Information

This section of the specification defines a method for communicating session timeout information to a client application. The necessary change applies to the following text:

pwd-expire-key ::= Expr = pwd-expr , expr-warn-per CRLF

Indicates when a users password will expire. The parameter pwd-expr is a date in RFC 1123 format. And expr-warn-per is the number of days (1*3DIGIT) prior to expiration that the user should be warned of the upcoming password expiration. A expr-warn-per value of (-1) indicates that the password expiration is disabled.

This text refers to RFC1123 and should be replaced with the following text.

pwd-expire-key ::= Expr = pwd-expire-date "," pwd-expire-warn CRLF

pwd-expire-date ::= ISODATETIME
pwd-expire-warn ::= 1*3DIGIT

The pwd-expire-key indicates when a user's password will expire. The second value, listed as pwd-expire-warn, represents the number of days prior to expiration that the user should be warned of the upcoming password expiration. A pwd-expire-warn value of (-1) indicates that the password expiration is disabled.

The above changes set the format of the response to ISODATETIME format to support time zone in the data. If time zone is not declared in the data the time zone defaults to the System TimeZoneOffset metadata information.

Section 5.11 – Multipart Responses

The subsection 5.11.1 defines the general formatting for multipart-mime content as used by the Get Object specification. Since this format is based on the existing multipart-mime format, the format of the Date header value must conform to the multipart-mime standard defined by RFC 2045, which in turn depends on RFC 1123. For the purpose of RETS multipart-mime content this date MUST represent GMT time as defined in Section 2.4.

Section 7.7.1 – Query Language BNF

The existing query language BNF contains date formatting defined in a manner consistent with ISO date and time, but does not allow for communicating time zone in the search or the response. To correct this, the BNF is modified to the following which references section 2.4 as appropriate.

date ::= ISODATE | TODAY
time ::= ISOTIME | NOW
datetime ::= ISODATETIME | NOW

Since the rest of the date time format is detailed in section 2.4 the following items are removed from the Query Language BNF.

fraction, second, minute, hour, day, month, year

Section 7.7.2 – Query Language Interpretation

This section details how the search query information is interpreted based on the type and use of the query data. This section contains the following text which must be modified to handle time zone information in the query.

All datetimes submitted in queries MUST be in GMT. All other dates or times are interpreted in host time. The host MUST interpret the token NOW as the current date and time, and the token TODAY as the current date. If a DateTime is used in a Date context, the host MUST either reject the expression or interpret the date as a datetime with a time of 0000.

Since this text declares a mandatory default time zone it must be changed to the following for this proposal.

Dates and times submitted in a query MAY declare time zone using the time zone formatting of the ISODATETIME and ISOTIME format. If a time is submitted with time zone data it MUST be interpreted according to the time zone declared in the submitted query. If the time zone is not declared in the query the server MUST interpret the submitted time according to the System default time zone offset. If no TimeZoneOffset is declared for the System the default time zone MUST be interpreted as UTC/GMT. The use of the TODAY ~~and NOW~~ tokens MUST also adhere to the time zone offset declared in the metadata. If a DateTime is submitted in the context of a Date, the server MUST either reject the date or interpret the date as a DateTime with a Time of 00:00:00 adhering to the time zone offset declared in the metadata. For backwards compatibility if a client application requests a RETS version of 1.7d6 or lower, the server MUST treat all submitted dates and times without time zone data as UTC/GMT. In this case the advertised time zone offset is ignored since the client is not expected to be aware of the correct offset.

The query language interpretation is written this way to support limited backwards compatibility. Servers which do not expose TimeZoneOffset metadata are required to use UTC/GMT as the default. Additionally the server is not required to send time zone offset for all date time content in responses. As a result if a server has no time zone offset metadata and does not send time zone data in the response bodies, the server functions the same as prior versions and adheres to UTC/GMT as defined.

Section 11.2.1 – System Metadata

The existing System Metadata response does not declare any time zone information for the date and time data of the server. To facilitate this, the following response data format is modified to the following.

<METADATA-SYSTEM Version="system-version" Date="system-date" **[TimeZoneOffset="time-zone-offset"]** > I would make this a separate element. The attributes usually define which part of Metadata is going on (the position of this piece of metadata in the Metadata hierarchy), while elements provide the metadata info (the data being communicated by this metadata).
<SYSTEM SystemID="code-name" SystemDescription="long-name " />
[<COMMENTS>
*(comment)
</COMMENTS>]
</METADATA-SYSTEM><METADATA-SYSTEM Version="system-version" Date="system-date" >
<SYSTEM SystemID="code-name" SystemDescription="long-name" TimeZoneOffset="time-zone-offset"/>
[<COMMENTS>
*(comment)
</COMMENTS>]
</METADATA-SYSTEM>

Where in this metadata format the ISOTIMEZONE is define in Section 2.4. Please note that the Date attribute value is unchanged and remains in the legacy date format taken from RFC 1123. The following subsection is added in order to describe the time-zone-offset format in the response body.

System Default Time Zone
time-zone-offset ::= ISOTIMEZONE

To support time zone aware functionality the server MAY provide time zone offset data in the System metadata. The optional TimeZoneOffset value declares the time zone as a time offset from UTC. The format is defined in Section 2.4. The server adheres to this value as defined in Section 7.7.1. Client applications SHOULD use this value in order to calculate the correct date and time criteria for requests.

This additional section directly refers to the Query Language interpretation and explicitly recommends that Client application use this value when calculating the correct dates and times for requests. This is made clear to encourage better support for incremental updates on both server and client implementations.

Section 11.3.2 – Table

To support communicating the time zone in a response the definition of field DataType must be modified to support the ISODATETIME and ISOTIME formatting. To do this the following definition of DataType should be used in ***Table 11-9***.

DataType	Date	A date in ISODATE format. (see section 2.4)
	DateTime	A date and time in ISODATETIME format. (see section 2.4)
	Time	A time in ISOTIME format. (see section 2.4)

This redefinition refers to Section 2.4 for clarity. Note that the time zone data is optional and when unused the date retains the format prior to this change proposal. **How is the timezone optional?**

Section 13.3 – Transmission Standards
This section defines the formatting for COMPACT and COMPACT-DECODED data in response bodies. Since Table 13-1 defines the date and time formats it should be changed to the following.

Date	A date in ISODATE format. (see section 2.4)
Time	A time in ISOTIME format. (see section 2.4)
DateTime	A date and time in ISODATETIME format. (see section 2.4)

This redefinition again refers to the formal definition in Section 2.4 for clarity. Note here that the time zone formatting is optional and servers returning the plain time or date time values will not be affected. **How is the timezone optional?**
Appendix B – Sample COMPACT Metadata Responses
This section contains many examples that would need to be updated according to the proposed changes above. I have omitted them for brevity.

4. Compatibility

Since this proposal directly affects the use and representation of existing data types in metadata or content there are possible backwards compatibility issues to address. This section will list the areas where changes defined above may or will produce changes that are not backwards compatible. Areas that are not affected will be omitted. If an area appears to be omitted in error please refer to the changes defined above for suggested best practices or interpretations of the specification as they apply to this proposal.
Metadata Format Complication
The addition of a new Metadata attribute or field for System, Resource, and Class changes the DTD for validating the Metadata responses. This will be a compatibility concern as old clients and servers may not accept the new XML nodes and fail to parse them correctly. Although document validation should be associated to the DTD declaration in the response body it is possible that client and server applications cache this DTD locally for performance reasons. Any hosts that do not adopt the new DTD will fail document validation.
Query Language BNF
Changes to the Query Language BNF will require servers to adopt the formatting in their query parsing software. For some platforms this may be difficult. Fortunately most modern software platforms recognize the ISO8601 date time formatting as a valid representation of a date or time. This may mitigate any difficulty in parsing dates and times that contain full time zone data.
Atoms and Primitives
Although I have tried to isolate all of the existing locations affected by updating the Atoms and Primitives, it is possible that some parts of the specification refer to the DATE format in a conflicting manner. The additional tokens ISODATE, ISODATETIME, and ISOTIME were added to help prevent any conflict in the specification. Adding new BNF may or may not be a problem depending on the on how a software vendor chooses to use it. Presently the BNF used by the RETS 1.x specification line does not work with most parser generators. Therefore it is assumed that adding to the BNF will not produce a backwards compatibility issue.
Server Information Request
The server information request appears to be an entirely new capability that was added to the specification in order to resolve a similar issue. I have not declared any modifications to this section or declared a removal of it. This change proposal seeks to resolve the time zone issue from within the existing design of the RETS 1.x metadata system. To that end the Server Information approach has been left intact for future discussion.

RETS Change Proposal 72 - LookupType String Length

Original document: RCP 72 - LookupType String Length

Note: This RCP Affects the Following Sections

- Section 11.4.3 Lookup Type

The above sections have been updated or modified since the previous RETS version.

Click the links above to go to sections having changes adopted in this version.

Information

Author: Matthew McGuire
Organization: MarketLinx Inc.
Telephone Number: (865) 470-1500
Address: 1400 Centerpoint Blvd. Suite 100, Knoxville, TN 37932
Email: mmcguire@marketlinx.com

Status: Adopted
Date: Nov. 14, 2007
Version: 1.7d6
Incorporated in Version: 1.7.2

1 Synopsis

This proposal addresses the string length for Long Value, Short Value, and Value of Lookup Type metadata. Presently the Short Value and Value lengths are set to 32 characters. Although this encourages using the values for strings shorter than the long value, there are instances where both the Short Value and Value may be longer than 32 characters within the database of the vendor system.

2 Rationale

The Lookup Type metadata in the specification is used to describe coded values for records that may contain one or more values. Conceptually these coded values will typically be short strings or even numeric codes that represent longer human readable strings such as City or Street names. However in some cases these values can be longer than 32 characters. For instance if a system contains School Names and stores the names as full length strings there are no encoded values for the Long Value. In this instance the Long, Short, and "actual" values are the same string as no encoding is performed. Although uncommon this is a valid practice and the specification should not prevent a system from representing Lookup Values as appropriate to the implementation.

3 Proposal

3.1 Specification Changes

Section 11.4.3 Lookup Type
This section defines the structure of Lookup Type metadata. Here Table 11-20 defines the character representation of Long Value and Short Value and Value as the following:
LongValue 1*128TEXT
ShortValue 1*32TEXT
Value 1*32ALPHANUM
This proposal recommends that the above character representations be changed to the following:
LongValue 1*128PLAINTEXT
ShortValue 1*128PLAINTEXT
Value 1*128PLAINTEXT | 1*32ALPHANUM
These changes will assure that system providers may represent the lookup values as appropriate to the system containing the data exposed by the RETS server.

4. Compatibility

The changes in this proposal would allow for all previous Lookup Type metadata content. Therefore there appears to be no backwards compatibility issues with this proposal.

Version 1.8.0

Change proposals that were added to the RETS 1.8.0 version:

- RCP 59 - Revised Update Transaction
- RCP 60 - Metadata Changes for Update
- RCP 61 - Validation Expression Replacement
- RCP 63 - Object Data and Upload
- RCP 65 - Session information tokens
- RCP 68 - Search Has Key Index Support
- RCP 69 - LookupType Value
- RCP 70 - Metadata Role Support
- RCP 74 - Location Availability in Object Metadata
- RCP 75 - Offset Availability in the Metadata
- RCP 76 - GetPayloadList
- RCP 77 - Maximum Field Length
- RCP 78 - Specification Errata Changes
- RCP 79 - Add Preferred Flag to GetObject Responses
- RCP 80 - Optional Query
- RCP 82 - LookupMulti Quoting Rule
- RCP 87 - RETS 1.7.2 Errata Document
- RCP 90 - Deprecate CommonInterest Class Well-Known Name
- RCP 91 - StandardNames Version Information in Login Transaction
- RCP 92 - RESO Payload Transport-Level Metadata Support
- RCP 93 - Add Content-Sub-Description to GetObject
- RCP 94 - Improved Error Handling in GetObject
- RCP 98 - Additional Information Fields in METADATA-SYSTEM and Login
- RCP 99 - Mixing StandardNames and SystemNames
- RCP 100 - Alternate Standard Names
- RCP 101 - Child Rows Support

- RCP 102 - GetObject URL as Default Location

RCP 59 - Revised Update Transaction

Original document: RCP 59 Revised Update Transaction

Note: This RCP Affects the Following Sections:

- Section 10.1 Required Request Arguments
- Section 10.2 Optional Request Arguments
- Section 10.5 Update Response Body Format
- Section 10.6 Record Locking
- Section 10.8 Reply Codes
- Section 11.3.3 Update

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Sergio Del Rio
Organization	Templates 4 Business, Inc.
E-mail	Sergio dot Del dot Rio at t4bi dot com
Author	Libor Viktorin
Organization	MarketLinx
Email	lviktorin at marketlinx dot com
Submitted Date:	March 18, 2005
Revision Date:	August 7, 2005
Revised Date	October 30, 2007
RETS Version	1.8.0
Status	Adopted

1. Synopsis

This change request consists of a set of changes that are designed to enhance the functionality of the RETS 1.7 specification. These enhancements are geared towards enabling a fully metadata driven RETS Update client.

2. Rationale

We have been striving to create RETS Update Client that is 100% metadata driven, yet still presents itself to users as a fully functional update client that presents data to them in a way that they are comfortable with.

When looking at what is available in the specification as it stands today, we realized that this goal was not fully attainable unless some changes were made to the specification. With that in mind, we set out to build it.

With all the changes that are presented below, we were able to build a RETS Update Client that is as functional as we believe is necessary to provide users with a good user experience and allow them to enter their listing data in the most effective way possible. This client is now in beta testing and is almost identical to it's predecessor which was interacting directly with the MLS database.

3. Proposal

This proposal affects several sections of the RETS 1.7.2 document as follows.
10.1 Required Request Arguments

Each Update request MUST specify following arguments:
Resource = *resource -name***ClassName =** *class -name***Validate =** *validate -flag***Action =** *update-action*
Record = *field-name* **=** *field-value* **(** ***field-delimiter field-name* *=** *field-value*)
resource-name ::= *1*32ALPHANUM*The name of the resource to be updated, as specified in the metadata. This is the SystemName as defined in Section 11.2.2.
class-name ::= *1*24ALPHANUMRETSNAME*The name of the class to be updated, as defined in the metadata. This is the ClassName as defined in section 11.3.1.
validate-flag ::= **0** | **1** | **2**
If this parameter is set to one ("**1**"), then the partial record is validated by the host; any fields that are not provided are not validated. Any fields with metadata field "Attributes" set to "Autopop" in the metadata (see Section 11.3.4) will have their field values filled in by the server and returned to the client. This mode is used primarily for auto-population of the data record. The record in the server database is not updated. If this entry is set to zero ("**0**") and there are no errors in the record the record is updated on the server. If this entry is set to two ("2"), the server validates all

fields and returns any errors found, but does *not* store the record.
update-action ::= *1*24 ALPHANUM*The action to perform, as specified by the metadata. This is the UpdateAction as defined in Section 11.3.4.
field-name ::= *RETSNAME* Changed to RETSNAME
The name of the field to be updated, as specified in the metadata. This is the SystemName as defined in Section 11.3.2.
field-delimiter ::= *OCTET*Field delimiter as specified by the optional argument Delimiter, or ASCII HT character (09) if that argument is missing.
field-value ::= <varies depending on the field>
The text representation of the field value as defined by the metadata in Section 11.3.2 subject to the business rules. The value MUST be submitted as if in COMPACT format.
The Record specifies values for fields that are to be changed, or are otherwise important for the update transaction. For Clone, Change and Delete update actions, the value for the KeyField (as defined in 11.3.3) MUST be specified, to identify a record which is to be modified. ~~If an Update transaction with the same KeyField value has been previously answered by the server, the client MUST include in the Record argument at least all the fields returned by the server in the previous response~~.

10.2 Optional Request Arguments

Delimiter = *HEX HEX*
Specifies the octet which will separate fields in the record. If this is not specified, an ASCII HT character (09) is assumed.
WarningResponse = *warning-response* *(*field-delimiter warning-response*)
warning-response::= *field:warning-num* **=** *user-response*
warning-num ::= *1*16DIGIT*
Fixed warning-response to include field and increased size of warning-num*user-response* ::= **1024TEXT excluding* delimiter Increased size of user-response
The *warning-num* value is the host warning number that was returned in the prior Update Response body. The *user-response* value is the text of the warning response in response to the specified warning. If a *warning-num* sent in the prior UpdateResponse body had a *response-required* value of 2, then the *user-response* value MUST NOT be null.
Lock = *lock-time*
lock-time ::= 1*DIGIT
Specifies a request to lock the requested record to avoid changes by other clients.
If the lock-time is a number greater than zero, the server SHOULD lock the requested record for the next *lock-time* seconds. If the record has already been locked by this user, the lock timeout is extended to the next *lock-time* seconds. Server MAY use a lower lock timeout, or ignore the lock request completely: the requested *lock-time* is merely a hint from the client about how long it may take before final update on the record will be requested.
If the lock-time argument is missing, the server will use its own discretion about locking the record.
Negative or zero value in lock-time request argument means that the client does not want to lock the record. The server will still use its own locking policy. Namely, if the UpdateAction is BeginUpdate, the server MAY lock the record, and the client MUST request another Update transaction to release the lock. Because of possible connectivity problems, however, the server should not rely on the next request to come, and should implement a lock-expiring strategy.
LockKey = *lock-key*
If the server locks the requested record, it MAY return a lock-key. The lock-key value MUST be sent back in the next Update transaction to allow the server to verify that the client operates with the values the record had at the time it had been locked.
If the client fails to provide the correct lock-key, the server MAY fail the update request. In that case, the client has to request another BeginUpdate action to lock the record and get the actual data, and then request the needed Update request with the new lock-key.

Select = *field-name* **(**,* *field-name*)
Specifies a list of fields that should be returned in the response. Server MUST return current (updated) values for all fields in this list; ~~the server MAY return more fields if it considers them needed for a proper update transaction~~.
The field-names in the Select argument must be the Standard names as defined in the Table metadata (11.3.2).

10.5 Update Response Body Format

The body of the update response has the following format when there are no errors:
<RETS *1*SP* **ReplyCode=** *quoted-reply-code 1*SP***ReplyText=** *quoted-string* ***SP* *>** *CRLF* transaction-id-tag
[lock-tag]
[delimiter-tag]
column-tag
compact-data
[<RETS-STATUS *1*SP* **ReplyCode=** *quoted-end-reply-code 1*SP***ReplyText=** *quoted-string* ***SP***/>*
</RETS> *CRLF*
The body of the update response has the following format when there are errors or warnings:
<RETS *1*SP* **ReplyCode=** *quoted-reply-code 1*SP***ReplyText=** *quoted-string* ***SP* *>** *CRLF* transaction-id-tag
[lock-tag]
[lock-key]
[delimiter-tag]
column-tag
compact-data
[error-block]
[warning-block]
</RETS> *CRLF*
Lock-tag ::= **Lock=***lock-time*
The lock-time is the number for which the record will be locked on the server. If lock-time is zero, the lock has been already released. Note that the lock-time MAY be less than requested, if the server uses any internal logic to limit the time for which a record will be locked.
Server which supports record locking MUST return the Lock-tag in the response. If no lock was requested, server which supports locking MUST return Lock=0. Missing Lock-tag indicates that the server does not support record locking.
Lock-key::= **LockKey**=*lock-key*

If the server locks the requested record, it MAY return a lock-key. The lock-key value MUST be sent back in the next Update transaction to allow the server to verify that the client operates with the values the record had at the time it had been locked.
If the client fails to provide the correct lock-key, the server MAY fail the update request. In that case, the client has to request another BeginUpdate action to lock the record and get the actual data, and then request the needed Update request with the new lock-key.
If a BeginUpdate action was requested on a record that is already locked by the same user, the LockKey, if present, MUST be the same as when the record was originally locked. That allows the client to verify that the record has not changed since the original lock has been granted.

error-block ::=
<ERRORBLOCK > *CRLF* 1*(**<ERRORDATA>***fielderror-numerror-offseterror-text***</ERRORDATA>**) **</ERRORBLOCK>**
warning-block ::=
<WARNINGB LOCK>
1*(**<WARNINGDATA>***fieldwarning-numwarning-offset*
*warning-textresponse-required***</WARNINGD ATA>**)
</WARNINGBLOCK>
The format of the <ERRORDATA> and <WARNINGDATA> tag content is identical to COMPACT format.

10.5.1 Error block

An Error Block is returned when there is a problem with one or more of the fields. The error block contains information about the fields that have errors. It contains the field name, an error number, some additional text about the error (*error-text*), and where in the field data the error occurred (*error-offset*).
error-num ::= *1*16DIGIT* Increased Size.
This is the host error number. This number along with the *error-text* MAY be displayed to the user when looking at the corresponding field in the client application.
error-offset ::= *1*5DIGIT*This is the offset into the field data that was sent by the client application to the server. It indicates at what character in the field data the problem was encountered. This number is set to zero ("0") if the offset of the error is unknown.
error-text ::= **TEXT* Increased size by making it unlimited.
This is the error text generated by the host to assist the user in determining the problem with the field data. This text is associated with the *error-num*.
The error return format follows the COMPACT data format in all particulars. This affects primarily the quoting of special characters and the selection of the delimiter that separates the field values. In effect, the error return is a COMPACT data block without the usual COLUMNS element.

10.5.2 Warning block

A Warning Block is returned when there is a problem with one or more of the fields that would not prevent the record from being saved in the database. It contains a field name, a warning number, some additional text about the warning (*warning-text*), where in the field data the warning occurred (*warning-offset*) and an indicator whether an end-user response to this warning is requested or required. The delimiter is the same as the one defined for the *error-block*.
field ::= *1*32ALPHANUM*The SystemName of the field to which the warning applies.
warning-num ::= *1*16DIGIT* Increased Size
The host warning number. This number, along with the *warning-text*, MAY be displayed to an end-user in association with the corresponding field in the client application.
warning-text ::= **TEXT* Increased Size by making it unlimited.
warning-offset ::= *1*5DIGIT*The offset into the field data that was sent by the client application to the server. It indicates at what character in the field data the problem was encountered. This number is set to zero if the offset of the error is unknown or if an offset is inapplicable.
response-required::= **0** | **1** | **2**
The *response-required* value indicates whether an end-user response is requested or required:
0 No response is permitted.
1 A response is requested.
2 A response is mandatory.
If the *response-required* field indicates that a response is mandatory, the client MUST send the end-user response for the specific warning-num in the WarningResponse request argument in order for this record to be saved to the database.

10.6. Locking records

Clients are encouraged to use record locking to avoid intermittent changes by other parties. Typical procedure for updating record in a GUI based client would be as follows:

1. Client requests an Update transaction with UpdateAction=BeginUpdate, Record specifying only the key value of the requested record, and (optionally) Lock set to several minutes. Server locks the requested record and sends its current data.
2. Client presents the current data to the user and lets them make necessary changes. If the lock-time returned by the server expires before the user is ready, the client requests another Update transaction with UpdateAction=BeginUpdate , Record specifying only the key value of the requested record, and Lock set to several more minutes. The server extends the lock on the record. If returning the LockKey, the server returns the value returned with the original lock, so that the client may make sure that nothing changed in the mean time.
3. If the client input form has several pages and the client wants to update the record one page at a time, it requests Update transactions with Record specifying the data collected on each page and Lock set to several more minutes. The server updates the requested part of the record and keeps it locked. It returns the original LockKey to show that the lock has not been released.
4. After the user has finished filling in new data, the client requests a final Update transaction with Lock=0. The server updates the record for the last time and releases the lock.

Typical procedure for updating record in non-GUI based client would be as follows:

1. Client requests an Update transaction with Record specifying only the key value of the requested record, and Lock set to several

seconds. Server locks the requested record and sends its current data.

2. Client calculates new data and requests an Update transaction with Record specifying all the new data and Lock=0. Server updates the record and releases the lock.

If in any case the client fails to continue requesting Update transactions as described above, the server will release the record lock when the lock-time expires.
The client may try to send a "real" update request without first requesting the BeginUpdate action. If a server allows for updating a non-locked record, then the record is updated, the server sends back a response with lock-time being zero (or missing), and that finishes the transaction. If the server does require the BeginUpdate action, it will return with error 20314. Not that the server MUST specify RequiresBegin=1 in the metadata if it requires the BeginUpdate action.
The server may not implement the locking mechanism. If this is the case, the response to any BeginUpdate action will immediately return without lock-tag, just providing the requested data for a record (client could get the same data using a SEARCH transaction). The missing lock-tag tells the client that locking is not in effect, and the client can come at any time with the Change (or any other) action.

10.7 Reply Codes

Additional reply codes:

20313	Incorrect LockKey. The lock on the record may have been released before the update request came.
20314	Record has not been locked. The BeginUpdate action has to be requested before the Update. (This error will be returned if the server requires the BeginUpdate, and the request did not specify the LockKey, indicating that no lock has been requested).
21315	Record lock expired. The BeginUpdate action has to be requested before the Update. (This error will be returned if the record is not locked, and the request did specify the LockKey, indicating that a lock has been previously requested).
21316	Unknown UpdateAction. Requested UpdateAction is not defined for this class.
21317	Unknown Resource or ClassName.
21318	Requested record does not exist.

11.3.3 Update

Table 11-11 Metadata Content – Update

Metadata Field	Content Type	Description
…		
UpdateAction	1*24ALPHANUM	This identifies the nature of the update, such as "add" or "modify". Some update types, such as changes to a property record (e.g "Sell", "Back on Market"), will imply a set of business rules specific to the server. However, where possible, the following standard type names should be used:
Update Name	Function	
Add	Add a new record	
Clone	Create a new record by copying an old one	
Change	Change an existing record	
Delete	Delete an existing record	
BeginUpdate	MAY be requested before any other Update request to get the specified record's actual data, and to put a lock on the specified record. The server MAY lock the requested record until another Update for that record is requested.	
CancelUpdate	MUST be used after BeginUpdate, if no other update is requested on the locked record. It is not an error to request CancelUpdate on a record that is not locked.	
ShowLocks	Request to show which records are currently locked by this user. The server MUST response with column-tag showing KeyField and LockTime, with copmact-data containing one line for each locked record, showing the KeyField value of the record and number of seconds before the lock will expire.	
RequiresBegin	BOOLEAN	If this value is true (1), the BeginUpdate action MUST be called before this update action.
…		

4. Development Impact

This proposal does not change anything or remove anything from the existing specification, it merely adds additional metadata fields to existing metadata tables.

As such, development impact depends on current RETS client and server implementations, but should not have any impact on future RETS client and server implementations. Each existing RETS server and client, prior to passing of this proposal, have the option to change accordingly to be RETS compliant. Future RETS server and client implementation after approval of RETS 1.7 MUST follow the RETS 1.8 standard.

5. Compatibility

Servers that do not implement any of the additional metadata will remain backwards compatible to RETS 1.7. Current servers that add the additional metadata fields will also remain backwards compatible to RETS 1.7 since the specification allows for additional fields to be returned by servers. Current servers that want to become RETS 1.8 compliant MUST implement all of the additional metadata fields.

RCP 60 - Metadata Changes for Update

Original document: RCP 60 - Metadata Changes for Update

Note: This RCP Affects the Following Sections:

- Section 11.1.1 Metadata Organization
- Section 11.2.3 Foreign Keys
- Section 11.2.4 Filter
- Section 11.3.2 Table
- Section 11.3.4 Update Type
- Section 11.4.2 Lookup
- Section 11.4.3 Lookup Type
- Section 11.4.8 Validation Lookup Type
- Section 11.4.10 Validation External
- Section 11.4.11 Validation External Type
- Section 11.5.1 Column Group Set
- Section 11.5.2 Column Group
- Section 11.5.3 Column Group Control
- Section 11.5.4 Column Group Table
- Section 11.5.5 Column Group Normalization

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Sergio Del Rio
Organization	Templates 4 Business, Inc.
E-mail	Sergio dot Del dot Rio at t4bi dot com
Author	Libor Viktorin
Organization	MarketLinx
Email	lviktorin at marketlinx dot com
Submitted Date:	March 18, 2005
Revised Date	October 30, 2007
RETS Version	1.8.0
Status	Adopted

1. Synopsis

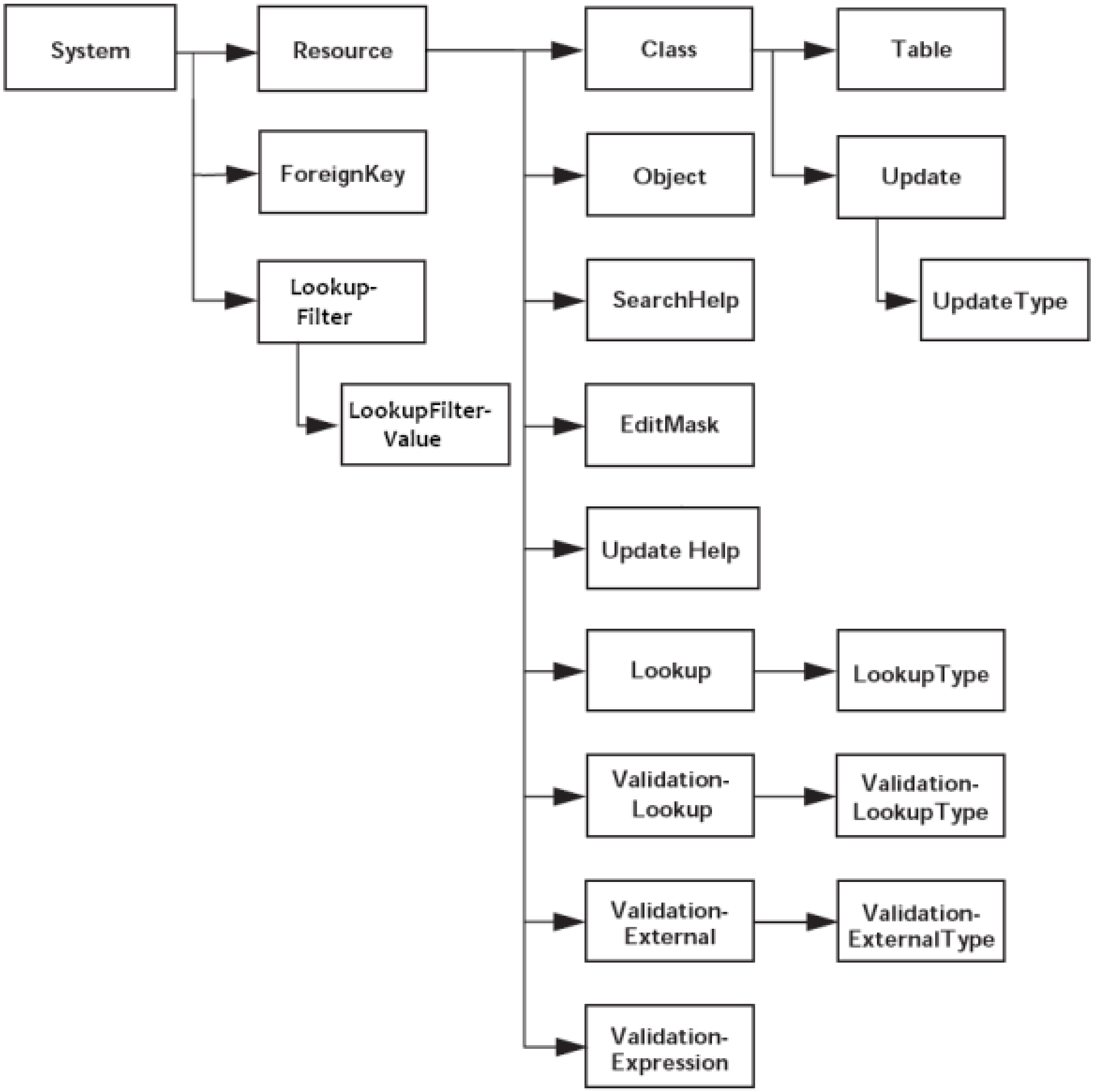
2. Rationale

3. Proposal

This proposal affects several sections of the RETS 1.7.2 document as follows.

Metadata Extensions Existing Metadata Extensions

This section outlines all the existing Metadata Output Formats that have been extended for the Update Transaction.
The following existing Metadata Output Formats have been extended
The corrected and upgraded Metadata structure



METADATA-FILTER

Filters and Filter_values describe a parent-child relation between Lookups by specifying which values in the Child lookup are legal based on the value in Parent lookup. If filtered lookups are used in the Table metadata (meaning a LookupName and ParentField metadata items are not empty), the server MUST guarantee that the values in the child field will not fall outside the set specified by the filter based on the value in the parent field. The client SHOULD use the filter information when checking values for the Update transaction. The LookupFilter argument in GetMetadata can also be used to limit the amount of metadata information sent by the server.
Filter header tag arguments- Version- Date
Filter metadata fields:

FilterID	RETSNAME	A unique ID that represents this filter
ParentResource	RETSNAME	ResourceID of the parent lookup
ParentLookupName	RETSNAME	LookupName of the parent lookup
ChildResource	RETSNAME	ResourceID of the child lookup

ChildLookupName	RETSNAME	LookupName of the child lookup
NotShownByDefault	BOOL	If true the server will by default not include the FilterValue data of this filter in any metadata request, unless specifically asked to using the LookupFilter argument in GetMetadata.

METADATA-FILTER_VALUE

Filter Value tag arguments:- FilterID- Date- Version
Filter metadata fields:

FilterValueID	RETSNAME	A unique ID that represents this FilterValue
ParentValue	1*128ALPHANUM	Value from the LookupType in the parent lookup
ChildValue	1*128ALPHANUM	Value from the LookupType in the child lookup

Metadata Lookup additional field:

FilterID	RETSNAME	FilterID of an existing filter. If present, the range of valid LookupType values in this lookup is limited by the value of a parent lookup.
NotShownByDefault	BOOL	If true, the server will by default not include the LookupType data of this lookup in any metadata request, unless specifically asked to using the LookupFilter argument in GetMetadata. This flag MUST be set to 0 (or empty) unless a FilterID is non-empty

Metadata Table additional field:

FilterParentField	RETSNAME	Specifies that values allowed in this field are limited by the Lookup's filter, using the contents of the field named here as ParentValue. FilterParentField may only be specified with fields that have a LookupName, where the named Lookup has a non-empty FilterID.

Lookup GetMetadata Extensions

Some servers may use lookups that list hundreds of values (like community names or street names). Such lookups will make the full metadata very large. Therefore clients may opt to get metadata with limited sets of values. To do that, an optional argument is allowed in the GetMetadata transaction:
LookupFilterArgument ::= **LookupFilter=(*Filter**)* **(*,(*Filter**)*)

LookupFilter=(=-1)	LookupFilter=(**=**)* Filter::= FilterID **=** ParentValue ParentValue::= *	-1	1*128ALPHANUM If LookupFilter argument is present, the server that implements this extension MUST limit lookup data in the response. Specifically, LookupType data for a Lookup whose FilterID is included in the LookupFilterArgument MUST be limited to the child values for the requested ParentValue. Also, FilterValue data for a filter included in the LookupFilterArgument MUST be limited to those with requested ParentValue. The asterisk (**{}) in place of FilterValue specifies that all values should be listed; *-1** specifies that no values should be listed. If the LookupFilter argument is specified as **(**=-1)*, the server MUST NOT send any LookupType data for lookups that have non-empty FilterID, nor any FilterValue data at all. If the LookupFilter argument is specified as (**=**), the server MUST send all requested LookupTypes without limitations. This extension is meant to be used with requests for LookupType metadata, Filter metadata, or any metadata with ID ending with asterisk ⭐ .

METADATA-TABLE

The following table outlines the extensions to the TABLE metadata that were necessary to make for a better user experience:
Table 11-9.1 Metadata Content – Extensions – Table

Metadata Field	Content Type	Description
Interpretation	URI	An arbitrary URI or URL that is fully qualified and that an application will be able to successfully link to.
DefaultSearchOrder	Numeric	The order that fields should appear in a default search screen that is executed in order to give the user a list of existing records to select from. Fields that should not appear in the default search screen should have a value of 0. Fields that should never be visible to the user should have a value of -1.
FilterParentField	RETSNAME	Specifies that values allowed in this field are limited by the Lookup's filter, using the contents of the field named here as ParentValue. FilterParentField may only be specified with fields that have a LookupName, where the named Lookup has a non-empty FilterID.
Case	UPPER, LOWER, EXACT, MIXED	A value which indicates whether the server stores the data in this text field in the specified case. This allows a client to automatically convert data in these fields to upper case for both searches and updates. EXACT and MIXED indicate that data is stored as entered by the user in the database. EXACT indicates that a case sensitive search will be performed by the server. MIXED indicates that a case insensitive search will be performed by the server. UPPER and LOWER indicate that the server will perform a case insensitive search.

METADATA-LOOKUP_TYPE

The following table outlines the extensions and changes to the LOOKUP_TYPE metadata that were necessary to make for a better user experience:
Table 11-20.1 Metadata Content – Extensions – LookupType

Metadata Field	Content Type	Description
LongValue	1*128PLAINTEXT	The value of the field as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; expected uses include displays on reports and other presentation contexts.
ShortValue	1*32PLAINTEXT	An abbreviated field value that is also localizable and human-readable. Use of this field is implementation-defined; expected uses include picklist values and other human interface elements.
FilterID	RETSNAME	FilterID of an existing filter. If present, the range of valid LookupType values in this lookup is limited by the value of a parent lookup.
NotShownByDefault	BOOL	If true the server will by default not include the LookupType data of this lookup in any metadata request, unless specifically asked to using the LookupFilter argument. This flag MUST be set to 0 (or missing) unless a FilterID is non-empty
DisplayOrder	1*5DIGIT	The order in which to display the Lookup Type within the list for the Lookup. All records returned by the server MUST either all return numbers or all be blank within one lookup. If the DisplayOrder is blank or missing, the client SHOULD display the data in the order in which it is received. The DisplayOrder does not have to be a contiguous numbering scheme but no duplicates are allowed.

We are also proposing to increase the size of ShortValue and Value to 128.

METADATA-UPDATE_TYPE

The following table outlines the extensions to the UPDATE_TYPE metadata that are necessary to make for a better user experience:
Table 11-13.1 Metadata Content – Extensions – Update Type

Metadata Field	Content Type	Description
Attributes	See Table 11-13 in RETS document.	Additional Values Required:
Value	Meaning	Description
6	AutopopRequired	Indicates that this field is mandatory when calling the Update transaction for Auto Population (validate-flag=1)
7	Hidden	Indicates that this field may be used in ValidationExpressions but is to remain hidden from the user.
SearchResultOrder	Numeric	The order that fields should appear in a default one-line search result that is executed in order to give the user a list of existing records to select from for updating them. Fields that should not appear in the default one-line format should have a value of 0. Fields that should never be visible to the user should have a value of -1.
SearchQueryOrder	Numeric	The order that fields should appear in a default search screen that is executed in order to give the user a list of existing record to select from for updating them. Fields that should not appear in the default search screen should have a value of 0. Fields that should never be visible to the user should have a value of -1.

METADATA-FOREIGNKEYS

The following table outlines the extensions to the METADATA-FOREIGNKEYS metadata that were necessary to make for a better user experience:
Table 11-5.1 Metadata Content – Extensions – Metadata ForeignKeys

Metadata Field	Content Type	Description

OneToManyFlag	Boolean	A truth value which indicates whether the foreign key will return multiple rows if queried from the source to the destination.

METADATA-VALIDATION_LOOKUP_TYPE Validation Lookup

The Validation Lookups are now deprecated. METADATA-FILTER should be used instead to specify parent-child relations between lookups.METADATA-VALIDATION_EXPRESSION

The following table outlines the extensions to the METADATA-VALIDATION_EXPRESSION metadata that were necessary to make for a better user experience:

Table 11-37.1 Metadata Content – Extensions – Validation Expression

Metadata Field	Content Type	Description
Message	1*512TEXT	A message to be displayed in the following cases: 1. Expression Type of ACCEPT results in FALSE. 2. Expression Type of REJECT results in TRUE. 3. Expression Type of WARNING results in TRUE.

METADATA-VALIDATION_EXTERNAL

The following table outlines the extensions to the METADATA-VALIDATION_EXTERNAL metadata that were necessary to make for a better user experience:

Table 11-38.1 Metadata Content – Extensions – Validation External

Metadata Field	Content Type	Description
AutoQuery	Boolean	If true, there is no need to display an intermediary query screen to the user before executing a query (after restricting query with RestrictFields) on the external resource. If false, then such a query would return too many rows; in this case, the interface should allow the user to further refine the search by entering data into any of the specified SearchFields before executing the query. In either case, the user SHOULD be allowed to further refine the query after they are given the initial list of results.
AllowDirectEntry	Boolean	If true, users should be allowed to enter data into all enterable fields that are being auto-filled based on the ResultFields of the ValidationExternalType metadata. If false, all fields are not directly enterable by a user and are only auto-filled based on the ResultFields of the ValidationExternalType metadata.
AllowUnlistedData	Boolean	If true, users are not required to enter only data that is found in the ValidationExternal Resource/Class into all fields that are in the LookupButtons list. If false, users MUST enter only data that is found in the ValidationExternal Resource/Class into all fields that are in the LookupButtons list.

METADATA-VALIDATION_EXTERNAL_TYPE

The following table outlines the extensions to the METADATA-VALIDATION_EXTERNAL_TYPE metadata that were necessary to make for a better user experience:

Table 11-40.1 Metadata Content – Extensions – Validation External Type

Metadata Field	Content Type	Description
RestrictFields	1*1024PLAINTEXT	A comma separated list of valid field pairs joined by = (equal) the first is a target field in the table being updated and the second is a source field in the table being searched. The fields use a SystemName from Section 11.3.2.

If values have been entered in the table being updated for any of the above fields, the query executed on the source table must be restricted to the values entered.|

LookupButtons	1*512PLAINTEXT	A comma separated list of fields from the target table that should have lookup buttons displayed for them on a UI. It is expected that clicking on the lookup button will display the ValidationExternal lookukp screen.

Presentation Metadata Extensions

Need to add note to Presentation Metadata Specification that Compliance needs to get involved and determine how Compliance will view this new metadata.

This section outlines all the Metadata Output Formats that have been implemented as RETS 1.7 extensions to metadata that were required to make fully metadata driven Update Client with a functional presentation.

The following Metadata Output Formats have been implemented as RETS Extensions:

METADATA-COLUMN_GROUP_SET

This metadata defines a tree structure which should be used to render the data in any GUI system that is designed in order to satisfy the display requirements of an MLS.

The Column Group Set metadata starts with a <METADATA-COLUMN_GROUP_SET> tag with Resource, Class, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-24, followed by the <DATA> section, which contains the actual field information. The Column Group Set metadata has the following format:

<METADATA-COLUMN_GROUP_SET SP **Resource="***resource-id***"** SP **Class="***class-id***"** SP **Version="***column-group-set-version***"** SP **Date="***column-group-set-date***">**

<COLUMNS>*column-group-set-field* ***(***column-group-set-field***)</COLUMNS>**

(<DATA>column-group-set-data* ***(***column-group-set-data***)</DATA>)**

</METADATA-COLUMN_GROUP_SET>

*resource-id ::= 1*32ALPHANUM*This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.

*class-id ::= 1*32ALPHANUM*This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group Set applies.

*column-group-set-version ::= 1*2DIGITS* ***.*** *1*2DIGITS . 1*5DIGITS*This is the version number of this Column Group Set metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. Every time this Column Group Set metadata changes the version number should be increased.

*column-group-set-date ::= DATE*The latest change date of this Column Group Set metadata.

column-group-set-field ::= <Field Name from Table 11-24>{_}column-group-set-data ::= <valid value as defined in Table 11-24>_An example Table section follows:

GetMetadata request:
Type: METADATA-COLUMN_GROUP_SET
ID Format: Resource : Class
ID Example: Property : RES
Compact reply:
<METADATA-COLUMN_GROUP_SET Resource="Property" Class="RES" Version="1.00.000"
Date= "Sat, 20 Mar 2002 12:03:38 GMT" >
<COLUMNS>MetadataEntryIdColumnGroupSetNameColumnGroupSetParent
SequenceLongNameShortNameDescriptionColumnGroupNamePresentationStylePresentationColumnsURL</COLUMNS>
<DATA>10000123456Residential1Residential ListingResidentialThe top node of the Residential Listing Data Entry Hierarchy</DATA>
<DATA>10000123457WaterFrontResidential1WaterFront InformationWaterFrontDetails about water front for propertyWaterFrontEdit1</DATA>
<DATA>10000123457BedroomsResidential2Bedroom InformationBedroomsDetails about bedrooms for propertyBedroomsMatrix</DATA>
<DATA>10000123457AgentInfoResidential3Agent InformationAgentAgent Website
www.mywebsite.com/agent?Agent=.AGENTCODE.?Listing=.ListingID.</DATA>
</METADATA-COLUMN_GROUP_SET>

Table 11-24 Metadata Content – Column Group Set

Metadata Field	**Content Type**	**Description**
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged. In particular, it should be managed so as to allow the client to detect changes to the ColumnGroupSetName.
ColumnGroupSetName	1*32ALPHANUM	The name that uniquely identifies this Column Group Set within the Class.
ColumnGroupSetParent	1*32ALPHANUM	The ColumnGroupSetName of the Parent Column Group Set. If not specified, this Column Group Set is the top node in the tree.
Sequence	1*5DIGIT	The sequence that this Column Group Set is to be displayed in within it's parent group.
LongName	1*64ALPHANUM	The name of the Column Group Set as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this Column Group Set when it appears on a report.
ShortName	1*32ALPHANUM	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined; it is expected that clients will use this field in human-interface elements such as picklists.
Description	1*256ALPHANUM	A brief description of the purpose for this Column Group Set.
ColumnGroupName	1*32ALPHANUM	The name of the Column Group that is to be displayed in this Column Group Set. If not specified, this Column Group Set is to be treated as a node in the tree that displays no data. The ColumnGroupName must exist in the Column Group metadata for this Class.
PresentationStyle	1*32ALPHANUM	One of the following values: **Edit** – Basic Edit Block displayed in PresentationColumns number of columns. **Matrix** – Expected to be displayed using Normalization Grid. **List** – Show one record per row. **Edit List** – Show one record per row and allow the records to be added, edited and deleted. **GIS Map Search** – Special Case: Can only have 2 columns in Column Group. First column is Latitude and Second column is Longitude. These columns are expected to be filled in with results from GIS Map Search. **URL** – Indicates that this is to simply go to the specified URL, a ColumnGroup name MUST not be specified and a URL MUST be specified for this PresentationStyle.
URL	1*256TEXT	Indicates a URL that is to be accessed using this entry instead of a standard Column Group. You may not specify a ColumnGroupName and a URL. The URL may be formed with place-holders surrounded by the '[' and ']' characters so that a substitution for any valid SystemName within the class being displayed, Info Tokens from the Login Response or Validation Expression Special Operand Tokens as specified in Table 11-20. To differentiate between SystemNames and tokens, an additional character '.' is used to surround the tokens. Example: [http://www.example.com/agent?Agent=[.AGENTCODE.]] &Listing=[ListingID]

ForeignKeyID	1*32ALPHANUM	The identifier of the Foreign Key that is to be displayed in this ColumnGroupSet. If specified, the ForeignKeyID MUST exist in the METADATA-FOREIGNKEY metadata and the Parent MUST be the Property and Class of this ColumnGroupSet. When this is specified, it means that a multi-row block is expected to be displayed to the user within which he can Add, Edit or Delete records of the Child Resource and Class that is specified. Furthermore, the ChildSystemName field should always be filled from data found in the ParentSystemName field to provide for a proper Master/Detail relationship.

Notes: It is important to note that only one of ColumnGroupName, ForeignKeyId or URL may contain data. These three fields are mutually exclusive.

METADATA-COLUMN_GROUPS

This metadata defines grouping element which should be used to group columns together in any GUI system that is designed in order to satisfy the display requirements of an MLS.
The Column Group metadata starts with a <METADATA-COLUMN_GROUP> tag with Resource, Class, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-25, followed by the <DATA> section, which contains the actual field information. The Column Group metadata has the following format:
<METADATA-COLUMN_GROUP SP **Resource="***resource-id***"** SP **Class="***class-id***"** SP **Version="***column-group-version***"** SP **Date="***column-group-date***">**
<COLUMNS> *column-group-field* ***(***column-group-field***)</COLUMNS>**
(<DATA>** *column-group-data* ***(** *column-group-data)</DATA>)**
</METADATA-COLUMN_GROUP>
*resource-id ::= 1*32ALPHANUM*This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.
*class-id ::= 1*32ALPHANUM*This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group applies.
*column-group-version ::= 1*2DIGITS* **.** *1*2DIGITS . 1*5DIGITS*This is the version number of this Column Group metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. Every time this Column Group metadata changes the version number should be increased.
*column-group-date ::= DATE*The latest change date of this Column Group metadata.
column-group-field ::= <Field Name from Table 11-25>{_}column-group-data ::= <valid value as defined in Table 11-25>_An example Table section follows:
GetMetadata request:
Type: METADATA-COLUMN_GROUP
ID Format: Resource : Class
ID Example: Property : RES
Compact reply:
<METADATA-COLUMN_GROUP Resource="Property" Class="RES" Version="1.00.000"
Date= "Sat, 20 Mar 2002 12:03:38 GMT" >
<COLUMNS>MetadataEntryIdColumnGroupNameControlSystemName
LongNameShortNameDescriptionVersionDate</COLUMNS>
<DATA>10001123456WaterFrontWaterFrontFlagWater Front InformationWater FrontListing data that contains water front information for the Listing1.00.000Thu, 3 Feb 2005 20:35:15 GMT</DATA>
</METADATA-COLUMN_GROUP>
Table 11-25 Metadata Content – Column Group

Metadata Field	Content Type	Description
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged. In particular, it should be managed so as to allow the client to detect changes to the ColumnGroupName.
ColumnGroupName	1*32ALPHANUM	The name that uniquely identifies this Column Group within the Class.
ControlSystemName	1*32ALPHANUM	The SystemName of the Table Metadata that identifies the data element that is used to control the display of this Column Group.
LongName	1*64ALPHANUM	The name of the Column Group as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this Column Group when it appears on a report.
ShortName	1*32ALPHANUM	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined; it is expected that clients will use this field in human-interface elements such as picklists.
Description	1*256ALPHANUM	A brief description of the purpose for this Column Group.
Version	1*2DIGIT . 1*2DIGIT . 1*5DIGIT	The latest version of the Column Group metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. The version number is advisory only.
Date	DATE	The date on which any of the Column Group metadata child elements were last changed. Clients MAY rely on this date for cache management.

METADATA-COLUMN_GROUP_CONTROL

This metadata defines the valid ranges of values that the specified SystemName may have that control the display of the Column Group. If the SystemName contains any of the values that fall within the ranges specified in this table, the Column Group may be displayed. If it does not, the

Column Group should not be displayed. The data is returned as a list of high and low values that determine whether the Column Group should be displayed.
The Column Group Control metadata starts with a <METADATA-COLUMN_GROUP_CONTROL> tag with Resource, Class, ColumnGroup, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-26, followed by the <DATA> section, which contains the actual field information. The Column Group Control metadata has the following format:
<METADATA-COLUMN_GROUP_CONTROL SP **Resource="***resource-id***"** SP **Class="***class-id***"** SP **ColumnGroup="*column-group-id**"* SP **Version="***column-group-control-version***"** SP **Date="***column-group-control-date***">**
<COLUMNS> *column-group-control-field* ***(****column-group-control-field***)</COLUMNS>**
(<DATA>** *column-group-control-data* ***(** *column-group-control-data)</DATA>)**
</METADATA-COLUMN_GROUP_CONTROL>
*resource-id ::= 1*32ALPHANUM*This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.
*class-id ::= 1*32ALPHANUM*This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group Control applies.
*column-group-id ::= 1*32ALPHANUM*This value MUST be a ColumnGroupName found in the ColumnGroup metadata for this Class. It is the ColumnGroup for which the control applies.
*column-group-control-version ::= 1*2DIGITS* **.** *1*2DIGITS . 1*5DIGITS*This is the version number of this Column Group Control metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. Every time this Column Group Control metadata changes the version number should be increased.
*column-group-control-date ::= DATE*The latest change date of this Column Group Control metadata.
column-group-control-field ::= <Field Name from Table 11-26>{_}column-group-control-data ::= <valid value as defined in Table 11-26>_An example Table section follows:
GetMetadata request:
Type: METADATA-COLUMN_GROUP_CONTROL
ID Format: Resource : Class : ColumnGroup
ID Example: Property : RES : WaterFront
Compact reply:
<METADATA-COLUMN_GROUP_CONTROL Resource="Property" Class="RES" ColumnGroup="WaterFront" Version="1.00.000" Date= "Sat, 20 Mar 2002 12:03:38 GMT" >
<COLUMNS>MetadataEntryIdLowValueHighValue</COLUMNS>
<DATA>1001112345611</DATA>
</METADATA-COLUMN_GROUP_CONTROL>
Table 11-26 Metadata Content – Column Group Control

Metadata Field	Content Type	Description
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged. In particular, it should be managed so as to allow the client to detect changes to an individual pair of High/Low values.
LowValue	1*64ALPHANUM	The minimum value that the ControlSystemName field of the ColumnGroup is allowed to have in order to display the ColumnGroup. It is expected that the actual data type returned is interpreted as per the data type of the ControlSystemName of the ColumnGroup.
HighValue	1*64ALPHANUM	The maximum value that the ControlSystemName field of the ColumnGroup is allowed to have in order to display the ColumnGroup. It is expected that the actual data type returned is interpreted as per the data type of the ControlSystemName of the ColumnGroup. If the restricting data is not a range, then HighValue may be left blank.

METADATA-COLUMN_GROUP_TABLE

This metadata defines the set of SystemNames that are to be displayed within a Column and the order in which they are to be displayed.
The Column Group Table metadata starts with a <METADATA-COLUMN_GROUP_TABLE> tag with Resource, Class, ColumnGroup, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-27, followed by the <DATA> section, which contains the actual field information. The Column Group Table metadata has the following format:
<METADATA-COLUMN_GROUP_TABLE SP **Resource="***resource-id***"** SP **Class="***class-id***"** SP **ColumnGroup="*column-group-id**"* SP **Version="***column-group-table-version***"** SP **Date="***column-group-table-date***">**
<COLUMNS> *column-group-table-field* ***(***column-group-table-field***)</COLUMNS>**
(<DATA>** *column-group-table-data* ***(** *column-group-table-data)</DATA>)**
</METADATA-COLUMN_GROUP_TABLE>
*resource-id ::= 1*32ALPHANUM*This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.
*class-id ::= 1*32ALPHANUM*This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group applies.
*column-group-id ::= 1*32ALPHANUM*This value MUST be a ColumnGroupName found in the ColumnGroup metadata for this Class. It is the ColumnGroup for which the table applies.
*column-group-table-version ::= 1*2DIGITS* **.** *1*2DIGITS . 1*5DIGITS*This is the version number of this Column Group Table metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. Every time this Column Group Table metadata changes the version number should be increased.
*column-group-table-date ::= DATE*The latest change date of this Column Group Table metadata.
column-group-table-field ::= <Field Name from Table 11-27>{_}column-group-table-data ::= <valid value as defined in Table 11-27>_An example Table section follows:
GetMetadata request:
Type: METADATA-COLUMN_GROUP_TABLE
ID Format: Resource : Class : ColumnGroup

ID Example: Property : RES : WaterFront
Compact reply:
<METADATA-COLUMN_GROUP_TABLE Resource="Property" Class="RES" ColumnGroup="WaterFront" Version="1.00.000" Date= "Sat, 20 Mar 2002 12:03:38 GMT" >
<COLUMNS>MetadataEntryIdSystemNameDisplayOrder</COLUMNS>
<DATA>10111123450WaterFront1</DATA>
<DATA>10111123451WaterAccess2</DATA>
<DATA>10111123452WaterFrontage3</DATA>
<DATA>10111123453WaterView4</DATA>
</METADATA-COLUMN_GROUP_TABLE>

Table 11-27 Metadata Content – Column Group Table

Metadata Field	**Content Type**	**Description**
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged.
SystemName	1*32ALPHANUM	The SystemName of the field that is to be displayed in the ColumnGroup. This MUST be a valid SystemName for this Class. A SystemName MUST be unique within the ColumnGroup. This MUST not be specified if a ColumnGroupSetName is specified.
ColumnGroupSetName	1*32ALPHANUM	The name of a ColumnGroupSet to display in place of a single field. It is expected that this is a ColumnGroupSet that does not display a large number of columns. This MUST not be specified if a SystemName is specified. The ColumnGroupSet MUST not contain a ColumGroup that also specifies a ColumnGrupoSetName in the COLUMN_GROUP_TABLE metadata.Mark will try it out on presentation styles other than Matrix and will get back to the group on how easy it is to implement this.
LongName	1*64ALPHANUM	The name of the Column Group Table (data field) as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this Column Group when it appears on a report.
ShortName	1*32ALPHANUM	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined; it is expected that clients will use this field in human-interface elements such as picklists.
DisplayOrder	1*5DIGIT	The order within the ColumnGroup that this SystemName is to be displayed in. DisplayOrder values MAY contain gaps and may have the same value as other columns. If multiple columns have the same value, the client SHOULD display the columns in Alphabetical order.
DisplayLength	1*5DIGIT	The number of characters to allow when displaying data for this column.
DisplayHeight	1*5DIGIT	The number of rows to display the data in. A value greater than one in this column implies a multi-line data entry field of DisplayLength width. If users enter data into this field that is longer than will fit within this text box, it is expected that the field will scroll to allow further data entry.
ImmediateRefresh	Boolean	A truth value which indicates whether a change to this field by the user should cause an automatic GUI refresh. This is primarily intended for use

METADATA-COLUMN_GROUP_NORMALIZATION

This metadata defines a grid that can be used by a client to display related fields in a manner more appropriate for data entry.
The Column Group Renormalization metadata starts with a <METADATA-COLUMN_GROUP_NORMALIZATION> tag with Resource, Class, ColumnGroup, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-28, followed by the <DATA> section, which contains the actual field information. The Column Group Normalization metadata has the following format:
<METADATA-COLUMN_GROUP_NORMALIZATION SP **Resource="***resource-id***"** SP **Class="***class-id***"** SP **ColumnGroup="*column-group-id** "* SP **Version="***column-group-normalization-version***"** SP **Date="***column-group-normalization-date***">**
<COLUMNS> *column-group-normalization-field* *****(***column-group-normalization-field***)</COLUMNS>**
*****(<DATA>** *column-group-normalization-data* *****(** *column-group-normalization-data***)</DATA>)**
</METADATA-COLUMN_GROUP_NORMALIZATION>
*resource-id ::= 1*32ALPHANUM*This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.
*class-id ::= 1*32ALPHANUM*This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group Normalization applies.
*column-group-id ::= 1*32ALPHANUM*This value MUST be a ColumnGroupName found in the ColumnGroup metadata for this Class. It is the ColumnGroup for which the grid applies.
*column-group-normalization-version ::= 1*2DIGITS* **.** *1*2DIGITS . 1*5DIGITS*This is the version number of this Column Group Normalization metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. Every time this Column Group Normalization metadata changes the version number should be increased.
*column-group-normalization-date ::= DATE*The latest change date of this Column Group Normalization metadata.
column-group-normalization-field ::= <Field Name from Table 11-28>{_}column-group-normalization-data ::= <valid value as defined in Table 11-28>_An example Table section follows:
GetMetadata request:
Type: METADATA-COLUMN_GROUP_NORMALIZATION
ID Format: Resource : Class : ColumnGroup
ID Example: Property : RES : WaterFront
Compact reply:
<METADATA-COLUMN_GROUP_NORMALIZATION Resource="Property" Class="RES" ColumnGroup="WaterFront" Version="1.00.000" Date=

```
"Sat, 20 Mar 2002 12:03:38 GMT" >
<COLUMNS>MetadataEntryIdTypeIdentifierSequenceColumnLabelSystemName</COLUMNS>
<DATA>10211123450Bedroom1LengthBedroom1Length</DATA>
<DATA>10211123451Bedroom1WidthBedroom1Width</DATA>
<DATA>10211123452Bedroom1AreaBedroom1Area</DATA>
<DATA>10211123453Bedroom2LengthBedroom2Length</DATA>
<DATA>10211123454Bedroom2WidthBedroom2Width</DATA>
<DATA>10211123455Bedroom2AreaBedroom2Area</DATA>
<DATA>10211123456Bedroom3LengthBedroom3Length</DATA>
<DATA>10211123457Bedroom3WidthBedroom3Width</DATA>
<DATA>10211123458Bedroom3AreaBedroom3Area</DATA>
<DATA>10211123459LivingRoomLengthLivingRoomLength</DATA>
<DATA>10211123460LivingRoomWidthLivingRoomWidth</DATA>
<DATA>10211123461LivingRoomAreaLivingRoomArea</DATA>
<DATA>10211123462DiningRoomLengthDiningRoomLength</DATA>
<DATA>10211123463DiningRoomWidthDiningRoomWidth</DATA>
<DATA>10211123464DiningRoomAreaDiningRoomArea</DATA>
<DATA>10211123465KitchenLengthKitchenLength</DATA>
<DATA>10211123466KitchenWidthKitchenWidth</DATA>
<DATA>10211123467KitchenAreaKitchenArea</DATA>
</METADATA-COLUMN_GROUP_NORMALIZATION>
```

Example Screen Display based on Above Compact Reply:

Type	Sequence	Length	Width	Area
Bedroom	**1**	**Bedroom1Length**	**Bedroom1Width**	**Bedroom1Area**
Bedroom	**2**	**Bedroom2Length**	**Bedroom2Width**	**Bedroom2Area**
Bedroom	**3**	**Bedroom3Length**	**Bedroom3Width**	**Bedroom3Area**
LivingRoom		**LivingRoomLength**	**LivingRoomWidth**	**LivingRoomArea**
DiningRoom		**DiningRoomLength**	**DiningRoomWidth**	**DiningRoomArea**
Kitchen		**KitchenLength**	**KitchenWidth**	**KitchenArea**

The user's data entry area would be the greyed out section and the data that they enter would be assigned to the SystemName in that grid position.

Table 11-28 Metadata Content – Column Group Normalization

Metadata Field	Content Type	Description
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged.
TypeIdentifier	1*32ALPHANUM	Y Axis – Row Label – The Label that is to be displayed on the left side of the screen that identifies the Type of data that the user is entering.
Sequence	1*5DIGIT	Y Axis – Row Sequence – The Sequence number that is to be displayed on the left side of the screen after the TypeIdentifier. This itemizes the Type of data that the user is entering.
ColumnLabel	1*32ALPHANUM	X Axis – Column Label – This is the label that is to appear at the top of the screen for data within this column. It is expected that all data in this grid with the same ColumnLabel be displayed in the same column on the screen.
SystemName	1*32ALPHANUM	The SystemName of the field that is to be displayed in this position in the Grid for the ColumnGroup. This MUST be a valid SystemName for this Class and MUST be within the ColumnGroupTable of the ColumnGroup. Fields that appear within a ColumnGroup, but not within the Normalization for the ColumnGroup, are to be treated as separate data entry fields that are not part of the grid. The SystemName MUST be unique within the ColumnGroup.

4. Development Impact

5. Compatibility

RCP 61 - Validation Expression Replacement

Original document: RCP 61 Validation Expression Replacement

Note: This RCP Affects the Following Sections:

- Section 11.3.2 Table
- Section 11.4.9.1 Validation Expression Types and Data Types
- Section 11.4.9.2 Validation Expression Special Operand Tokens
- Section 11.4.9.3 Validation Expression Functions and Operators
- Section 11.4.9.4 Validation Expression BNF Representation

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Sergio Del Rio
Organization	Templates 4 Business, Inc.
E-mail	Sergio dot Del dot Rio at t4bi dot com
Author	Libor Viktorin
Organization	MarketLinx
Email	lviktorin at marketlinx dot com
Submitted Date	August 7, 2005
Revised Date	October 30, 2007
RETS Version	1.8.0
Status	Adopted

1. Synopsis

This proposal replaces the current validation expression description with a formalized BNF that is machine readable.

2. Rationale

Currently the Validation Expression section does not take into a account a number of factors such as data type and format that would provide a consistent interpretation between RETS Servers. This proposal corrects this by defining the validation expression elements required to successfully determine the validity of a data element against the business rules of the system.

3. Proposal

3.1 Section 11.3.2 – Additional field to Table Metadata

Following row should be added to table 11-36 (Validation Expression Metadata Content):

Message	*TEXT	Message to be shown to the user if a field is rejected, or a warning is issued, because of this validation expression
IsCaseSensitive	BOOL	When true, the string comparisons in the expressions are case sensitive.

Section 11.4.9 should be rewritten as follows:

This section describes the ValidationExpression table that is referenced in Section 11.3.4. There MUST be a corresponding table entry for each ValidationExpressionID referenced in the METADATA-UPDATE_TYPEs for a Resource.

The table contains expressions that are to be evaluated when a field value is entered by the user. Expressions in the list MUST be evaluated in the order in which they appear in the list.

Note that the key for an update field (see table 11-11) has a specific role. If there is a validation expression associated with that field, it must be evaluated even if the field itself is not part of the update request (in ADD update). If the expression for this field evaluates as REJECT, the whole record is rejected and the client SHOULD NOT send the Update request.

There are several types of validation expressions, each introduced by a keyword preceding the expression:

Keyword Expression type	Value Type	Purpose
ACCEPT	Boolean	If the expression is true, the field value is considered accepted without further testing. Immediately following SET expressions MUST be executed. If the expression is false, following validation expressions MUST be executed. If the expression is ERROR (evaluation failed) in client, the client SHOULD act as if the field was accepted, allowing the server to make the final decision.
REJECT	Boolean	If the expression is true, the field value is considered rejected without further testing. Subsequent SET expressions MUST NOT be evaluated. If the expression is false, following validation expressions MUST be executed. If the expression is ERROR, evaluation failed in client, the client SHOULD act as if the field was accepted, allowing the server to make the final decision.
WARNING	Boolean	If the expression is true, the client should show a warning message to the user, and if the warning is OK-ed by the user, include a Warning-Response in the UPDATE request. If the user does not OK the warning, the field is considered rejected and following SET validation expressions MUST NOT be evaluated. If the expression is false, following validation expressions MUST be evaluated.

SET	Assignment	The expression MUST begin with a field name and an equal sign. The following expression is evaluated and the result stored in the designated field.
SET_DEFAULT	Appropriate Data Type of Assigned field.	This expression MUST be executed ONLY when a NEW record is created. Supersedes the default value as indicated in the Update Metadata.
SET_REQUIRED	Boolean	Expressions of this type are designed to evaluate an expression and set the field the rule is being executed on to Required if the expression returns true and to Non Required if the expression returns false.
SET_READ_ONLY	Boolean	Expressions of this type are designed to evaluate an expression and set the field the rule is being executed on to Read Only if the expression returns true and to Updateable if the expression returns false.The client application is expected to lock the value of the field the rule is being executed on to the value at the time the SET_REQUIRED expression is evaluated.
RESTRICT_PICKLIST	List of CHARs	Expressions of this type are designed to return one or more LOOKUP values that are to be removed from the LOOKUP list that is defined for the field the rule is being executed on. This is always the entire set of values to remove from the lookup. In other words, if this returns a blank list or .EMPTY., the entire set of LOOKUP values is to be displayed. The value of this expression MUST be a <List>, rather than <Exp>, as defined in 11.4.9.1. All members of the list MUST be of the same type as the type of the field the rule is being executed on.
SET_PICKLIST	List of CHARs	Expressions of this type are designed to return one or more LOOKUP values that are to be used in the LOOKUP list that is defined for the field the rule is being executed on. The value of this expression MUST be a <List>, rather than <Exp>, as defined in 11.4.9.1. Every member of the list MUST exist in the Lookup list as defined in the metadata for the field the rule is being executed on.
SET_DISPLAY	Boolean	Expressions of this type are designed to allow a client to make fields visible or invisible based on the evaluation of an expression. The result of this expression has no effect on whether a field is READ ONLY or not.

The Server MAY expose expressions with other keywords; the keyword of such an expression MUST begin with a prefix “X-”. Clients MUST ignore any expression with a keyword they do not recognize.

If the field is not rejected after processing all validation expression, a client MUST consider it accepted.

Every validation expression has one of following types:

Table 3-x Validation Expression Data Types

CHAR	any string of ASCII characters
INT	any integer (tiny, small, int, long)
FLOAT	decimal number with fraction part
TIME	time, date, datetime
BOOLEAN	true or false
LIST	list of several values of the same type. The type of the values may be any of those above.
EMPTY	missing data (similar to NULL in database systems)
ERROR	this is the type of an expression which cannot be parsed

11.4.9.x Validation Expression BNF

```
<Exp>  ::= <OrExp>
<List> ::= <LParen> <RParen>
| <LParen> Exp \*( , <Exp> ) <RParen>\\
 <OrExp>  ::= <AndExp> \*( .OR. <AndExp> )\\
 <AndExp>  ::= <NotExp> \*( .AND. <NotExp> )\\
 <NotExp>  ::= .NOT. <NotExp> | <EqExp>\\
 <EqExp>  ::= <CmpExp> |
| <CmpExp> = <CmpExp> |
| <CmpExp> \!= <CmpExp>\\
 <CmpExp> ::= <CntExp> |
| <CntExp> <= <CntExp> |
| <CntExp> >= <CntExp> |
| <CntExp> < <CntExp> |
| <CntExp> > <CntExp> \\
 <CntExp> ::= <SumExp> |
| <SumExp> .CONTAINS. <SumExp> |
| <SumExp> .IN. <List>\\
 <SumExp> ::= <ProdExp> \*( ( + | -  | <Concat> ) <ProdExp> )\\
 <ProdExp> ::= <AtomExp> \*( ( * | / | .MOD. ) <AtomExp> )\\
 <AtomExp> ::= <LParen> <Exp> <RParen> |
| <Value> |
| <FuncExp>\\
 <FuncExp> ::= <Func> <LParen> <Param> \*( , <Param> ) <RParen>\\
 <Func ::= ALPHA \*( ALPHANUM )\\
 <Param> ::= <Exp>\\
 <Value> ::= <SpecValue> |
| <CharValue> |
| <IntValue> |
| <FloatValue> |
| <TimeValue> |
| <TimeSpanValue> |
| <FieldName>\\
 <Concat> ::= |
<FieldName> ::=  RETSNAME  \|  \[ RETSNAME \]
<SpecValue> := . RETSNAME .
<CharValue> ::= ' PLAINTEXT ' \| " PLAINTEXT "
<TimeValue> ::= # DATE #
<IntValue> ::= 0*1(\+ \| \-) 1*(DIGITS)
<FloatValue>::= <IntValue>  . \*(DIGIT)
<LParen> ::= (
<RParen> ::= )
```

The value of a Validation Expression MUST conform to the <Exp> syntax in the grammar above, except for RESTRICT_PICKLIST and SET_PICKLIST expressions, whose value MUST conform to the <List> syntax. Any expression with keyword starting with “X-” MAY have a <List> value as well.

The text in CharValue must not include the (single or double) quote used to delimit the value.

TimeValue must be expressed in the ISO8601 format and enclosed in hashmarks(#) (ex. #2007-09-11T14:30:00#).

A <FieldName> is a name of a field belonging to the same class as the field to which this expression is attached, and has a type of that field specified by the metadata. If used in brackets, its value is the value of the field as it was in the database before the current updates took place. If used without brackets, the updated value of the field MUST be used.

A <TimeValue> has TIME type.

A <CharValue> has CHAR type.

A <IntValue> has INT type.

A <FloatValue> has FLOAT type.

A <SpecValue> may be one of these values:

Table Special Token Names

Token Name	Data Type	Description
.TRUE.	BOOLEAN	Boolean value of TRUE (1)
.FALSE.	BOOLEAN	Boolean value of FALSE (0)
.EMPTY.	EMPTY	A value that matches an empty or all-blank field. Supplies an empty (zero-length) field when used in a SET expression.
.TODAY.	TIME	The current date.
.NOW.	TIME	The current time.
.ENTRY.	type of the current field	The current field text, as a string.
.OLDVALUE.	type of the current field	The text that was in the field as returned from the host in the search operation. If the field is new, .OLDVALUE. is an EMPTY value.
.USERID.	CHAR	The value of the user-id field returned in the Login transaction, unless an info-token-key named USERID has been returned in the Login transaction.
.USERCLASS.	CHAR	The value of the user-class field returned in the Login transaction, unless an info-token-key named USERCLASS has been returned in the Login transaction.
.USERLEVEL.	CHAR	The value of the user-level field returned in the Login transaction, unless an info-token-key named USERLEVEL has been returned in the Login transaction.
.AGENTCODE.	CHAR	The value of the agent-code field returned in the Login transaction, unless an info-token-key named AGENTCODE has been returned in the Login transaction.
.BROKERCODE.	CHAR	The value of the broker-code field returned in the Login transaction, unless an info-token-key named BROKERCODE has been returned in the Login transaction.
.BROKERBRANCH.	CHAR	The value of the broker-branch field returned in the Login transaction, unless an info-token-key named BROKERBRANCH has been returned in the Login transaction.
.UPDATEACTION.	CHAR	Name of the UpdateAction for which this validation is performed.
.any.	(see Description)	If the name of the SpecValue (stripped of the first and last dot) is equal to a name of one of the info-token-keys returned as part of the Login response, then the type and value of this SpecValue is defined by that info-token-key. If no such info-token-key exists, the value is ERROR.

A <FuncExp> is a function with parameters. Following functions are defined:

Function	parameter types	type
BOOL	BOOLEAN or CHAR	BOOLEAN
CHAR	Any, except for FLOAT	CHAR
CHARF	FLOAT, INT	CHAR
TIME	TIME or CHAR	TIME
DATE	TIME or CHAR	TIME
INT	INT or FLOAT or BOOL or CHAR	INT
FLOAT	INT or FLOAT or BOOL or CHAR	FLOAT
SUBSTR	CHAR,INT,INT	CHAR
STRLEN	CHAR	INT
LOWER	CHAR	CHAR
UPPER	CHAR	CHAR
IIF	BOOLEAN,any,any	any
YEAR	TIME	INT
MONTH	TIME	INT

DAY	TIME	INT
WEEKDAY	TIME	INT

The BOOL, CHAR, TIME, DATE, INT and FLOAT functions are used just to change a type of expression. The DATE and TIME functions are synonyms. Note that any of these functions may fail (return an ERROR value) if the parameter can not be transformed to the appropriate type.

In conversion from BOOLEAN to INT or FLOAT, .TRUE. is converted to 1 and .FALSE. is converted to 0. Casting FLOAT to INTEGER discards the fractional part.

When converting to CHAR, BOOL values are represented as “0” and “1”, TIME values are represented using format defined in RFC 1123 with digital timezone, INT values are represented with no leading zeroes.

When converting from CHAR to BOOL, values “0”,”1”,”YES”,”NO”,”TRUE” and “FALSE” (no matter what the case) MUST be understood.

When converting from CHAR to TIME, any RFC 1123 --compliant format MUST be understood. A leading and/or trailing # MUST be removed before conversion.

When converting from CHAR to INT or FLOAT, usual formats MUST be understood. Scientific format (with exponent) MUST NOT be understood. FLOAT numbers with empty integral part (.5, -.4) MUST be understood as long as there is at least one digit after the decimal point.

The CHARF function converts a Float number, and in the second parameter specifies how many decimal digits MUST appear after the point.

The SUBSTR function returns a substring of its first parameter. Second parameter is a starting position of the substring, third parameter is the ending position of the substring. Positions are 1-based.

The STRLEN function returns the length if its parameter.

The Upper function returns its parameter upper-cased.

The IIF function return the value of its second parameter if the first parameter evaluates to true, or the value of its third parameter otherwise. Types of second and third parameter must be same, and it is the type of the result.

The YEAR,MONTH,DAY and WEEKDAY parse the date part of TIME value. They return values ranging from 1 to the appropriate maximum. WEEKDAY returns 1 for Sunday, 2 for Monday etc.

Other functions may be defined later (HOUR and MINUTE are first candidates). If a server uses a function the client does not recognize, the client MUST evaluate it as ERROR.

The operators may be applied on certain types, and the resulting type is defined by the following table:
Operators

Operator	Left operand	Right operand	Result	Meaning
.MOD.	INT	INT	INT	Arithmetic MODULO operation
/,*	INT	INT	INT	Integral division and multiplication
/,*	INT	FLOAT	FLOAT	Division and multiplication
/,*	FLOAT	INT	FLOAT	Division and multiplication
/,*	FLOAT	FLOAT	FLOAT	Division and multiplication
+	INT	INT	INT	Addition
+	INT	FLOAT	FLOAT	Addition
+	FLOAT	INT	FLOAT	Addition
+	FLOAT	FLOAT	FLOAT	Addition
+	FLOAT	TIME	TIME	Time shift
+	TIME	FLOAT	TIME	Time shift
||	CHAR	CHAR	CHAR	String concatenation
-	INT	INT	INT	Subtraction
-	INT	FLOAT	FLOAT	Subtraction
-	FLOAT	INT	FLOAT	Subtraction
-	FLOAT	FLOAT	FLOAT	Subtraction
-	TIME	FLOAT	TIME	Time shift
-	TIME	TIME	FLOAT	Time shift
.CONTAINS.	CHAR	CHAR	BOOLEAN	String containment (true if right operand is a substring in left operand)
.IN.	Any	List of operands, all of the same type as left	BOOLEAN	List inclusion (true if left is equal to any member of the list)
=,!=	Any	Same as left	BOOLEAN	Equality
.AND.,.OR.	BOOLEAN	BOOLEAN	BOOLEAN	Boolean operations

.NOT.	BOOLEAN	BOOLEAN	BOOLEAN	Boolean negation

Arithmetic operations between dates use number of days as the FLOAT parameter (or result); e.g. 0.25 represents a time span of 6 hours.

String operations are case sensitive or not, based on the IsCaseSensitive field in the expression's metadata.

Any operation with an ERROR argument MUST evaluate to ERROR. An EMPTY value may be compared (=,!=) against any value.

Appropriate casting functions (BOOL, CHAR, TIME, INT, FLOAT) MUST be applied to the parameters. If a function or an operator is applied to a datatype different than shown in the above tables, the expression MUST evaluate to ERROR.

The using of strong types will help in optimizing and preprocessing the validation expressions, as well as it specifies more precisely how to calculate the expression values. However, it forced the use of conversion functions, so I added some basic other functions as well. More functions may be added to the specifications if the need shows.

The rest of 11.4.9 (Validation Expression Metadata), following table 11-34, should remain unchanged (except for the new row in table 11-36, see top of 3.1 in this proposal).

4. Development Impact

Because the proposal changes the validation expressions as they are currently defined it will have a big impact on current client and server software not currently following this format.

5. Compatibility

This is not backward compatible.

RCP 63 - Object Data and Upload

Original document: RCP 63 Object Data and Upload

Note: This RCP Affects the Following Sections:
- Section 5.3 Required Request Arguments
- Section 5.4 Optional Request Arguments
- Section 5.6 Optional Server Response Header Fields
- Section 5.12 ObjectData Classes
- Section 11.4.1 Object
- Section 13 - PostObject Transaction

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Libor Viktorin
Organization	MarketLinx
E-mail	lviktorin at marketlinx dot com
Submitted Date	July 20, 2007
Originating Workgroup	
RETS Version	1.8.0
Status	Adopted

1. Synopsis

This proposal adds a missing functionality to the RETS specifications. It specifies a method to upload binary objects, such as images, to the server.
It also addresses the need to store and expose detailed information about these objects, and makes it easier to search for them.

This is a second updated version of "RETS Change Proposal: Object Upload" filed on August 8, 2003.

2. Rationale

The RETS currently does not specify a way to upload objects to the server. This proposal adds a PostObject transaction, which complements the GetObject transaction already specified. It also adds a way of exposing additional data about objects such as descriptions, captions, image sizes, modification dates, etc. With this additional info, a client can search for objects matching some criteria, rather than requesting objects only by their order number.

3. Proposal

3.1 Specification Changes & Additions 3.1.1 ObjectData field in Object metadata

The following line should be added to the Table 11-16 (Object Metadata Content). Adequate change should be done to the METADATA.dtd file.

Field Name	Content Type	Description
PostSupport	0 1	PostObject transaction is unavailable for this object type PostObject transaction is available for this object type
ObjectData	RETSNAME:RETSNAME	Resource and Class of a table that provides additional data about objects described by this metadata. If an object contains no additional data, this field MUST be empty or missing.
MaxFileSize	Numeric	Indicates that maximum file size (in bytes) that is accepted by the server. The server MAY refuse any objects sent that are greater than this size. The server MAY return an http error code if an object bigger than this is received.

Object Data classes

The following should be added to the specs as chapter 5.12. The current chapter 5.12 (Error codes) should be renumbered to 5.13, and its table 5-1 should be renumbered to 5-2.

5.12 ObjectData classes

The server MAY expose additional data for object files. If it does, such a data MUST be exposed in a class linked to an object type via the ObjectData metadata field in the Object metadata. Any table linked this way to the Objects MUST hold exactly one record for each object file available through the GetObject transaction on the linked Resource and ObjectType. The data must always correctly describe the object (eg. if a FileSize field is exposed, its value MUST be the length of a file sent by the GetObject transaction).

A good practice would be for the server to expose all such tables within one resource named OBJECT; however, the server is free to use any resource and class name as it sees fit.

Any class linked to an Object metadata item MAY expose any data about the object; however, if the data is one of the fields listed in table 5-1, the class MUST use the standard names from that table. Any class MUST expose fields shown in **bold**.

Data from the ObjectData table will be communicated via HTTP headers (see 5.4.2 and X 1.2).

Table 5-1

Standard Name	DataType	Description
UID	CHARACTER	Unique ID. This field must be unique within the class and its resource.
ObjectType	CHARACTER	ObjectType of this object. This is the Type parameter in the GetObject request.
ResourceName	CHARACTER	Standard name of the resource this object belongs to. This is the Resource parameter in the GetObject request.
ResourceID	CHARACTER	Value of the Key Field identifying a record within the resource described by ResourceName. This is the first part of the ID parameter in the GetObject request.
ObjectID	INT	Ordinal number of this object within all objects belonging to the record identified by ResourceName and ResourceID. This is the ObjectID in the GetObject request.
MimeType	CHARACTER	MimeType of the object
IsDefault	BOOL	1 if this object is the default one (sent when an object with ObjectId= 0 was requested). This is the main object that should be displayed for the ObjectType.
ObjectModificationTimestamp	DATETIME	Time of the last modification of the object
ModificationTimestamp	DATETIME	Time of the last modification of this data record (including the object modification)
OrderHint	INT	Provides an override for the ObjectId value so a client can specify an alternate ordering of ObjectData while preserving the one specified by the ObjectIds. This allows clients performing updates using the UID field perform operations such as resource reordering more easily. Unlike the ObjectId, where ordinal values are defined by starting with the number 1 and using the formula n+1 to arrive at each consecutive term in the sequence, the OrderHint values can form any integer set provided that sequence of items within the set is preserved. If A and B are two objects in a collection then the following expression must be true: If ObjectA.OrderHint > ObjectB.OrderHint then ObjectA.ObjectId > ObjectB.ObjectId If the OrderHint is supported, then the server must maintain the number as sent by the client.
Description	CHARACTER	Description of the object.
Caption	CHARACTER	Short title of the object.
FileSize	INT	Size, in bytes, of the object
WidthPix	INT	Width of an image, in pixels
HeightPix	INT	Height of an image, in pixels
Duration	INT	Length of a movie or audio, in seconds
WidthInch	INT	Width of an image, in inches
HeightInch	INT	Height of an image, in inches
…		Other well-known fields will be defined later

Searching for objects: Any ObjectData class is searchable as any other class. The object data may be searched by any searchable field specified in the class. The ResourceName, ResourceID and Order or UID of the matching results may be used as parameters in a subsequent GetObject transaction requesting the matching objects.

Updating object data: The server MAY expose updates for the ObjectData class. However, the data MUST stay consistent with the object files, so eg. the system must not allow changing the WidthPix field unless it is able to crop or resize the underlying image file.

The system MUST NOT expose an Add update for the ObjectData class, since objects will be added via the PostObject transaction. The system MAY expose a Delete update, which will have a similar effect as the PostObject transaction with the Update=Delete (namely, it will remove a record from the Object table AND delete the object file). The system MAY also expose a CLONE transaction to allow for reusing an existing object in more than one record.

The system MAY expose an UPLOAD update. Clients SHOULD NOT use this update directly; if they do, server MUST refuse it. This update type is used to hook up validation expressions and update help to the PostObject transaction. If the UPLOAD update exists, both client and server SHOULD use it to check data sent via the PostObject transaction. The UPLOAD update's validation expressions may place constraints on any field defined in the Object table, thus specifying acceptable file characteristics.

Any document listing Standard Names should be updated to reflect the names in Table 5-1.

3.1.3 GetObject transaction modification

The following section should be added to the end of RETS specification, chapter 5.3 (GetObject Required Request Arguments):

UID The string identifying the object(s) being requested:

UID	::= TOKEN *("," TOKEN)

The UID argument allows for requesting objects by their UIDs, as described in table 5-1. Either ID or UID argument MUST be present in the request, but not both of them. If the server does not support UIDs for the requested type of objects, it MUST respond with an error (Preferred code would be 20403: No object found. Servers that do not implement the functionality proposed in this document may respond with 20402: Invalid Identifier). However, if the requested type of objects has an ObjectData class linked in its metadata, the server MUST support this argument.

The following section should be added to the RETS specifications, chapter 5.4 (GetObject Optional Request Arguments):

5.4.2 ObjectData

ObjectData	*| FieldName *(, FieldName)

FieldName	::= RETSNAME

This parameter indicates that data relevant to the object should be sent as HTTP headers in the server response . If ObjectData is set to "*", the server MUST include a header line for each field in the ObjectData class linked to the requested Object Type, as described in chapter 5.6. If ObjectData is set to a list of fields, ObjectData headers for the requested fields only MUST be sent in the response. If this argument is missing, no ObjectData will be sent.

The following section should be added to the RETS specification, chapter 5.6 (GetObject Optional Server Response Header Fields)

UID

The UID of the object. This field is required and MUST be returned if the request used UID rather than ID argument, and MAY be sent if ID has been used.

ObjectData

If the client has submitted a request with "ObjectData" the header of the response MUST contain the ObjectData header field(s). The server MUST include a header line for each requested field in the ObjectData class linked to the requested Object Type. Each such header will have the name of "ObjectData", and its value will be the SystemName of the field, followed by an equals sign, followed by the value of the field.

Example:

```
ObjectData: PropMediaCaption=caption for kitchen
ObjectData: PropMediaDescription=details about kitchen
```

3.1.4 PostObject transaction

The following section should be added to the RETS specifications as a special chapter (replace X with the chapter number)

Section X: PostObject Transaction

The PostObject transaction is used to upload structured information related to known system entities. This transaction allows the client to send one file as a MIME type along with more information. The server's response is similar to that of the Update transaction.

If the server supports the PostObject transaction, it sets the PostSupport field in the Object metadata to 1.

Before a client tries to upload an object, it SHOULD check for the presence of the ObjectData class in the metadata. If such a class exists, the client SHOULD check all its validation expressions, supplying the characteristics of the uploaded file as values of the known fields of the ObjectData table. See the table 5-1.

The PostObject request MUST be sent using POST method. The request arguments for the transaction are sent using HTTP headers.

X.1 Required request header fields

In addition to the Required client request header fields specified in section 3.4, the header of any single-file message MUST contain the following fields:

UpdateAction	=*Add* | *Replace* | *Delete*
Content-type	= <mime-type as defined in 5.1>
Content-length	= <length of the posted data>
Type	= <object type as defined in table 11.11>
Resource	= <ResourceID of a resource as defined in table 11.3>

X.2 Conditionally required request header fields

ResourceID	=*resource-id*
ObjectID	=*1*5DIGIT*
UID	=*TOKEN*
OrderHint	=*1*5DIGIT*

resource-id is a value from the KeyField of the Resource for which the object is to be uploaded.

ObjectID is the order number of an object within the ID. It corresponds to the Object-ID argument for the GetObject transaction. UID is the UID of an existing object, as reported by the server in a previous PostObject, or in the related ObjectData class. OrderHint is a number suggesting where in the sequence of all objects belonging to the same ResourceID an uploaded object should be placed. Unlike ObjectID numbers, which must be an uninterrupted sequence of integral numbers starting with 1, the OrderHint may be any number. After an update, the server must modify ObjectID values for a given ResourceID, to ensure that the ObjectID order is the same as the OrderHint order. ObjectID numbers MUST follow the ordering of OrderHint numbers in the sense that if the OrderHint for object A is lower than the OrderHint for object B, then the ObjectID for object A MUST be lower that ObjectID for object B. In the case where multiple objects are set to share the same OrderHint value, the resulting ObjectID ordering is non-predictive. In the case of a multi-part upload, any ObjectID reordering that needs to be done to synchronize with the OrderHint order will be done by the server after all the objects are loaded.

The OrderHint MUST NOT be used in the PostObject request if it is not exposed in the ObjectData class linked to this object metadata (see 5.12 and Table 5-1). If it is used, the server should ignore the parameter.

Depending on the UpdateAction value, the ResourceID, ObjectID, UID or OrderHint may be missing from the request.

If UpdateAction=Add and the ResourceID and either ObjectID or OrderHint number is used, the uploaded file will be added to the list ofobjects. The server will adjust ObjectIDs to ensure that the uploaded object assumes appropriate position in the list of existing objects. If ObjectID is used, the server MUST increase the ObjectID by one for existing objects that have an ObjectID that is of the same value or greater than the ObjectID of the inserted object. If the number of existing objects is less then the ObjectID, the uploaded object becomes the last one in the list. If OrderHint is used, the server recalculates ObjectIDs of all objects so that they stay in sync with the order of the OrderHint values. If a client sends values for both ObjectID and OrderHint, the server MUST return error, preferably 20804 (Inconsistent parameters).

If UpdateAction=Add and the UID is given, the behavior is the same as if ResourceID and ObjectID of the object identified by the UID were requested. Namely, the uploaded file will be inserted before the object identified by UID. If any of ResourceID, ObjectID or OrderHint are used along with UID, the server MUST return error, preferably 20804 (Inconsistent parameters).

If UpdateAction=Add and none of ObjectID, OrderHint or UID is provided, the uploaded file becomes the last in the list of existing objects. ResourceID is required in this case. The server may set its OrderHint to any number higher that all existing OrderHints for this ResourceId.

If UpdateAction=Replace, either the ResourceID and ObjectID, or the UID MUST be used. The uploaded file replaces the original object. Any ObjectData fields not sent with the PostObject request (and not affected by the uploaded file) keep their previous values. If a client sends values for both ObjectID and UID, the server must return error, preferably 20804 (Inconsistent parameters).

If UpdateAction=Delete, either the ResourceID and ObjectID, or the UID MUST be used, and they MUST identify an existing object. The body of the request SHOULD be empty and MUST be ignored by the server. It is expected that the server will re-adjust the ObjectIDs such that there is no gap introduced by a Delete. If a client sends values for both ObjectID and UID, the server must return error, preferably 20804 (Inconsistent parameters).

The Delete request may also be sent with ResourceID without ObjectID, in which case all objects for the given ResourceID are to be deleted.

X.3 Optional request header fields

The client MAY specify any other headers. If the header coincides with a System Name of a field in a correspondent ObjectData class (see 5.12), the server MAY use the header's value to update that field. It's the server's decision whether such a field will be set to the client provided value, or calculated based on the uploaded file or other data. However, the server SHOULD calculate the values (rather than using the client-provided values) whenever it’s able to do so. Specifically, the field with standard name FileSize SHOULD always reflect the length of the file as it is stored with the server.

The client SHOULD check the ObjectData class to see what headers may be needed for the server.

Warning-Response	= *warning-num=user-response*

In case there were any warning while validating the request, the client MAY sent Warning-Response header. If a server responded to a previous PostObject request with a WarningBlock (see X.1.4), the client SHOULD include a Warning-Response header(s) when re-posting the request. The syntax and semantics of this header should resemble that of the WarningResponse argument from the Update transaction.

X.4. Request body

The body of the request is the file being uploaded. If UpdateAction=Delete, the body MAY be empty, and SHOULD be ignored by the server.

X.5 Server response body format

The response from the server is similar to that of the Update transaction (10.5):

```
<RETS 1*SP ReplyCode= quoted-reply-code 1*SP ReplyText= quoted-string *SP> CRLF
[pendingrcps: delimiter-tag ]
column-tag
compact-data
[pendingrcps:activation-tag]
[pendingrcps:error-block]
[pendingrcps:warning-block]
[<RETS-STATUS 1*SP ReplyCode= quoted-end-reply-code 1*SP ReplyText= quoted-string *SP/>
</RETS> CRLF
```

In the compact-data, the server MUST send the values of Resource, Type, ResourceID, ObjectID and UID, if they were used in the request. The UID, if it exists in the related ObjectData class, MUST be sent even if it was not requested. The UID MAY also be sent if no ObjectData class is linked to this Object metadata, but the server is able to honor GetObject requests with UID.

Unless the UpdateAction requested was Delete, all other fields from the ObjectData table that were requested or changed MUST also be sent. Other fields from the ObjectData table MAY be sent as well.

activation-tag	::= TIMESTAMP [pendingrcps: ; TEXT]

If the object is not immediately accessible, the server MUST send a datetime when it is supposed to be activated. An explanation why the object is delayed MAY be appended.

The reply code MUST be zero (success) even if the object is accepted for testing only. However, if the server knows in the time of the transaction that the object will be refused, it SHOULD reply with an error reply code (eg. 20809).

error-block and *warning-block* are explained in section 10.5.

X.6. Reply Codes Table X.1 Standard Reply Codes

0	Upload successful.
20800	Unknown resource
20801	Invalid object type
20802	Invalid identifier
20803	Invalid update action
20804	Invalid (inconsistent) request parameters
20805	No object found (for Delete)
20806	Unsupported MIME type
20807	Unauthorized
20808	Some objects not deleted (in case of Delete without ObjectID or UID, if some objects could not be deleted, while some were)
20809	Refused: object does not meet business rules
20810	FileSize too large Note that some servers MAY respond with HTTP status “413 – Request entity too large” if the uploaded file is larger than any acceptable limit.
20811	Timeout
20812	Too many outstanding requests
20813	Miscellaneous error

Chapter 4.10 (Capability URL List) should be modified to account for an optional PostObject transaction.

4. Compatibility

The ObjectData and PostSupport metadata fields in the Object metadata are optional. Clients implementing this proposal should make sure that they can handle metadata without these items to stay compatible with older servers. Older clients may have problems with new servers, since the metadata DTD is changed, but there is a chance even old clients will work correctly, if they are able to just ignore unknown metadata items.

Object data classes are regular classes, so they should not pose any compatibility risk.

Since previous versions of the specs did not allow for object upload, there are no compatibility issues with the PostObject transaction.

5. Acknowledgements

Thanks to Sergio Del Rio and all participants in the Rets Upload Workgroup for revising this proposal and making many valuable comments, which shaped this version of the document.

RCP 65 - Session information tokens

Original document: RCP 65 Session information tokens

Note: This RCP Affects the Following Sections:

- Section 4.4.1 Broker Code Argument
- Section 4.6 Login Response Body Format
- Section 4.7.1 Broker - deprecated
- Section 4.7.2 Member Name - deprecated
- Section 4.7.3 Metadata Version Information - deprecated
- Section 4.7.4 User information - deprecated
- Section 4.7.5 Session Information Tokens
- Section 4.8.1 Accounting Information - deprecated
- Section 4.8.2 Access Control - deprecated
- Section 4.8.3 Office List Information - deprecated
- Section 11.4.9.1 Validation Expression Types and Data Types
- Section 11.4.9.2 Validation Expression Special Operand Tokens

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Libor Viktorin
Organization	MarketLinx
E-mail	lviktorin at marketlinx dot com
Submitted Date	July 20, 2007
Originating Workgroup	
RETS Version	1.8.0
Status	Adopted

1. Synopsis

2. Rationale

Validation expressions currently use several special operand tokens (see table 11-34) to account for data such as UserID, BrokerCode etc, describing the current user. The values for these tokens are set in the Login transaction response.

With broader use of the update transaction and validation, it's becoming clear that the current tokens may not be sufficient for all situations. RETS needs a way to let the server provide any information pertaining to the current session. This document proposes a way to do it.

3. Proposal

In chapter 4.6, the normal "OK" response format should be given as:

```
<RETS 1*SP ReplyCode=quoted-reply-code
1*SP ReplyText=quoted-string *SP >
<RETS-RESPONSE>
[pendingrcps:member-name-key ]
[pendingrcps:user-info-key ]
[pendingrcps:broker-key ]
[pendingrcps:metadata-ver-key ]
[pendingrcps:metadata-timestamp-key ]
[pendingrcps:min-metadata-timestamp-key ]
[pendingrcps: office-list-key ]
[pendingrcps: balance-key ]
[pendingrcps: timeout-key ]
[pendingrcps: pwd-expire-key ]
*( info-token-key )
capability-url-list
</RETS-RESPONSE> [<RETS-STATUS [pendingrcps:1*SP ReplyCode=quoted-end-reply
-code
1*SP ReplyText=quoted-string * SP]/>
</RETS> CRLF
```

In each of chapters 4.7.1, 4.7.2, 4.7.3, 4.7.4, 4.8.1, 4.8.2 the following should be added:

> This argument is deprecated and will be replaced by the Session Information Token (see 4.7.5).
> Servers that claim to be backwards compatible with previous versions of RETS MUST still use this argument, but MUST also send the same information in a info-token-key.
> Servers that do not claim to be backwards compatible MUST NOT use this argument.
> Clients MUST use this information ONLY if an info-token-key with appropriate name is not provided; otherwise, the *info-token-value* of that token MUST be used instead.
> Clients that do not claim being backwards compatible with previous versions of RETS MUST NOT use this argument.

In chapter 4.8.3 the following should be added:

This argument is deprecated and will be replaced by the Session Information Token (see 4.7.5).
Note that in the OfficeList information token the values are delimitted by commas, rather than semicolons.
Servers that claim to be backwards compatible with previous versions of RETS MUST still use this argument, but MUST also send the same information in a info-token-key.
Servers that do not claim to be backwards compatible MUST NOT use this argument.
Clients MUST use this information ONLY if an info-token-key with appropriate name is not provided; otherwise, the *info-token-value* of that token MUST be used instead.
Clients that do not claim being backwards compatible with previous versions of RETS MUST NOT use this argument.

Following text should be inserted before Chapter 4.7.5. Current chapter 4.7.5 should be renumbered.

4.7.5. Session Information Tokens

info-token-key	::= **Info** = *info-token-name* [pendingrcps: ; *info-token-type*] ; *info-token-value CRLF*
info-token-name	::= *RETSNAME*
info-token-type	::= *TOKEN*
info-token-value	::= *TEXT*

Information token represents a named and typed piece of information about the current session. This information is sent by the server to the client to use in various occations, eg session and password management, creating search queries targetted to the current user, or in validation expressions (see table 11-34).

Any number of information tokens can be sent in the Login response, provided all of them have unique names.

The *info-token-type* is one of the DataTypes defined in table 11-9. If *info-token-type* is missing, the token's data type defaults to Character; the *info-token-value* MUST NOT include semicolons in this case. Otherwise, *the inf-token-type* may be any of the DataTypes specified in Table 11-9. The *info-token-value* must conform to the token's data type.

Names and types of well known tokens are listed in Table 4-1. The server MUST specify tokens shown in bold in that table. These tokens replace the deprecated information described in 4.7.1-4.7.4. The server also MUST specify well-known tokens providing information specified in optional response arguments (see 4.8), if these arguments are used. Clients MUST use the token values rather than the data in the response arguments; however, if the tokens are not present, the response argument values MUST be used.

If a token with a well-known name is specified without the *info-token-type*, client MUST cast it to the type shown in Table 4-1.

Names and types of well known tokens are listed in Table 4-1.

Table 4-1. Well-Known Information Tokens

Token name	Data type	Deprecated argument
USERID	Character	user-id
USERCLASS	Character	user-class
USERLEVEL	Int	user-level
AGENTCODE	Character	agent-code
BROKERCODE	Character	broker-code
BROKERBRANCH	Character	broker-branch
MEMBERNAME	Character	member-name
MetadataVersion	Character	metadata-version
MetadataTimestamp	DateTime	metadata-timestamp-key
MinMetadataTimestamp	DateTime	min-metadata-timestamp-key
Balance	Decimal	balance
TimeoutSeconds	Int	timeout-key
PasswordExpiration	Date	pwd-expr
WarnPasswordExpirationDays	Int	expr-warn-per
OfficeList	Character	office-list-key The value of the OfficeList token will be comma-delimited, rather than semicolon-delimited as it was in the case of the OfficeList response argument

More well-known Information Tokens may be added in later version of this document.

Following table should replace Table 11-34. The DataType Column should be used only if the Validation Expressions, as suggested by the Update Workgroup, use data types.

Token Name	Data Type	Description
.TRUE.	BOOLEAN	Boolean value of TRUE (1)
.FALSE.	BOOLEAN	Boolean value of FALSE (0)
.EMPTY.	EMPTY	A value that matches an empty field. Supplies an empty field when used in a SET expression.
.TODAY.	TIME	The current date.
.NOW.	TIME	The current time.
.ENTRY.	type of the current field	The new field value.
.OLDVALUE.	type of the current field	The original value of the field as returned from the host in the search operation. If the field is new, .OLDVALUE. is an EMPTY value.
.USERID.	CHAR	The value of the user-id field returned in the Login transaction, unless an info-token-key named USERID has been returned in the Login transaction.
.USERCLASS.	CHAR	The value of the user-class field returned in the Login transaction, unless an info-token-key named USERCLASS has been returned in the Login transaction.
.USERLEVEL.	CHAR	The value of the user-level field returned in the Login transaction, unless an info-token-key named USERLEVEL has been returned in the Login transaction.
.AGENTCODE.	CHAR	The value of the agent-code field returned in the Login transaction, unless an info-token-key named AGENTCODE has been returned in the Login transaction.
.BROKERCODE.	CHAR	The value of the broker-code field returned in the Login transaction, unless an info-token-key named BROKERCODE has been returned in the Login transaction.
.BROKERBRANCH.	CHAR	The value of the broker-branch field returned in the Login transaction, unless an info-token-key named BROKERBRANCH has been returned in the Login transaction.
.UPDATEACTION.	CHAR	Name of the UpdateAction for which this validation is performed.
.any.	(see Description)	If the name of the SpecValue (stripped of the first and last dot) is equal to a name of one of the info-token-keys returned as part of the Login response, then the type and value of this SpecValue is defined by that info-token-key. If no such info-token-key exists, the value is ERROR.

4. Compatibility

While the proposed solution makes the Login response cleaner, better organized and more flexible, it poses a big thread to a backwards compatibility. That's why the deprecated items are left in place, allowing old servers and clients to stay compatible.

In moving forward, both servers and clients are encouraged to break backward compatibility and drop the deprecated items as soon as they believe all the end-points they communicate with have implemented this proposal.

Breaking backwards compatibility will prepare servers and clients to the next stage, where the deprecated items will really be dropped.

RCP 68 - Search Has Key Index Support

Original document: RCP 68 Search Has Key Index Support

Note: This RCP Affects the Following Sections:

- Section 7.4.3 Limit
- Section 7.4.5 Select
- Section 11.3.1 Class
- Section 11.3.2 Table

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Matthew McGuire
Organization	MarketLinx
E-mail	mmcguire at marketlinx dot com
Submitted Date	June 9th, 2008
Originating Workgroup	RETS 1.7.1 Document
RETS Version	1.8.0
Status	Adopted 2008-08-07

1. Synopsis

This proposal addresses a common concern among client applications designed to replicate the data available on a given RETS server. This proposal attempts to resolve these concerns by providing additional metadata and modifying the Search Transaction to retrieve the keys for a given Class.

2. Rationale

Presently if a client application wishes to replicate data they must either acquire all of the records using a single search request, multiple requests using OFFSET, or multiple requests using a local index of keys. The first approach is often limited to a maximum number of records. The second approach is based on a feature that may not be supported by all servers and may not guarantee all the records. The third approach depends on a previously acquired set of keys which may or may not be complete at the time of use. Additionally there is no way for a server vendor to advertise support for data replication. By providing a metadata specifically designed to advertise keys used for replication, the client can acquire a list of keys for retrieving the records accurately and the server can advertise support for replication by providing the metadata.

3. Proposal

3.1 Specification Changes

The following sections detail each area of the existing specification that needs to be changed or clarified and provides reasoning related to each change. Each area of the change will be listed according to the section of the specification using numbering in italics. For example changes to the METADATA-SYSTEM response format would look like the following: “***Section 11.2.1***”.

This proposal extends the Search Transaction and affects server support for records limits in a specific way. Language to this effect is added to the following sections.

Section 7.4.3 Limit

The following sentence should be added to the end of the section.

Any request that sets a numeric Limit disables support for unlimited key index results as described in ***section 7.4.5 Select***.

Section 7.4.5 Select

The following sentence should be added to the end of the section.

If the requested Class advertises HasKeyIndex as True in the Class Metadata and the client only selects fields advertised as InKeyIndex as True in the Table Metadata, the Server MUST return all the matching records unless the Client has declared a Limit other than NONE.

Section 11.3.1 Class

The following additional Metadata Field should be added to Table 11-7 Metadata Content: Resource Class.

HasKeyIndex – Boolean – This value declares that a Class supports the retrieval of key data for fields advertised in the Table Metadata as InKeyIndex.

Section 11.3.2 Table

The following additional Metadata Field should be added to Table 11-9 Metadata Content – Tables.

InKeyIndex – Boolean – This value declares that a field may be used in the Select argument to suppress normal Search Limit behavior following the rule outlined in Section 7.4.5.

5. Compatibility

The additional metadata and behavior should not impact systems before version 1.7.1. Given the common need for this behavior, including this in the 1.7.1 is desirable.

RCP 69 - LookupType Value

Original document: RCP 69 LookupType Value

Note: This RCP Affects the Following Sections:

- Section 7.6.1 Query Language BNF
- Section 11.4.3 Table

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Steve Clarke
Organization	MarketLinx
E-mail	sclarke at marketlinx dot com
Submitted Date	June 6th, 2008
Originating Workgroup	RETS 1.7.1 Document
RETS Version	1.8.0
Status	Adopted 2008-08-07

1. Synopsis

Certain MarketLinx site configurations observed already do have occasional *LookupType* values defined that violate the current specification requirement that a *LookupType* value is ALPHANUM. In some cases these are in fact string fields that may have had a Lookup metadata added after the fact. Data in the database is already stored with non-ALPHANUM characters. Furthermore, in these situations, there are already RETS clients that are using this data in their feeds.

Under the current specification, we would have only two options for 100% compliancy in this area:

1. We can map these *LookupType* values to some arbitrary (100% alphanumeric) value and perform this mapping both when processing the search request as well as returning the data. This would not only require significant server development, but it would also add performance overhead and would impact all current implementations against these servers. Actually changing the data stored in the native databases is not an option because other MLS applications depend on this.
2. We could simply eliminate the *Lookup* from the metadata and treat these fields strictly as string fields. This would bring the servers into compliance, but it would reduce the quality of the metadata and the value of the server.

This proposal intends to make *LookupType* values more flexible by allowing them to behave in a similar capacity as regular string fields, while still providing the value of the Lookup metadata.

2. Rationale

The RETS standard already supports non-alphanumeric data in both queries and results. The DMQL2 BNF already supports string-literal, which is any TEXT except double-quote itself. DMQL parsers already can deal with this type of quoted string in search criteria against string fields.

With minor changes to the specification, a Lookup field can be treated exactly like a string field, except that it brings additional metadata that defines the set of possible values. This approach gives added flexibility to the systems without introducing additional overhead and without impacting existing client use cases.

3. Proposal

3.1 Specification Changes 11.4.3 Lookup Type

Table 11-20 Metadata Content: Lookup Type

Change content type for Value element from 1*32ALPHANUM to 1*128TEXT.

> Note: this change was already discussed and voted on in December 2007 (Miami) RCP by Matt McGuire, but at the time it deferred spec review WG . Unfortunately, this was missed in the 1.7.1 work done recently.

7.7.1 Query language BNF

Change definition of lookup element to:

lookup	::= <any legal string-literal value as per metadata>|<any legal TEXT value for the field as per metadata>

Sample search criteria:

HEAT=%7CBSBRD,ELEC,GEOTH
HEAT=%7CBSBRD,ELEC,GEOTH,”HV/AC”
HEAT=%7C”BSBRD”,”ELEC”,”GEOTH”,”HV/AC”
HEAT=%7C”BSBRD”,”ELEC”,”GEOTH”,”HV/AC”
HEAT=%7E”BSBRD”,”ELEC”,”GEOTH”,”HV/AC”
HEAT=%2B”BSBRD”,”ELEC”,”GEOTH”,”HV/AC”

Compatibility

At certain sites, MarketLinx servers already expose non-alphanumeric *LookupType* values in RETS metadata. MarketLinx servers already support the capability of quoting a *LookupType* value in search criteria. A wide variety of client vendors are already successfully using this feature. A few vendors are unable to query using quoted *LookupType* values and are hesitant to implement because this is not presently described in the specification. This proposal will not change the way the affected servers operate. It merely formalizes the approach that is already supported. Vendors that do not support non-alphanumeric *LookupType* values and clients that choose not to support this type of query are not affected at all.

RCP 70 - Metadata Role Support

Original document: RCP 70 Metadata Role Support

Note: This RCP Affects the Following Sections:

- Section 4.7 Required Response Arguments
- Section 11.2.1 System

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Matthew McGuire
Organization	MarketLinx
E-mail:	mmcguire at marketlinx dot com
Submitted Date:	July 27, 2007
Originating Workgroup:	
RETS Version:	1.8.0
Status:	Adopted

1. Synopsis

This proposal attempts to eliminate Metadata versioning issues with regard to users that fall into multiple roles or other arbitrary groupings. Presently the specification does not provide an identity for the current metadata beyond the version numbering. This proposal resolves this issue by defining a Metadata ID or name that can be used to persistently cache metadata related to a given role or grouping.

2. Rationale

The specification presently states the Metadata version must be increased anytime there is a change regardless of the identity that has logged in. As a result server hosts must increment the version number anytime the metadata has changed, regardless of the session context of the prior user. The end result is an ever increasing version number so long as any two users who have different metadata log in to the server in alternating sequence. This artificially increases the version, forces a RETS client application to retrieve metadata inefficiently, and prevents effective metadata version caching on the client side. This proposal attempts to resolve this issue using a persistent Metadata ID or name associated to the Metadata version that can be used by a RETS client for the purpose of caching metadata versions in an efficient and reliable manner.

3. Proposal

3.1 Specification Changes

The following sections detail each area of the existing specification that needs to be changed or clarified and provides reasoning related to each change. Each are of the change will be listed according to the section of the specification using the specification numbering in italics. For example changes to the METADATA-SYSTEM response format would look like the following: “***Section 11.2.1***”.

Section 4.6 Login Response Body Format

This section defines the information returned to a client during the login process. This response sets up session information for the RETS connection and can be used to know if the client metadata cache is up to date. The current response format is defined as follows.

```
<RETS 1*SP ReplyCode=quoted-reply-code 1*SP
ReplyText=quoted-string *SP >
<RETS-RESPONSE>
member-name-key
user-info-key
broker-key
metadata-ver-key
metadata-timestamp-key
min-metadata-timestamp-key
[pendingrcps: office-list-key ]
[pendingrcps: balance-key ]
[pendingrcps: timeout-key ]
[pendingrcps: pwd-expire-key ]
capability-url-list
</RETS-RESPONSE>
[<RETS-STATUS [pendingrcps:1*SP ReplyCode=quoted-end-reply-code 1*SP
ReplyText=quoted-string * SP]/>
</RETS> CRLF
```

Here the metadata version number and dates are advertised based on the login data of the user. Unfortunately this ties the metadata versioning to the ID of the user, which can be a problem for RETS client applications that need to cache metadata for multiple users. As a result multi-user RETS client applications must cache metadata for each user, which on a large system is neither scalable nor maintainable. To produce a scalable solution the Metadata System ID (see changes to ***Section 11.2.1 System Metadata*** below) can be reused to represent the current metadata model as defined for that session. By doing this a given metadata system ID can be used to cache a specific metadata model on the server. This will permit the multi-user client application to cache only the unique or separate metadata models as advertised by the server. On the server side, the server MUST manage metadata versioning relative to the unique System ID. By doing this the server may host multiple metadata roles without the need to increment the metadata version artificially as described above. To accomplish this, the above Login response format is changed to the following.

```
<RETS 1*SP ReplyCode=quoted-reply-code 1*SP
ReplyText=quoted-string *SP >
<RETS-RESPONSE>
member-name-key
user-info-key
broker-key
[pendingrcps:metadata-id-key]
metadata-ver-key
metadata-timestamp-key
min-metadata-timestamp-key
[pendingrcps: office-list-key ]
[pendingrcps: balance-key ]
[pendingrcps: timeout-key ]
[pendingrcps: pwd-expire-key ]
capability-url-list
</RETS-RESPONSE>
[<RETS-STATUS [pendingrcps:1*SP ReplyCode=quoted-end-reply-code 1*SP
ReplyText=quoted-string * SP]/>
</RETS> CRLF
```

This version adds a new response argument, metadata-id-key, which provides the necessary ID for efficient metadata caching. Since this ID must represent a persistent Metadata model for versioning the ID is directly related to the System metadata defined in ***Section 11.2.1 System Metadata***. As a result the next change defines this key as needed.

Section 4.7.3 Metadata Version Information

This section defines the existing metadata version keys defined in the Login response body format. Since the changes to Section 4.6 add a new required key, the following additional text for section 4.6 defines how this key is correctly used.

The metadata id key designates a persistent ID associated with the metadata applied to the current user session. This ID is defined in BNF as follows.

metadata-id -key	::= MetadataID = metadata-id CRLF	
metadata-id	::= 1*128(ALPHANUM | “_”)	

The metadata-id -key advertised by the server MUST match the Metadata ID attribute defined in ***Section 11.2.1 System Metadata***. This requirement explicitly binds the metadata advertised by the Login response to the metadata advertised by the Get Metadata response. This relationship is necessary to eliminate confusion and assist metadata updates. If following a Login response the Metadata ID does not match the Login response metadata-id -key value, the client MUST recognize that the metadata for the current session has changed and properly update the client metadata.

The additional language attempts to define the exact use of this login response information in relationship to the Metadata ID. Since the key defined above is a new string the Metadata ID is defined later in this proposal.

Section 11.2.1 System Metadata

The existing System Metadata response defines the System element with the following text.

System Description

code-name	::= 1*10ALPHANUM
long-name	::= 1*64PLAINTEXT
comments	::= TEXT

This definition sets the code name to a maximum of 10 characters. Since the Metadata ID will represent a persistent ID it needs more characters to facilitate uniqueness. To that effect the above text is changed to the following by this proposal.

System Description

code-name	::= 1*128(ALPHANUM | "_")
long-name	::= 1*128PLAINTEXT
comments	::= TEXT
metadata-id	::= 1*128(ALPHANUM | "_")

Here both the code and long name definitions have been increased to reflect the necessary detail. Note that the proposal defines the code name so that the underscore character is legal. This was done to improve readability of the code names advertised by the server.

Compatibility

Since this proposal adds new metadata (the Metadata ID) and adds new information to the Login response it is reasonable to expect backwards compatibility issues with client applications that do not adhere to the changes.

RCP 74 - Location Availability in Object Metadata

Original document: RCP 74 Location Availability in Object Metadata

Note: This RCP Affects the Following Sections:
- Section 11.4.1 Object

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Libor Viktorin
Organization	MarketLinx
E-mail	lviktorin at marketlinx dot com
Submitted Date	July 28, 2008
Originating Workgroup	Standards Committee
RETS Version	1.8.0
Status	Adopted 2009-04-07

1. Synopsis

This proposal recommends adding a flag to the Object metadata, revealing whether the server supports the Location=1 request.

2. Rationale

Retrieving objects' URL by specifying Location=1 is an option in the GetObject request, that may not be honored by the server. Currently, if the server does not support this functionality, it SHOULD respond with an error 20414.

The problem is, a client have no means of detecting if this functionality is supported before it sends a real request, and even then (since the error 20414 is optional) may not know for sure. Servers which do not support the Location=1 option, will send the full object data, which makes for a lot of data that the client did not ask for.

This document proposes a simple method for the server to advertize its support for Location=1.

3. Proposal

Add a line to table 11-16 (Metadata Content: Resource Object) in chapter 11.4.1:

LocationSupport	BOOLEAN	When true, indicates that the server will honor the Location=1 parameter at least for some objects. When false, indicates that the server does not support the Location=1 functionality

Modify the rets-metadata-content-1_7_2.dtd accordingly by

- adding **<!ELEMENT LocationSupport [#PCDATA]>** to the DTD
- adding **LocationSupport?**, to the Object element definition

4. Impact

This proposal updates the DTD by adding an optional element. Any previously existing metadata will be valid. Validation of new metadata using an old DTD would fail due to an unknown element, however this situation will be avoided by correct usage of the RETS Version.

4. Compatibility

This change modifies the DTD.

RCP 75 - Offset Availability in the Metadata

Original document: RCP 75 Offset Availability in the Metadata

Note: This RCP Affects the Following Sections:

- Section 11.3.1 Class

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Libor Viktorin
Organization	MarketLinx
E-mail	lviktorin at marketlinx dot com
Submitted Date:	July 28, 2008
Originating Workgroup	Standards Committee
RETS Version	1.8.0
Status	Adopted 2009-04-07

1. Synopsis

This proposal recommends adding a flag to the Resource metadata, revealing whether the server supports the Offset parameter in the Search transaction.

2. Rationale

The Offset parameter in the Search transaction is used by many clients to iterate thru large blocks of data. However, there are many servers that do not support this functionality. With such servers, the clients keep requesting with increasing offset number, but getting the same data all the times. The specification does not give any way to let the client know whether the Offset feature is supported or not.

This document proposes a simple addition to the metadata which will show the server’s support for the Offset feature on a per-class level.

3. Proposal

Add a line to table 11-7 (Metadata Content: Resource Class) in chapter 11.3.1:

OffsetSupport	BOOLEAN	When true, indicates that the server will honor the Offset parameter when searching this class. When false, indicates that the server does not support the Offset functionality for this class.

Modify the rets-metadata-content-1_7_2.dtd accordingly by

- adding **<!ELEMENT OffsetSupport [#PCDATA]>** to the DTD
- adding **OffsetSupport?,** to the Class element definition

4. Impact

This proposal updates the DTD by adding an optional element. Any previously existing metadata will be valid. Validation of new metadata using an old DTD would fail due to an unknown element, however this situation will be avoided by correct usage of the RETS Version.

5. Compatibility

This change modifies the DTD.

RCP 76 - GetPayloadList

Original document: RCP 76 GetPayloadList

> **Note: This RCP Affects the Following Sections:**
> - Section 11.3.1 Class
>
> The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Steve Clarke
Organization	MarketLinx
E-mail	sclarke at marketlinx dot com
Submitted Date	February 26, 2009
Originating Workgroup	Standards Committee
RETS Version	1.8.0
Status	Adopted 2009-04-07

1. Synopsis

The objective is to define a discovery mechanism for predefined XML payload documents that can optionally be accessed via RETS 1.x queries. This proposal defines a new transaction that advertises the available payloads on a RETS 1.x server and describes how these payloads would be accessed from a RETS 1.x query request.

2. Rationale

There has been a longstanding desire to be able to access standardized search results (payloads) via a RETS 1.x query request. Most urgently, the RESO community is looking for a way to provide a transport mechanism to support the Syndication XSD. This RCP satisfies that need.

3. Proposal

3.1 Login Response

Add new *GetPayloadList* transaction definition to login response.

Example:

```
<RETS ReplyCode="0" ReplyText="Success.">
<RETS-RESPONSE>
Info=MEMBERNAME;Character;MarketLinx Sys Config
Info=USERID;Character;866424041
Info=USERLEVEL;Int;90
Info=USERCLASS;Character;!P
Info=AGENTCODE;Character;sys_config
Info=BROKERCODE;Character;GEAC
Info=BROKERBRANCH;Character;GEAC01
Info-METADATAID;Character;303_65_47_BRC
Info=METADATAVERSION;Character;17.73.76591
Info=METADATATIMESTAMP;DateTime;Tue, 12 Feb 2008 23:16:31 GMT
Info=MINMETADATATIMESTAMP;DateTime;Tue, 12 Feb 2008 23:16:31 GMT
Info=BOARD;Character;X
Info=BROKERRECIPFLAG;Boolean;Y
Info=MAINOFF;Character;GEAC
Info=OFFICE;Character;GEAC01
Info=SUL;Int;90
Info=UC;Character;!P
Info=USER;Character;sys_config
ChangePassword=/ChangePassword.asmx/ChangePassword
GetObject=/GetObject.asmx/GetObject
Login=/Login.asmx/Login
Logout=/Logout.asmx/Logout
Search=/Search.asmx/Search
GetMetadata=/GetMetadata.asmx/GetMetadata
Update=/Update.asmx/Update
PostObject=/PostObject.asmx/PostObject
GetPayloadList=/GetPayloadList.asmx/GetPayloadList
</RETS-RESPONSE>
</RETS>
```

3.2 GetPayloadList Transaction

The GetPayloadList transaction is used to retrieve a list of available payloads supported on the RETS Server. Payloads are defined as a subset of the “RESO schema” vocabulary.

Required Client Request Header Fields

There are no additional required client header fields.

Required Request Arguments

None.

Optional Request Arguments

ID
The ID argument in the GetMetadata transaction reflects the metadata hierarchy as shown in Figure 11.1. For any metadata element, the ID argument is a list of the names of the parent elements for the desired element, separated by colons. For example, to retrieve the payload list for a given named Resource, the argument is simply the ResourceID.

/GetPayloadList?*ID=Property*

To retrieve the payload list for a specific class within a resource:

/GetPayloadList?*ID=Property:RES*

Format
The format option does not apply to the ***GetPayloadList*** request because the response is always considered to be in COMPACT format. See below.

Required Server Response Header Fields

None.

Required Response Arguments

There are no required response arguments.

Optional Response Arguments

There are no optional response arguments.

GetPayloadList Response Body Format

Example COMPACT reply:

```
<RETSPayloadList Resource="Property" Class="2" Version="1.00.000" Date= "2009-02-11T12:17:12-05:00" >
<COLUMNS>PayloadNameResourceClassDescriptionURIMetadataEntryID</COLUMNS>
<DATA>RESO SyndicationProperty1RESO Standard Syndication Payloadhttp://rets.org/xsd/Syndication.xsd1x123</DATA>
<DATA>RESO SyndicationProperty2RESO Standard Syndication Payloadhttp://rets.org/xsd/Syndication.xsd2x123</DATA>
<DATA>RESO SyndicationProperty3RESO Standard Syndication Payloadhttp://rets.org/xsd/Syndication.xsd3x123</DATA>
</RETSPayloadList>
```

Note: If there are multiple versions of a payload, the URI for the payload be the **Namespace URI** for the version of the schema that is supported. For example, one of the syndication versions is http://rets.org/xsd/Syndication/2008-03.

Reply Codes

RETS 1.7 requires all server responses to be well-formed XML, and additionally requires ***GetPayloadList*** responses to be valid XML. In addition, RETS requires that clients parse server responses as XML, not as simple text streams. The response formats shown here are normative with respect to content, but not normative with respect to form. That is, servers are free to produce response XML in any format that complies with the W3C XML 1.0 recommendation, so long as it is valid with respect to the appropriate DTD. XML escaping of content is implied, as is XML processing of white space and line endings. See the W3C *XML Recommendation 1.0, Third Edition*, for full information on XML.

Table X-X *GetPayloadList* Reply Codes

Reply Code	Meaning
0	Operation successful.
20500	Invalid Resource The request could not be understood due to an unknown resource.
20503	No Metadata Found No matching metadata of the type requested was found.
20511	Timeout The request timed out while executing.
20513	Miscellaneous error The server encountered an internal error.

3.3 Search Transaction

/Search?SearchType=Property&Class=2&Query=(ListPrice=300000-

)&QueryType=DMQL2&Count=0**&Payload=***RESO Syndication*

Note, a search request that specifies a *&Payload* **MUST not** include the *&Select* and *&Format* arguments. The payload itself defines the desired data elements and the format of the response.

Additional Reply Codes:
(ReplyCodes tbd because these should be maintained centrally by document manager)

- Payloads not supported at all.
- Invalid Payload for this Resource/Class
- Cannot combine &Select with &Payload.
- Cannot combine &Format with &Payload.

4. Impact

Compatible with 1.7.2 and above. This is a new transaction which is self-contained. A server can simply leave this out of their capabilities URL list in the Login response. If no Payloads are advertised, then none need to be supported in the search request/response.

5. Compatibility

RCP 77 - Maximum Field Length

Original document: RCP 77 Maximum Field Length

Note: This RCP Affects the Following Sections:

- Section 11.3.2 Table
- Appendix D

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Matthew McGuire
Organization	MarketLinx
E-mail	mmcguire at marketlinx dot com

Submitted Date	February 26, 2009
Originating Workgroup	Standards Committee
RETS Version	1.8.0
Status	Adopted 2009-04-07

1. Synopsis

This proposal recommends clarification of the meaning of the ***MaximumLength*** element in the *METADATA-TABLE* metadata. This proposal does not change the meaning of the ***MaximumLength*** element.

2. Rationale

The current specification defines the ***MaximumLength*** field as the ‘maximum possible un-encoded length of a value’ of the field. The definition refers to HTTP encoding specifically since RETS uses the HTTP transport for communication. Since this definition includes the term un-encoded it is possible that some developers interpret this with regard to a Lookup Value to Long Value encoding. The definition refers to Appendix D for examples in an attempt to clarify this. In Appendix D the examples show that the max length for string values applies to the Lookup Value not Lookup Long Value. Since this is not explicit developers have found this language to be ambiguous or confusing. This proposal attempts to clarify how the ***MaximumLength*** metadata value should be calculated based on the data type of the field.

3. Proposal

To solve this ambiguity define a formula of calculating the maximum character length of a given field using existing metadata information. A formula is defined for each of the appropriate RETS Data Types in the appendix. This will provide clear guidelines for how each data type should use the ***MaximumLength*** metadata information.

It is the responsibility of the client to accurately calculate storage requirements locally, based on the metadata provided by the RETS server. However, server vendors should adhere to the formulas presented here so there is consistent functionality using the ***MaximumLength*** metadata information.

Example: The maximum storage length for a multi-valued lookup field is calculated with the following formula.

(MaxSelect * (MaxValueLength + 3)) -1

The following variables are used in this calculation:

MaxSelect - The MaxSelect element (METADATA-TABLE). If MaxSelect is not provided, then we assume it is 1. This is a factor because if a field allows multiple selections, then the data storage will require space for the maximum number of instances of the data, potentially enclosed in quotes and separated by commas.

Example: “HW”,”WWC”,”CT”

MaxValueLength - The length of the longest Value entry (METADATA-LOOKUP_TYPE) that applies to the field being stored. The length of the Lookup Value is used in this formula assures that clients can handle any and all of the possible lookup values in the result data.

Note that both Lookup and LookupMulti fields can use the same calculation formula defined above.

3.1 Specification Changes

Clarify the ***MaximumLength*** definition and properly refer to the updated appendix D. Then add the maximum length calculation formulas to the Maximum Length appendix D.

Section 11.3.2 – Table

This section defines the Table Metadata used to define Field information. Table 11-12 defines the ***MaximumLength*** property of a field. This definition will change to the following:

The maximum possible character length of the value of the field after all Transport layer encoding. Transport layer encoding includes both HTTP and XML based encoding, but does not include RETS Lookup Value to Lookup Long Value encoding. See Appendix D for a definition and examples of how a RETS server should calculate the ***MaximumLength*** of a field based on the RETS data type.

Appendix D – Maximum Length

The language of this appendix will change to the following:

Opening:
This appendix defines formulas used to determine the maximum character length of field data based on the data type of the field. These formulas describe how the RETS server should calculate the ***MaximumLength*** for reliable use by RETS client applications.

Appendix D.1 – Maximum Length - Boolean

Any field with the Boolean data type should represent the ***MaximumLength*** as ‘1’. The definition of the Boolean data type requires a single character representation of the data.

Appendix D.2 – Maximum Length - Characters

Fields designated as the Character data type with an interpretation of Lookup or LookupMulti should calculate the ***MaximumLength*** according to the following:

(MaxSelect * (MaxValueLength + 3)) -1

Multiply the Max Select of the field (or 1 if not provided) by the character length (plus 3 for delimiters and quotes) of the longest Lookup Value in the Lookup metadata defined by the Lookup Name of the field. Finally subtract 1 to represent one less delimiter.

Example: The Field ‘Appliances’ has 10 Lookup items and has a Max Select of 4 lookup items. The longest Lookup Value character length is 6 for the Value ‘FRIDGE’ used for the Long Value ‘Refrigerator’. (4 X (6 + 3)) – 1 = 35
Appendix D.3 – Maximum Length - Integers

Fields designated as Integer numbers include the Tiny, Small, Int, and Long data types. These numbers are limited in size by their respective binary representations. Therefore a Tiny number is 8 bits and has a maximum value of 256. This results in a Maximum Character Length of 3. This can also be called the maximum Decimal Precision of the number. The Maximum Length of the field if advertised should be equal to or greater than the maximum Decimal Precision of the number. For ease of use each are provided here:

Tiny – ±4 – 2 characters
Small – ±32786 – 6 characters
Int – ±2147483648 – 11 characters
Long – ±9223372036854775808 – 20 characters
Appendix D.4 – Maximum Length - Decimal

Fields designated as Decimal only include numbers represented using a Decimal point. The Precision of the field determines the maximum number of decimal characters following the decimal point of the number. However, the maximum decimal precision of the data includes the decimal spaces before the decimal as well. For a decimal number the Maximum Character Length should match the maximum decimal precision of the value plus one to represent the decimal point itself. So that a signed 16 bit floating point number with a precision of 3 is a maximum value of ±32.786 which is 7 characters. The RETS standard does not define Decimal numbers by binary size so the server should advertise the Maximum Length as appropriate to the data exposed via the RETS server interface.

The Currency interpretation follows the same rules as a Decimal number with 2 decimal points of precision.

4. Impact

None

5. Compatibility

This change proposal strictly defines the Maximum Length property of the Metadata which may be interpreted differently by older applications. However, the previous language of the specification implies the same meaning so these changes should be backward compatible.

RCP 78 - Specification Errata Changes

Original document: RCP 78 - Specification Errata Changes

Note: This RCP Affects the Following Sections:

- Section 11.3.2 Table
- Section 11.4.2 Lookup
- Section 15.4 Transmission standards

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Paul Stusiak
Organization	Falcon Technologies Corp.
E-mail	pstusiak at falcontechnologies dot com
Author	Ryan Bonham
Organization	Transparent Technologies Corp.
E-mail	ryan at transparent-tech dot com
Submitted Date	April 6, 2009
Originating Workgroup	Specification Review Workgroup
RETS Version	1.8.0, 1.7.2, 1.5
Status	Adopted 2009-04-07

Synopsis

The Board of Directors approved an exception to the Change Proposal Process to provide an omnibus proposal, each section to be voted on separately, to address issues discovered during 1.7.2 implementations requiring immediate attention. Adopted issues will be collected as errata to the appropriate document, 1.7.2 or 1.5. The errata will over-rule the specification document text.

Rationale

Several minor issues have been discovered by various parties during the implementations of 1.7.2 compliant servers and the compliance tool. Since many vendors and the compliance tool are currently being work on to meet the 1.7.2 specification and the next meeting of the RETS workgroup is not until September 2009, a group of resolutions to create an errata document so that the intended behavior is documented and can be implemented during this development cycle.

Proposal

A. Modify the specification document for 1.7.2, Table 11-21 Metadata Content Lookup element Version to LookupTypeVersion and Date to LookupTypeDate to match the DTD.
B. Update the 1.7.2 BNF to support NULL in metadata. There are many places where the content of a metadata element may be empty as a practical matter. This is common in current implementations. The existing descriptions of many fields do not permit the field to be empty in Section 11 of the document. Those elements that can be NULL will have their BNF representation modified to permit this, reflecting current practice.
C. Update the 1.5 BNF to support NULL in metadata. This has the same rational as issue B.
D. Change the document rets-metadata-content-1.7.2.dtd, line 330 under the !ATTLIST METADATA-TABLE, from CLASS CDATA #IMPLIED to CLASS CDATA #REQUIRED
E. Modify the specification document for 1.5, Table 11-21 Metadata Content Lookup element Version to LookupTypeVersion and Date to LookupTypeDate to match the DTD and instruct the compliance tool implementation to be lenient in the interpretation of this clause, to accept both LookupTypeVersion and Version and LookupTypeDate and Date and to provide a warning or information statement in the detail report that "1.7.2 uses LookupTypeVersion and LookupTypeDate rather than Version and Date" or similar language.
F. Modify the specification document for 1.7.2, Table 11-12, Content-Type DataTime from "full-date" to "RETSDATETIME".
G. Modify the specification document for 1.7.2, Table 11-12, Content-Type Date from RETSDATE to full-date and Table 13-1 Type Date from "full-date RETSDATE" to "full-date".

Impact

This will require changes to existing implementation to bring them into compliance.

Compatibility

1.7.2 or 1.5 as appropriate.h3.Document History

Date	Version	Author	Description
06/04/09	1.0.0	P Stusiak	Initial Release
07/04/09	1.0.0	P Stusiak	Add changes F and G.

RCP 79 - Add Preferred Flag to GetObject Responses

Original document: RCP 79 Optional Query

> **Note: This RCP Affects the Following Sections:**
> - Section 5.6.3 Preferred
>
> The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Troy Davisson
Organization	FBS Data Systems, Inc.
E-mail	tdavisson at fbsdata dot com
Submitted Date	July 13, 2009
Originating Workgroup	RETS 1.7.1 Document
RETS Version	1.8.0
Status	Adopted 2009-09-23

1. Synopsis

This proposal recommends adding a flag to GetObject responses that indicates if the given object is the preferred or primary object for the requested record.

2. Rationale

GetObject currently allows a RETS client to request back an object with an object-id of 0 (zero) which will return the “preferred object” for that record, and it supports the ability for the preferred object to be an object other than object-id 1; however, no other indicator about the preferred object is currently provided back to the user unless they specifically request an object-id of 0.

A popular use of RETS involves RETS clients replicating available data and objects to their own systems. In these scenarios, RETS clients are requesting back all available objects for specific records, but due to the lack of a preferred object flag, they are unable to determine which of the objects is deemed preferred for that record without making an additional request for the 0 object-id.

In some MLS software systems, the ability for photos to be uploaded also allows for a particular photo to be marked as the “preferred” or “primary” photo for that record (without it necessarily being the first photo uploaded or moved as first in the list of photos) which indicates that this photo should be used on reports and other displays when only a single photo is shown.

4. Proposal

Add Section 5.6.3 under **5.6 Optional Server Response Header Fields**:
5.6.3 Preferred

Preferred	If the requested object is determined by the server to be the preferred object for the requested record, the server MAY return the “Preferred:” header with a value of 1 (true). If the server does not support this funtionality or if the requested object is not the preferred object, the server MUST return either “Preferred:” with a value of 0 (false) or not provide the “Preferred:” header.
Preferred	::= **Preferred:** *BOOLEAN*

Example: **Preferred: 1**

If the server is returning a multipart response, this header MAY be included in the MIME part headers for each object it applies to and MUST NOT be included in the MIME part headers for objects it doesn't apply to.

If the client is sending a request with an object-id of 0, the server SHOULD only include a “Preferred” header if the server further supports identifying preferred or primary objects when not using an object-id of 0.

4. Impact

Since providing the “Preferred” header is optional for the server for both single object responses and multipart responses, servers may choose to not implement this functionality and will still be compliant. RETS clients that don't recognize the header will simply ignore it. Any RETS clients that are able to recognize the “Preferred” header can provide additional information back to the end user or backend system. Because of this, there shouldn't be any compatibility issues.

5. Compatibility

No known compatibility issues.

RCP 80 - Optional Query

Original document: RCP 80 Optional Query

Note: This RCP Affects the Following Sections:
- Section 7.3.2 Query Specification(moved)
- Section 7.4.9 Query Specification (new)

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Troy Davisson
Organization	FBS Data Systems, Inc.
E-mail	tdavisson at fbsdata dot com
Submitted Date	August 17, 2009
Originating Workgroup	
RETS Version	1.8.0
Status:	Adopted 2009-09-23

1. Synopsis

This proposal recommends changes to RETS to allow the Search transaction Query and QueryType parameters to be optional.

2. Rationale

A large percentage of RETS usage in production environments is made up of 3rd party programs or vendors connecting to an MLS server and downloading all available property data. This is often referred to as “replicating” which allows the client system to contain the same information available from the source RETS server.

Currently, the DMQL2 query language used by RETS requires that a search contain at least 1 query condition in order to pull records back. For example, a client is able to send “Query=(ListingOfficeId=123456)” to retrieve back only records available from a specific listing office.

In a scenario where replicating the MLS data is desired, a “filler” query is often sought in order to provide the required single condition but must be done in a generic way as to not limit the returned results in order to successfully replicate a full dataset. For example, a client may send “Query=(ListingPrice=0+)” in order to return all records where the ListingPrice is $0 or greater. When a “filler” query isn't immediately obvious, other creative solutions often soon follow. For example, “Query=(YearBuilt=1700+)” or “Query=(Remarks=**a**)|(Remarks=**e**)|(Remarks=**i**)|(Remarks=**o**)|(Remarks=**u**)”.

On the client side, users are forced to investigate field definitions prior to being able to see any records from the RETS server. Once a field has been identified and a generic query condition has been thought up, the idea must then be translated into proper DMQL2 syntax. These additional steps make it much harder for someone to begin downloading and using data.

On the server side, users are generating queries that may work against optimizations made to the storage of data. For example, a server may not respond as quickly to a vowel search on the Remarks field as it would a YearBuilt field. As a result, the required query condition is adding burden to the end users and forcing them to do something that may not be efficient for the server to handle.

By making the Query and QueryType parameters optional, users only need to provide 2 required parameters (SearchType and Class) and are able to instruct the server to return all records available to their account without the need to:

1. interrogate field data types and possible lookup values,
2. learn DMQL2 syntax for one-time use,
3. generate inefficient queries, and
4. generate additional support requests (to MLS staff, vendor support staff, open source software mailing lists, etc.).

A server is able to handle an absent query condition in a way that makes the most sense to that implementation.

In the same way that by not providing a 'Select' parameter instructs the server to return all available fields, not providing a 'Query' parameter would instruct the server to return all available records.

3. Proposal

Under **7.3.2 Query Specification**, add the following text at the end of the Query paragraph:

Clients are not required to provide a Query parameter unless one or more fields in the requested Class are marked as Required (see Table 11-12) in which case an error type of 20203 MUST be returned if the server is refusing to accept this request. If the server accepts this request, it MUST interpret the absence of a Query parameter as a request by the client to forfeit it's option of filtering records past the filters that may be automatically applied by the server.

Under **7.3.2 Query Specification**, add the following sentence at the end of the QueryType paragraph:

Clients MUST provide the QueryType parameter if the Query parameter is also sent.

Move **7.3.2 Query Specification** to s**ection 7.4 Optional Request Arguments**.

Reassign section number as needed.

4. Impact

Currently, servers are expecting to receive Query and QueryType parameters and must be modified to make these parameters optional and to interpret this request correctly.

Clients will need to be modified to allow for Query and QueryType to be optional if a query isn't given but continuing to provide Query and QueryType parameters does NOT break their compliance. The ability for a client to request all available, allowed records using this technique is optional.

Compatibility

For versions of RETS prior to the version this is adopted for, a server CAN implement this functionality without affecting users; however, RETS clients can only expect to be able to use this functionality in the adopted version and therefore can only expect results consistent with the requirements set forth in this proposal for the adopted version.

RCP 82 - LookupMulti Quoting Rule

Original document: RCP 82 LookupMulti Quoting Rule

Note: This RCP Affects the Following Sections:
- Section 11.4.3 Lookup Type

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Matthew McGuire
Organization	MarketLinx Inc.
Email	mmcguire at marketlinx dot com
Status	Adopted
Submission Date	October 3, 2008
Version	1.8.0

1 Synopsis

This proposal corrects an omission in the LookupMulti metadata definition.

2 Rationale

The description of Interpretation = LookupMulti states the following:
The character strings MAY be quoted text following the rules for Value of section 11.4.3 Lookup Type.
Unfortunately there are no rules in section 11.4.3 for this reference.

3 Proposal

3.1 Specification Changes

1. The following text can be added to the end of section 11.4.3 to correct this omission.
2.

In a search response the values of a LookupMulti field may include comma characters. Any values of a LookupMulti field that contains the comma character MUST be quoted as a string-literal.
Examples:
"Value1","Value2","Comma,Value"
"Value1",Value2,"Comma,Value"
Any code that is quoted MUST be interpreted without the quotes when referencing the Lookup value in the metadata.

4. Compatibility

As a clarification this proposal poses no backwards compatibility concerns.

RCP 87 - RETS 1.7.2 Errata Document

Original document: RCP 87 RETS 1.7.2 Errata Document

Note: This RCP Affects the Following Sections:
- Section 3.3 Required Client Request Header Fields
- Section 3.5 Response Format
- Section 3.8 Data Compression in RETS Transactions
- Section 3.10 Computing the RETS-UA-Authorization Value
- Section 11.2.2 Resources
- Section 11.3.2 Table
- Section 11.4.2 Lookup
- Section 11.4.8 Validation Lookup Type DEPRECATED
- Appendix B.3 Foreign Keys
- Appendix B.8 Object
- Appendix B.18 Validation External Type
- Section 15.1 Overall format
- Section 15.4 Transmission standards

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Paul Stusiak
Organization	Falcon Technologies Corp
Email:	pstusiak at falcontechnologies dot com
Workgroup:	Standards Committee
Submission Date	2010-01-21
Status:	Adopted, 2010-03-24
Version:	1.8.0

Synopsis

Several errata have been discovered in the 1.7.2 document and RCPs. They have been collected into an errata document to assist users in implementing the 1.7.2 standard correctly.

This document incorporates the critical errata.

Rationale

Several problems that may be major or critical have been identified within the 1.7.2 standard and may impede interoperability between client and server and between server systems.

Proposal

The document "RETS 1.7.2 Errata 2010-04" contains the known major and critical errata to the 1.7.2 standard. Adoption of this document is recommended to assist developers in implementing 1.7.2.

The document is provided in-line for convenience.

Introduction

In general, the specification document body text represents the correct information. This is not true in all cases. This document describes several problems with the 1.7.2 document.

Erratum 1

Table 11-40, Validation External refers to a Field called ValidationExternalName. This is correct. Table 11-42, Validation External Type refers to a Field called ValidationExternalName. This is correct. The example in B.18, ValidationExternalType has a field tag of ValidationExternal. This is not correct. It should be replaced by ValidationExternalName.

Reported by Paul Stusiak

Reported by Libor Viktorin

Reported by Phillip Paulson

Erratum 2

Table 11-12, METADATA-TABLE the 'Date' data type is *RETSDATE*. There is no *RETSDATE* defined in 'Section 2.3, Atoms and Primitives'. The correct value for any reference to this atom is *full-date.*Using the atom *full-date* matches the format from the 1.5 specification document.

Table 13-4 also has a reference to *RETSDATE*. This should read*full-date.*

Reported by Paul Stusiak

Reported by Joshua Vosper

Reported by Mark Klein

Reported by Troy Davisson

Erratum 3

Table 11-12, METADATA-TABLE the 'DateTime' data type is *full-date.*The correct value for this data type is *RETSDATETIME*. Using the atom full-date matches the format of the Time RCP. This is different from the 1.5 specification document.

Reported by Mark Klein

Reported by Troy Davisson

Erratum 4

In section 11.4.8, Table 11-34 Validation LookupType refers to a tag*PARENTFIELDS* that is not defined in the document or the DTD. By inference, the correct information is *PARENTFIELD1* and*PARENTFIELD2*, described in Table 11-32 of section 11.4.7 previous section.

Reported by Libor Viktorin

Erratum 5

In section 3.3, the term *RETS-Version* describes the version as a three element value of the form "<major>.<minor>.<release>" and discusses it in terms of RFC 2616. RFC2616 only has a two element version described. The intent is to make sure that the version is treated numerically instead of alphabetically when comparing versions.

The relevant section of RFC 2616 is section 3.1

HTTP uses a "<major>.<minor>" numbering scheme to indicate versions

of the protocol. The protocol versioning policy is intended to allow

the sender to indicate the format of a message and its capacity for

understanding further HTTP communication, rather than the features

obtained via that communication. No change is made to the version

number for the addition of message components which do not affect

communication behavior or which only add to extensible field values.

The <minor> number is incremented when the changes made to the

protocol add features which do not change the general message parsing

algorithm, but which may add to the message semantics and imply

additional capabilities of the sender. The <major> number is

incremented when the format of a message within the protocol is

changed. See RFC 2145 [36] for a fuller explanation.

The version of an HTTP message is indicated by an HTTP-Version field

in the first line of the message.

*HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT*

Note that the major and minor numbers MUST be treated as separate

integers and that each MAY be incremented higher than a single digit.

Thus, HTTP/2.4 is a lower version than HTTP/2.13, which in turn is

lower than HTTP/12.3. Leading zeros MUST be ignored by recipients

and MUST NOT be sent.

The meaning of this is that the version should have the following BNF representation:

*RETS-Version = 1*DIGIT "." 1*DIGIT "." 1*DIGIT*

While in general, this should be a sufficient definition, the use of the*RETS-Version* as a maker for versioning metadata may require additional digits to correctly represent the version of metadata. Specifically, implementers should be permissive in the use of*RETS-Version* and should accept values where there are more than a single digit for the release or minor positions.

Reported by Mark Klein

Erratum 6

In the specification document, on page 11-19 in 'Table 11-21 Metadata Content: Lookup' the field names include 'Version' and 'Date'. This is incorrect. The document values should be changed to LookupTypeVersion and LookupTypeDate.

The 1.7.2 DTD set has the correct values, 'LookupTypeVersion' and 'LookupTypeDate'.

Reported by Ryan Bonham

Erratum 7

The METADATA BNF does not allow for empty fields. This contradicts the descriptions next to the BNF, that often states that the field is optional and the DTD that states that they are not required. Common practice also has implementations that permit empty fields. The BNF is incorrect and the BNF for metadata should be revised to allow for NULL values. People implementing RETS systems should be aware that many server systems may permit empty fields.

Reported by Ryan Bonham

Erratum 8

The Foreign Key Metadata element has an inconsistency in the 1.7.2. The tag label for the metadata content is referenced in three places.

METADATA-FOREIGN_KEYS is specified on page 11-9 of Section 11.2.3

METADATA-FOREIGN_KEYS is specified in the RETS 1.7.2 DTD on page 6 in line 273

METADATA-FOREIGNKEYS is specified on page B-2 of Appendix B-2

It is also referenced in Table 11-12, Metadata Content, Field ForeignKeyName in the Description column as METADATA-FOREIGNKEYS.

The correct use is METADATA-FOREIGN_KEYS.

Reported by Mark Klein.

Erratum 9

The BNF definition of RETS-Response has a missing character '['. It should read

RETS-response::=body-start-line

response

[rets-status]

[body-end-line]

Reported by Gary Little

Erratum 10

The text of Section 3.5, reply codes is confusing. In 1.7.2, it reads

Applicable reply-codes can be found under specific transactions.

A revised version of this sentence is

reply-codes are specific to a transaction. Please refer to the applicable transaction for the meaning of the reply-code or refer to Appendix C of this document for a consolidated list.

Reported by Gary Little

Erratum 11

The last sentence of the first paragraph of Section 3.8 should read

If the server supports one of the compression methods accepted by the client, it can include a Content-Encodingheader in its response indicating the compression method it has chosen.

Reported by Gary Little

Erratum 12

In Table 11.6 of the METADATA-RESOURCE section, the *KeyField* is currently defined as a *RETSID*. This should be replaced by*RETSNAME* to make it consistent with the later reference to SystemName in Table 11-12.

Reported by Mark Klein

Erratum 13

The multipart example on page 43 under section 5.11.1 General Construction and again under 5.11.2 contains a very minor formatting error.

In a multipart response, all boundaries need to appear as "CRLF--boundary" which means that every boundary will have a CRLF in front of it.

Every HTTP response needs to contain a blank line between the HTTP header and HTTP body. As a result, the first boundary needs to be 2 blank lines away from the HTTP headers. On servers that don't send the "2" CRLF's, RETS clients miss the first boundary and discard it (since they treat it as the preamble) and start processing at the 2nd boundary for the 2nd photo. As the document is right now, the example is sharing the blank line between the HTTP and boundary divider.

The work around on the client-side (unless the developer thinks that putting in this "hack" is too dirty): automatically pad the HTTP body with CRLF's since everything before the first boundary and everything after the last boundary is ignored by default. If the server already has this correctly, the extra line breaks are ignored anyway. If the server doesn't have this correctly, the padding makes the response good automatically.

Reported by Troy Davisson

Erratum 14

Table 11-12 has a field named ModTimeStamp. The description is incorrect. It currently reads

When true, indicates that changes to this field update the class’sModTimeStamp field.

There is no ModTimeStamp field in the Class metadata. It should read

When true, indicates that changes to this field will cause the Class header *Date*field and the Class body *TableDate* field to have the value of the date and time of the change.

Reported by Joshua Vosper

Erratum 15

The improvements to LookupMulti require the change proposal LookupMulti Quoting Rule, adopted at the December 2008 meeting in Scottsdale, Arizona.

Reported by Matt McGuire

Reported by Ryan Bonham

Erratum 16

Section 13.1 requires revision to "... if a particular field for some record is undefined or is suppressed for authorization reasons, the value MUST be represented by two delimiters with no intervening space unless the restricted indicator is set. In that case, the value MUST be represented by the restricted indicator."

Reported by Ryan Bonham

Erratum 17

Section 3.10 describes the calculation of the User Authorization digest. This description conflicts with the definition described in Section 3.4. Specifically, section 3.10 defines the calculation as HEX while section 3.4 defines the product of section 3.10 as LHEX. Since this is based on the digest authentication scheme of RFC 2617 which uses LHEX, it is suggested that section 3.10 be changed to refer to LHEX rather than HEX. Thus, the product term should read:

ua-digest-response::= LHEX(MD5(LHEX(a1):RETS-Request-ID:session-id:version-info))

Reported by Rob Overman

Erratum 18

The example in Section B.8, METADATA-OBJECT includes a field called StandardName. This field is not described in the definition for METADATA-OBJECT in Table 11.19 and should be removed from the example.

Reported by Sergio Del Rio

Erratum 19

The definition of MIMEType for METADATA-OBJECT in Table 11.19 implies that only a single mime type may be expressed for a specific object type. This is not how RFC 2616 and RFC 2045 describe the Accept parameter. It should be possible for a single ObjectType to have more than one mime-type, for example, a Photo may have image/jpg and image/gif as mime-types. The correct Content Type for this field should read "A comma separated list of MIME type/subtype per 2045". The correct Description for this field should read "The mime-type/subtypes of the object type. This is the collection of object media encodings available for the objects on this system. Objects may have one or more mime-type of those listed in this field. This list is the mime-types that can be passed by the client in the "Accept" parameter in the GetObject transaction. All objects can return a mime-type of text/xml as an error code/error reply when a fault occurs in the GetObject transaction."

Reported by Paul Stusiak

Clarification 1

The Compliance Workgroup requested that a note be added to Figure 11.1 Metadata Structure. This note should identify that the box labels are not the header tag values used in a GetMetadata request. The suggested working is "The names of the metadata provided in the figure are not indicative of the header tag values that should be used in a GetMetadata transaction Request or Response. For the proper metadata-id value please refer to the RETS 1.7.2 DTD."

Reported by the Compliance Workgroup

Impact Compatibility

This change proposal is only compatible with the 1.7.2 standards document

RCP 90 - Deprecate CommonInterest Class Well-Known Name

Original document: RCP 90 - Deprecate CommonInterest Class Well-Known Name

> **Note: This RCP Affects the Following Sections:**
> - Section 11.3.1 Class
>
> The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Author	Paul Stusiak
Organization	Falcon Technologies Corp
Email:	pstusiak at falcontechnologies dot com
Workgroup:	Data & Schema
Submission Date	2010-01-21
Status:	Adopted, 2010-03-24
Version:	1.8.0

Synopsis

The well-known Class name CommonInterest is not widely used in implementations of RETS. Many implementations treat the CommonInterest property as either a sub-type or as an ownership type. It is recommended that the well-known name CommonInterest be deprecated.

Rationale

During the revision to the StandardNames data dictionary, the workgroup found that the participants did not use the well-known Class name CommonInterest. The participants were concerned that having this name was causing confusion in the industry, since the standards body names a Class that is not commonly used in the industry.

After discussion it was recommended that the Class name CommonInterest be deprecated as a well-known name in the standard.

Proposal

The well-known class name CommonInterest should be marked as deprecated in the standards documents.

Impact

Updates to the standards documents will be required. There should be no impact to system users.

Compatibility

1.7.2 and later.

RCP 91 - StandardNames Version Information in Login Transaction

Original document:RCP 91 - StandardNames Version Information in Login Transaction

> **Note: This RCP Affects the Following Sections:**
> - 4.7.5 Session Information Tokens
>
> The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Organization:	Falcon Technologies Corp
Name:	Paul Stusiak
Email:	pstusiak@gmail.com
Version:	1.8.0
Status:	Adopted
Adopted Date:	September 28, 2010

Synopsis

Different versions of StandardNames exist. This change proposal provides a method to express which version of StandardNames is used by a specific instance of a RETS server.

Rationale

Client vendors are not currently able to determine which set of StandardNames a specific instance of server has using the standard. This determination is currently done by contacting the operator and asking. This means that the client vendor cannot simply deploy their software and must manually configure the software to identify the version of the StandardNames that are used.

Proposal

The change proposal will add an optional body response line that will provide the version information of the set of StandardNames that this server uses. This will use the format described in RCP 65 - Session information tokens and described in **Section 4.6, Login Response Body Format**, response "OK" of the RETS 1.8.0 document.

A new section will be created and will read;

<new-text>

Section 4.7.5 Session Information Tokens

Add to Table 4-1 Well-Known Information Tokens

StandardNamesVersion	Character	*standard-names-version*

standard-names-version ::= *1*128TEXT CRLF*

The *standard-names-version* indicates the date version of StandardNames that this system supports. A system is only expected to support a single version of the StandardNames and, in most cases, this will be the current version.

Server systems that do not provide this optional field make no representation about the version of StandardNames that they support, therefore, client applications should not assume any specific version of the StandardNames.

Server systems that do provide this optional field **MUST** return a value for the standard-names-version that matches one of the values from the Adopted StandardNames List from Real Estate Transaction Standard website.

The format of the *standard-names-version* is a string *YYYY-MM* where *YYYY* is the year of adoption and *MM* is the month of adoption. For example, a version of the StandardNames is *2010-04*

</new-text>

Impact

No impact

Compatibility

Forward compatibility with 1.8 and higher versions of the Standard.

RCP 92 - RESO Payload Transport-Level Metadata Support

Original document:RCP 92 - RESO Payload Transport-Level Metadata Support

Note: This RCP Affects the Following Sections:

- 7.5 Search Response Body Format

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Organization:	Falcon Technologies Corp
Name:	Paul Stusiak
Email:	pstusiak@gmail.com
Version:	1.8.0
Status:	Adopted
Adopted Date:	September 28, 2010

Synopsis

Add to Section 7.6 an example response body for Payload=<RESO schema name> and modify the rets-xml-search-response-1_8_0.xsd to provide responses for each of the adopted RESO schemas.

Rationale

RCP 76 added a GetPayloadList Transaction to the standard that permits the use of RESO schema as a response format on a Search

Transaction.

RCP 76 is silent on the response body format that using this argument will produce. This change proposal resolves that question.

Proposal

Correct the structure of Section 7.6 Search Response Body Format to place the elements with the appropriate format type and place the common elements at the bottom of the section.

Modify Section 7.6 to add an example of the response body when the Payload=<RESO Schema Name> is used on a SearchTransaction.

<new text>

The body of the search response has the following format when replying to a payload request for a RESO schema:
<?xml version="1.0" ?>
[doctype]
<RETS 1*SP ReplyCode= quoted-reply-code 1*SP ReplyText= quoted-string *SP >
[count-tag]
*({{ RESO-data-record)}}
[max-row-tag]
[<RETS-STATUS [rcpdropbox:1*SP ReplyCode= quoted-end-reply-code 1*SP
ReplyText= quoted-string *SP]/>]
</RETS> CRLF

RESO-data-record ::= <A record structure as defined in the rets-xml-search-response-1_8_0.xsd document>

</new text>

Update the rets-xml-search-response-1_8_0.xsd to provide for the response of each of the root RESO Schema: Listings, Media, Members, Offices, Properties, PublicRecords, Syndication, Teams.

Impact

None.

Compatibility

Compatible with 1.8

RCP 93 - Add Content-Sub-Description to GetObject

Original document:RCP 93 - Add Content-Sub-Description to GetObject

Note: This RCP Affects the Following Sections:
- 5.6 Optional Server Response Header Fields

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Organization:	FBS Data Systems
Name:	Troy Davisson
Email:	tdavisson@fbsdata.com
Version:	1.8.0
Status:	Adopted
Adopted Date:	September 28, 2010

Synopsis

This proposal recommends changes to RETS to add a "Content-Sub-Description" header to GetObject responses.

Rationale

The RETS specification currently supports sending a "Content-Description" header within GetObject responses to help describe the object being returned. While this description is often very helpful for those users retrieving objects, it often doesn't include all of the descriptive data that's saved with those objects. For example, when photos are posted within an MLS system, agents can often enter a description and separate caption for the pictures. Delivering both text descriptions within RETS is not currently possible using the single "Content-Description" header currently supported.

Proposal

Under **5.6 Optional Server Response Header Fields**, add the following sub-section:

> ***5.6.X Sub-Description***

> | | |
> |---|---|
> | *Sub-Description* | *A secondary description of the object.* |
>
> | | |
> |---|---|
> | *Sub-Description* | *::=* ***Content-Sub-Description:*** `*1024<PlainTEXT, excluding CR/LF>` |
>
> *Example:* `Content-Sub-Description: Enjoy the evening sunsets from the front porch`
>
> *If the object does not have a sub-description or if the server does not support this feature, the header MAY not be returned. If the object has a sub-description and the server is returning a multipart response, then this header MUST be included in the MIME part headers for the object.*

Impact

Slightly larger GetObject response sizes to accommodate the new information.

Compatibility

Including this header in prior versions of RETS servers should not cause issues for clients parsing GetObject responses correctly, but clients can only expect to see this optional field when using this version of RETS or higher.

RCP 94 - Improved Error Handling in GetObject

Original document:RCP 94 - Improved Error Handling in GetObject

Note: This RCP Affects the Following Sections:

- 5.6 Optional Server Response Header Fields

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Organization:	FBS Data Systems
Name:	Troy Davisson
Email:	tdavisson@fbsdata.com
Version:	1.8.0
Status:	Adopted
Adopted Date:	September 28, 2010

Synopsis

This proposal recommends changes to RETS to better report errors within GetObject responses.

Rationale

The RETS specification allows a RETS server to deliver all kinds of object types through GetObject responses including image/jpeg, text/html, image/png and text/xml. When a RETS server returns a text/xml object in a GetObject response (as either a single object or part of a multipart response), a RETS client is unable to determine if the object itself represents a wanted XML document or if the XML describes a RETS error simply by parsing the HTTP response. The RETS client must go further and parse the actual XML response to determine if the XML document describes a RETS error to determine what further steps might be needed.

By making the changes outlined in this proposal, a RETS client is able to determine if a GetObject response describes an error or not prior to parsing the response body.

Proposal

Under **5.6 Optional Server Response Header Fields**, add the following sub-section:

> #### *5.6.X RETS-Error*
>
> | | |
> |---|---|
> | *RETS-Error* | *If a server is unable to deliver a requested object which generates an error, "RETS-Error" MUST be included and have a value of 1 (true). Otherwise, RETS-Error MUST NOT be included as a header.* |
>
> | | |
> |---|---|
> | *RETS-Error* | *::=* ***RETS-Error:*** `BOOLEAN` |
>
> Example: `RETS-Error: 1`
>
> *If the server is returning a multipart response, this header MUST be included in the MIME part headers for each object it applies to and MUST NOT be included in the MIME part headers for objects it doesn't apply to.*

Under **5.11.2 Error Handling**, the entire section should read (text changes marked in red):

> *When a client requests multiple objects in a single transaction, one or more of those objects may be unavailable. In this case, the server communicates the failure by including a "RETS-Error" header and RETS return message in place of the unavailable object. In this case, the Content-Type MUST be text/xml, and the content will be a RETS response:*
>
> *Example:*
>
> ```
> HTTP/1.1 200 OK
> Server: Apache/2.0.13
> Date: Fri, 22 OCT 2004 12:03:38 GMT
> Cache-Control: private
> RETS-Version: RETS/1.7.2
> MIME-Version: 1.0
> Content-type: multipart/parallel; boundary="simple boundary"
>
>
> --simple boundary
> Content-Type: image/jpeg
> Content-ID: 123456
> Object-ID: 1
>
> <binary data>
> --simple boundary
> Content-Type: text/xml
> Content-ID: 123457
> Object-ID: 1
> RETS-Error: 1
>
> <RETS ReplyCode="20403" ReplyText="There is no listing with that
> ListingID"/>
>
> --simple boundary--
> ```
>
> *If the server is supplying an error message for a wild-card object request (Object-ID of *), the Object-ID for the error part SHOULD be * as well.*

Impact

Improves clients ability to detect errors while adding minimal size to error responses

Compatibility

Clients that support prior versions of RETS will continue to need to parse text/xml responses to determine if the server reported an error and cannot expect servers to send a "RETS-Error" flag for older versions of RETS.

RCP 98 - Additional Information Fields in METADATA-SYSTEM and Login

Original document:RCP 98 - Additional Information Fields in METADATA-SYSTEM and Login

Note: This RCP Affects the Following Sections:

- 4.7 Required Response Arguments
- 11.2 System-Level Metadata

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Organization:	Falcon Technologies Corp
Name:	Paul Stusiak
Email:	pstusiak@falcontechnologies.com
Version:	1.8.0

Status:	Adopted
Adopted Date:	September 28, 2010

Synopsis

To correct the use of META-SYSTEM, several fields should be added for the Foreign Keys and the Filters. To assist Client developers, additional information fields would be useful in the during the login transaction.

Rationale

Minor variations exist between the operation, behavior and information presented in different locations and different Server vendors. To assist Client developers to manage their code, the metadata and data, adding information about the vendor, the operator (MLS) and some other information would make such management simpler.

The addition of ForeignKey and Filter at the same level as that of Resource (see Figure 11-1 in the Standards Document), should have caused additional version and date information to be included in the METADATA-SYSTEM information to allow queries of only METADATA-SYSTEM to compare the current metadata information against that of any cached information.

Proposal

Add to Table 4-1, Section 4.7.5 the following information tokens:

Token name	Data type	Deprecated argument
VendorName	Character	none
ServerProductName	Character	none
ServerProductVersion	Character	none
OperatorName	Character	none
RoleName	Character	none
SupportContactInformation	Character	none

VendorName is the name of the product vendor. It is required.

ServerProductName is the name of the server product provided by the vendor. It is required.

ServerProductVersion is the version of the server product. It is required.

OperatorName is the name of the MLS or Association operating the system. It is required.

RoleName is the name of the role restriction where the metadata may be restricted. It is optional.

SupportContactInformation is free text that provides a contact email, phone or website for development support. It is optional.

To the METADATA-SYSTEM Body add:

Field	Content Type	Description
ResourceVersion	resource-version	
ResourceDate	resource-date	
ForeignKeyVersion	foreignkey-version	
ForeignKeyDate	foreignkey-date	
FilterVersion	filter-version	
FilterDate	filter-date	

`resource-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS`

`resource-date ::= RETSDATETIME`

`foreignkey-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS`

`foreignkey-date ::= RETSDATETIME`

`filter-version ::= }}{{{}1*2DIGITS . 1*2DIGITS . 1*5DIGITS`

`filter-date ::= RETSDATETIME`

Impact

The structure of the Login transaction remains the same. The metadata for SYSTEM will have additional information and may have impact.

Compatibility

This change is not backward compatible.

RCP 99 - Mixing StandardNames and SystemNames

Original document:RCP 99 Mixing StandardNames and SystemNames

Note: This RCP Affects the Following Sections:

- 7.3 Required Request Arguments
- 7.4 Optional Request Arguments
- 7.5 Search Response Body Format
- 7.6 Query language

The above sections have been updated or modified since the previous RETS version. Click the links above to go to sections having changes adopted in this version.

Organization:	Falcon Technologies Corp
Name:	Paul Stusiak
Organization:	FBS Data Systems
Name:	Troy Davisson
Version:	1.8.0
Status:	Adopted
Adoption Date:	September 28, 2010

Synopsis

This proposal recommends a clarification to the RETS specification to allow a client to receive both fields with StandardNames and fields without StandardNames in a single response.

Rationale

Currently, RETS clients must choose between sending Search requests with either StandardNames=0 or StandardNames=1. With StandardNames=0, a server will reply with all available fields since all fields must have a StandardName assigned. When a client issues a Search request with StandardNames=1, the specification is conflicted and unclear on exactly what can happen.

Because of the ambiguity, many RETS servers have implemented StandardNames functionality in a strict way that would prevent the mixing of StandardNames and SystemNames in a single response. As a result, if a client is issuing a Search request with StandardNames=1, a server often will only reply with field values for fields that have a StandardName mapped. If a client has a specific use case or business need for a particular field that does NOT have a StandardName mapped, they must revert to using SystemNames and re-do any mappings that might currently be configured. This limitation greatly reduces the effectiveness of StandardNames and renders it useless for many applications that rely on the mappings of a large collection of fields.

Proposal

Under **7.3.1 Search Type and Class**, modify the last paragraph to instead read as:

> *Note that if StandardNames (Section 7.4.7) is set to 1, both the SearchType and Class arguments can be specified using either the SystemName or StandardName. If no StandardName is mapped to a specific Resource or Class, the server MUST accept the Resource and Class parameters by their SystemName even when StandardNames is set to 1.*

Under **7.4.5 Select**, add the following paragraph below the paragraph that begins with “This parameter is used to set the fields....”:

> *In requests where the StandardNames argument is set to 0 and values are given within the Select argument, the client and server MUST reference fields by their SystemName. In requests where the StandardNames argument is set to 1 and values are given within the Select argument, the client SHOULD reference fields by their StandardName when StandardName labels exist but MAY send SystemNames. In the response, the server MUST reference fields by their StandardName when StandardName labels exist and by their SystemNames where no StandardName label exists.*

Under **7.4.7 StandardNames**, modify the last paragraph to instead read as:

> *If this argument is set to 0 (zero) or is not present, the field names passed in the search are the SystemNames, as defined in the metadata. If this argument is set to 1 (one), the client SHOULD reference fields by their StandardName when StandardName labels exist but MAY send SystemNames. In the response, the server MUST reference fields by their StandardName when StandardName labels exist and by their SystemNames where no StandardName label exists. The StandardName designation applies to all names used in the `SearchType`, `Class`, `Query` and `Select` arguments.*

Under **7.6 Search Response Body Format**, directly below the “column-tag” definition, change the descriptive paragraph to instead read as:

> *If a “COMPACT” or “COMPACT-DECODED” format is specified in the request then a “<COLUMNS>” tag is also included containing a delimited list of the names of all of the fields being returned. If the StandardNames argument was set to 0 (zero) or*

> *not provided, these fields MUST be the SystemName reference for every field returned. If the StandardNames argument was set to 1, these fields MUST reference the StandardName label where they exist and the SystemName label when no StandardName label exists. A server MUST NOT return the SystemName for a field that has a StandardName label.*

Under **7.7.2 Query parameter interpretation**, the paragraph directly following the one that starts with “The special value .EMPTY. Is to be interpreted...” should instead read as:

> *Each field MUST be a SystemName, as defined in the metadata, when the StandardNames argument is set to 0 (or not given) in the request. When the StandardNames argument is set to 1, the client SHOULD reference the StandardName when StandardName labels exist but MAY reference SystemNames.*

Impact

For clients that are requesting StandardNames against a server with a strict StandardNames implementation, the proposed change would allow the server to deliver more fields in the response. Servers with this type of implementation must be modified to accept and deliver a hybrid of StandardNames and SystemNames when the optional StandardNames argument is set to 1 in Search requests.

Compatibility

For clients already making compliant StandardNames requests to the server, no compatibility issues exist; however, clients cannot expect a server to handle and deliver responses in the way outlined in this change proposal using prior versions of RETS.

RCP 100 - Alternate Standard Names

Submitter Name	Matthew McGuire
Submitter Organization	CoreLogic Inc.
Submitter Email	mpmcguire@corelogic.com
Co-submitter Name	
Co-submitter Organization	
Co-submitter Email	

Document Name	RETS 1.8.0
Document Version	1.8.0
Date Submitted	2013-05-06
Status	ADOPTED
Status Change Date	2015-04-10

Synopsis

The current RETS specification only allows Standard Names from the REData DTD as last defined for 1.8. This does not permit servers or clients to use the newly adopted Data Dictionary 1.0 standard released by RESO. The new dictionary does not share the same Standard Names so neither server or client vendor can use the names.

Rationale

The following itemizes the rationale for each of the necessary changes.

Expose the standard names and version using GetPayloadList transaction This permits the server to advertise alternate standard name spaces.

Designate the standard names and version shown in Metadata for field information This permits the server to include alternate standard names in the metadata.

Designate the standard names and version used for Search Query argument
This permits the client to request data using alternate standard name spaces.

Designate the standard names and version shown in Search Response formats
This permits the server to return data using alternate standard name spaces.

Extend the Search Payload argument to accept the same as the Format argument
This permits the client to use the Payload argument including the payload version.

RETS Change Proposal-Alternate Standard Names.pdfProposal

Specification Changes

In order to support extensible standard names the following changes need to be applied to the specification.
Add Version to GetPayloadList response data in Section 14.5

The current GetPayloadList transaction permits the server to list the supported payloads but does not provide version information. To supply version information change the GetPayLoadList transaction response columns to the following format:

column-tag	::= <COLUMNS> PayloadName Resource Class Description URI MetadataEntryID Version</COLUMNS> CRLF

Additionally assure that the format has tab delimiters between the column names in the column-tag. This will permit the server to expose the version of the supported payload and associated standard name space. Then add the following text and table below the column-tag table:

The following table defines each column returned by the transaction.

Column Name	Content Type	Description
PayloadName	RETSNAME	Name of the payload
Resource	RETSID	The ResourceID of a resource that supports the payload
Class	RETSNAME	The ClassName of a class that supports the payload
Description	1*256PLAINTEXT	Description of the payload
URI	URI	Valid location of DTD or Schema defining the payload format
MetadataEntryID	RETSID	A value that does not change as long as the semantic meaning of the Payload record does not change
Version	dtd-version | 1*64PLAINTEXT	A string representing the version of the payload. If the payload name is “STANDARD-XML” the version MUST be in dtd-version format as defined in section 7.5 Search Response Body Format

Add New GetMetadata Optional Argument: StandardNames

Following Section 11.1.4 Metadata Format add a new section with the following text: 11.1.5 Metadata StandardNames

In order to support alternate or changing standard names, the server MAY support the optional StandardNames argument for the GetMetadata transaction. This argument declares the name space of the Standard Names shown in the RETS metadata. The server MUST not change other metadata when supporting this parameter. Such that a client requesting ‘StandardNames=Standard-XML: RETS- 20080829.dtd’ will receive metadata containing Standard Names from the Standard-XML RETS-20080829.dtd version of the data standard. The following table defines the representation of the StandardNames argument in BNF form.

StandardNames	::= STANDARD-XML | payload-name “:” version
payload-name	::= RETSNAME
version	::= dtd-version | 1*64PLAINTEXT

Example cases:

If the StandardNames argument is not submitted or empty then return standard names from the default STANDRD-XML format in the StandardName metadata.

If StandardNames=STANDARD-XML is submitted then return standard names from the default STANDRD-XML format in the StandardName metadata.

If StandardNames=STANDARD-XML: RETS-20080829.dtd is submitted then return standard names defined by the default STANDRD-XML format in the StandardName metadata.

If StandardNames= DataDictionary:1.0 is submitted then return standard names defined by the DataDictionary 1.0 payload format in the StandardName metadata.

Edit Description for StandardName in table 11-15

Change the text of the Description for StandardName to the following:

The name of the field as it is known in the XML DTD or Schema designated by the StandardNames argument or the name of the field as it is known in the RETS STANDARD-XML DTD.

Extend Search Optional Argument: StandardNames

To support alternate standard names in DMQL consistently within the Search transaction modify the optional StandardNames argument to support declaring the payload format and version. To do this, change the definition of the argument in Section 7.4.7 StandardNames to the following:

7.4.7 StandardNames

Queries may use either standard names from a declared payload or system names in the query (Section 7.6.) If the client chooses to use

standard names then the client MUST indicate this using the StandardNames argument.
The StandardName designation applies to all names used in the SearchType, Class, Query and Select arguments of the transaction.

If the argument is set to zero or not present, then all names used in the request MUST use the SystemName of the field.

If the argument is set to 1, then all names used in the request MUST use the StandardName of the field as declared by the metadata for the default STANDARD-XML format or use the SystemName of the field as declared by the metadata.

If the argument is set to STANDARD-XML, then all names used in the request MUST use the StandardName of the field as declared by the metadata for the default STANDARD-XML format or use the SystemName of the field as declared by the metadata.

If the argument is set to STANDARD-XML: RETS-20080829.dtd, then all names used in the request MUST use the StandardName of the field as declared by the metadata for the STANDARD-XML at dtd-version RETS-20080829.dtd or use the SystemName of the field as declared by the metadata.

If the argument is set to DataDictionary:1.0, then all names used in the request MUST use the StandardName of the field as declared by the metadata for the DataDictionary payload at version 1.0 or use the SystemName of the field as declared by the metadata.

Extend Search Optional Argument: Format

Format=COMPACT
Return normal COMPACT results

Format=COMPACT-DECODED
Return normal COMPACT-DECODED results

Format=STANDARD-XML

StandardNames	::= 0 | 1 | STANDARD-XML | payload-name “:” version
payload-name	::= RETSNAME
version	::= dtd-version | 1*64PLAINTEXT

The StandardName designation applies to all names used in the SearchType, Class, Query and Select arguments of the transaction.

If the argument is set to zero or not present, then all names used in the request MUST use the SystemName of the field.

If the argument is set to 1, then all names used in the request MUST use the StandardName of the field as declared by the metadata for the default STANDARD-XML format or use the SystemName of the field as declared by the metadata.

If the argument is set to STANDARD-XML, then all names used in the request MUST use the StandardName of the field as declared by the metadata for the default STANDARD-XML format or use the SystemName of the field as declared by the metadata.

If the argument is set to STANDARD-XML: RETS-20080829.dtd, then all names used in the request MUST use the StandardName of the field as declared by the metadata for the STANDARD-XML at dtd-version RETS-20080829.dtd or use the SystemName of the field as declared by the metadata.

If the argument is set to DataDictionary:1.0, then all names used in the request MUST use the StandardName of the field as declared by the metadata for the DataDictionary payload at version 1.0 or use the SystemName of the field as declared by the metadata.

Extend Search Optional Argument: Format

Format=COMPACT
Return normal COMPACT results

Format=COMPACT-DECODED
Return normal COMPACT-DECODED results

Format=STANDARD-XML
Return normal standard xml results

Format=DataDictionary:1.0
The server returns data using the RETS response body format containing a RESO

DataDictionary XML format. This is designated using the extended format of the argument as follows:

Format=[PayloadName][:VERSION]

So that the first part is an identifier of the namespace and the second part is a version for the name space. Note the payload name may match payload names provided by the GetPayloadList Transaction.

Extend Search Optional Argument: Payload

To simplify usage modify the Payload argument to accept the same content as the Format argument.

Payload=DataDictionary:1.0

The server returns data using the RETS response body format containing a RESO

DataDictionary XML format. This is designated using the extended format of the argument as follows:

Payload=[PayloadName][:VERSION]

So that the first part is an identifier of the namespace and the second part is a version for the name space. Note the payload name may match payload names provided by the GetPayloadList Transaction. With this proposal the Payload argument becomes superfluous and may be deprecated in a future version.

Impact

Compatibility

The additional arguments and metadata should only be available for a 1.8 server. If the client uses any additional arguments and declares RETS Version lower than 1.8 then the server MUST return a Search or GetMetadata error as appropriate.

RCP 101 - Child Rows Support

Submitter Name	Ivaan Nazaroff
Submitter Organization	CoreLogic Inc.
Submitter Email	inazaroff@corelogic.com
Co-submitter Name	
Co-submitter Organization	
Co-submitter Email	

Document Name	RETS 1.8
Document Version	1.8.0
Date Submitted	2014-02-18
Status	ADOPTED
Status Change Date	2015-04-10

Synopsis

See the attached document under the tools menu for the original submission, including the diagrams. We are working to resolve the issue with adding images to this form.

This proposal add support for Child Rows to the RETS Update Transaction.

Rationale

By allowing Child Rows to be submitted along with their Parent’s Data RETS will provide a more transactional approach to inserts and updates where logically singular operations performed over more than one class can be submitted in one transaction. A child row goes beyond the notion of a one-to-many relationship. The intent is for Child Rows to describe data that is dependent on its parent. A room is a property of a house and cannot exist without it. It also doesn’t make sense to move a room from one listing to another.

Proposal

Modify the Introduction section of the Update Transaction.

The current introduction to the Update Transaction section doesn’t describe the expected behaviour of a server in the case where a parent record’s data is valid but a child’s is not. The proposal will replace the first paragraph with the following:

Section 10 – Update Transaction

> The update transaction is used to modify data on the server. The client submits information describing the update to perform. The

information is validated by the server. If there are any errors in the primary data or any child row data, the server returns an error reply and no changes are persisted. That is, the server MUST return to a state as if the failed Update Transaction was not attempted. If there are no errors, the record(s) as it was inserted/updated on the server will be returned. The record is returned in the same manner as a record is returned from a search.

Add Child Record support to the Update Transaction Optional Request Arguments

The current Update Transaction only supports data for the primary record. To facilitate adding/updating child rows add the following section to the standard.

10.2.6 ChildRecord

If the Update metadata (11.3.3) describes Child_Action metadata (11.3.5) then the client can submit updated child data with the ChildRecord parameter in the same transaction.

ChildRecord ::= "ChildAction=" ChildActionID field-delimiter "ChildRequestID=" ChildRequestID field-delimiter "Sequence=" ProcessOrder field-delimiter field_name"="field-value *(field-delimiter field_name "=" field-value)

The ChildRecord provides details for child rows of the class being updated. This MUST include the ChildActionID, the ChildRequestID, Sequence and fields that are to be changed. Multiple ChildRecord parameters may be specified both for the same ChildActionID and for different ChildActionIDs if the server describes multiple ChildActions for this UpdateAction. For example, a client can choose to update a room in one ChildRecord, remove a room in a second, and add an Open House with a third ChildRecord parameter, provided the server supports these three ChildActions, in the same Update Transaction.

ChildActionID ::= RETSID

The ChildActionID describes both the relationship between the parent row and this child row and the UpdateAction to apply for this child row. This is the ChildActionID as defined in section 11.3.5.

ChildRequestID ::= RETSID

The ChildRequestID is a means for the client to identify a particular child row to the server. The server will reference this ID when communicating errors, warnings, and success back to the client. These ids SHOULD be unique within a request.

ProcessOrder ::= POSITIVENUM

The ProcessOrder describes to the server in what order child rows are to be processed. The server MUST process child rows in ascending ProcessOrder order. ProcessOrder values need not be contiguous. Coincident ProcessOrder values (i.e. two child rows within the same Update Transaction with the same ProcessOrder) may lead to undetermined behaviour. It is at the server’s discretion to continue processing child rows after a data validation error has been encountered.

field-name ::= RETSNAME

The name of the field to be updated, as specified in the meta-data. This is SystemName as defined in Section 11.3.2.

field-delimiter ::= OCTET

The octet that will separate the fields in the record. This is as defined for Delimiter in Section 10.2.

field-value ::=<varies depending on the field>

The text representation of the field value as defined by the meta-data subject to the business rules.

Add ChildSelect Update Transaction Optional Request Argument

The current Update Transaction only supports describing the current state of the primary record. To facilitate describing child rows or row sets add the following section to the standard.

10.2.7 ChildSelect

ChildSelect ::= “ForeignKey="ForeignKeyID "," field-name *("," field-name)

The ChildSelect specifies a list of fields that should be returned in the response. Server MUST return current (updated) values for all fields in this list. If this argument is omitted, no child sections will be returned. Multiple ChildSelect parameters may be specified for additional child table data. Each ChildSelect parameter MUST specify a different ForeignKeyID.

Add Child Data content to Update Response Body

The current Update Response format only supports data from the primary record. To facilitate describing updated child record data in the Update Transaction responses adjust section 10.5 in the standard to read as follows.

10.5 Update Response Body Format

The body of the update response has the following format when there are no errors:

<RETS1*SP ReplyCode= quoted-reply-code 1*SP

ReplyText= quoted-string *SP > CRLF

[lock-tag]

[lock-key]

[delimiter-tag]

column-tag

compact-data

*(child-data)

[<RETS-STATUS 1*SP ReplyCode=quoted-end-reply-code1*SP

ReplyText=quoted-string *SP/>

</RETS> CRLF

The body of the update response has the following format when there are errors or warnings:

<RETS1*SP ReplyCode=quoted-reply-code1*SP

ReplyText=quoted-string *SP > CRLF

transaction-id-tag

[lock-tag]

[lock-key]

[delimiter-tag]

column-tag

compact-data

[error-block]

[warning-block]

*(child-data)

</RETS> CRLF

child-data ::= open-childdata-tag [column-tag] *(child-row) close-childdata-tag

An element including the child rows of primary rows given a particular Foreign Key.

open-childdata-tag ::= <ChildData ForeignKey="ForeignKeyID">

Opening tag for a Child Data section. The Foreign Key attribute describes the relationship between the rows in this section and the row in the primary table.

close-childdata-tag ::= </ChildData>

Closing tag for a Child Data section.

ForeignKeyID ::= RETSID

Describes the relationship between this child row and the parent row.

child-row ::= open-childrow-tag compact-data [error-block] [warning-block] close-childrow-tag

Contains the updated child row along with related error and warning blocks.

open-childrow-tag ::= <ChildRow ChildRequestID="ChildRequestID">

Opening tag for a Child Row section. The ChildRequestID attribute refers to the ChildRequestID specified by the client in the ChildRecord request argument.

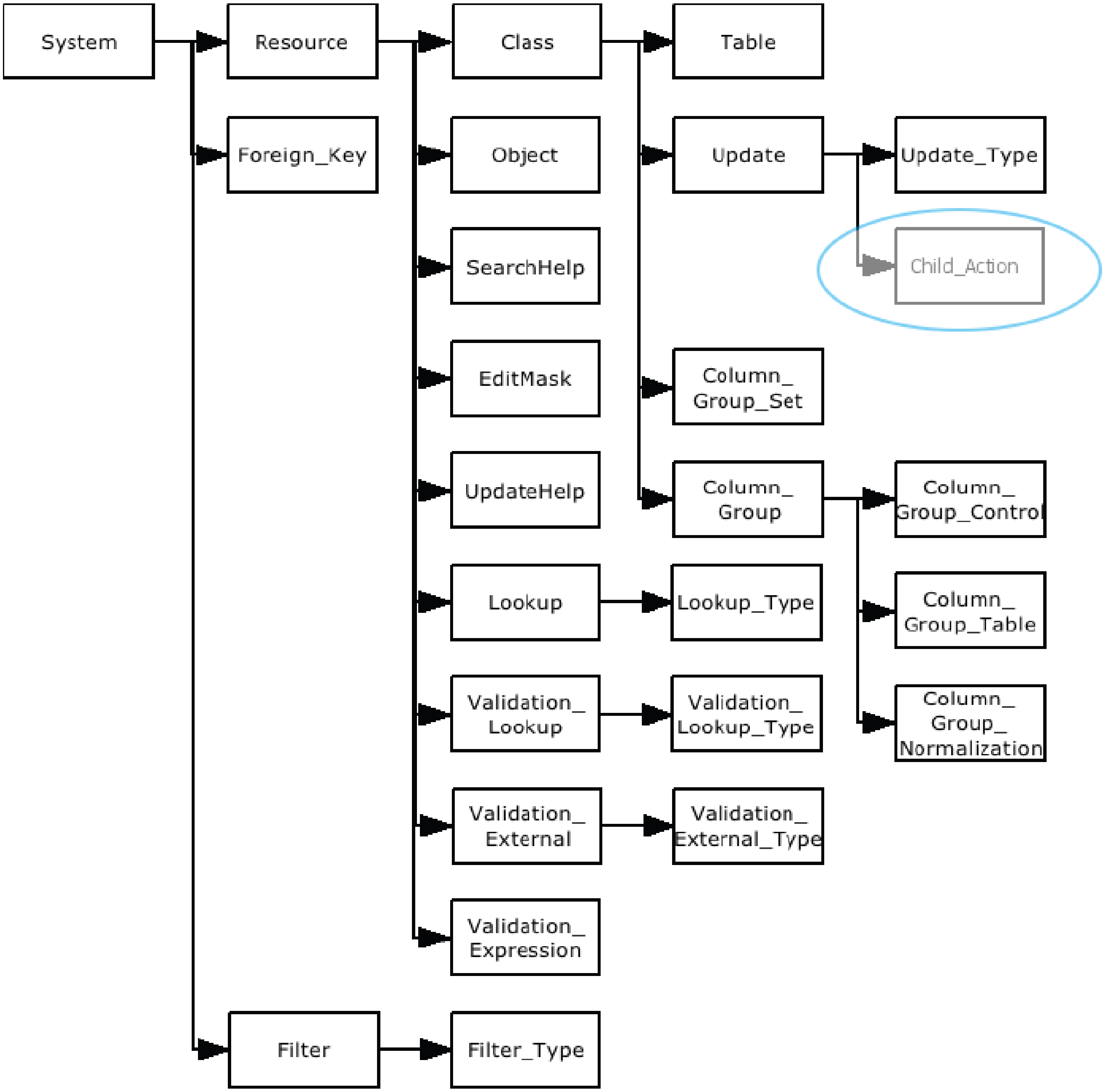
close-childrow-tag ::= </ChildRow>

Closing tag for a Child Row section.

Add Child_Action to the Metadata Structure diagram

The current Metadata Structure diagram needs to grow a Child_Action node as a child of the Update node.

Figure 11.1 – Metadata Structure



Add Child Row related metadata to Metadata-Update

The current Update Metadata does not describe which Update Actions are valid as child rows and which Update Actions MUST be submitted as child rows. To provide a means for servers to communicate these details to clients add the following to Metadata-Update.

11.3.3 Update

Table 11-17 Meta-data Content -Update

Meta-data Field	Content Type	Description

IsChildActionOnly	BOOLEAN	When true indicates this UpdateAction MUST be performed as a child (ChildRecord) of a parent's UpdateAction

The optional IsChildActionOnly flag is used to indicate that this UpdateAction must be performed as a child of a parent's UpdateAction. If not specified the default is false. If true then rows must exist in Child_Action metadata describing valid child actions. If true the client MUST submit this data in a ChildRecord parameter of a parent's update action.

Add Child Metadata-Child_Action section

11.3.5 Child Action

A given update action may have multiple permitted child actions. Child Actions describe actions that can be performed on related classes during an update. The child class is designated by specifying the RETSID of the Foreign Key in the ForeignKeyID column and the child's update action in the Update column.

Table 11-n Child Action Metadata Compact Header Attributes

Attribute	Content
Version	This is the version of the Update Type metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. Every time any contained metadata element changes the version number MUST be increased.
Date	The latest change date of any contained metadata. This MUST be in the format described in chapter 2 for RETSDATETIME.
Resource	The ResourceID for the Resource that this metadata table applies.
Class	The ClassName for the Class to which this metadata table applies
Update	The UpdateAction for the Update that this metadata table applies.

Table 11-m Metadata Content Child Action

Metadata Field	Content Type	Description
ChildActionID	RETSID	The value used to identify this child action for the specified parent's update action. This is the first token in a ChildRecord parameter.
ForeignKeyID	RETSID	The ForeignKeyID (Table 11-8) that describes the relationship between parent and child. The indicated ForeignKey metadata MUST have ParentResourceId and ParentClassId that match the Resource and Class header attributes of this Child Action. The ChildResourceID and ChildClassID then designate the child's Resource and Class. The OneToManyFlag MUST be true.
Update	1*24ALPHANUM	The UpdateAction (Table 11-17) of the child class.

RETS Change Proposal-ChildRows.docx

Impact

Compatibility

The intention is that these additional parameters have no effect in environments where child rows don’t play a role. Servers that do not support child records should not provide the additional metadata. The additional metadata is an extension over previous RETS versions, which should be ignored by clients not supporting this feature. The extended metadata format may break some clients that are not prepared to ignore unknown metadata parts.

RCP 102 - GetObject URL as Default Location

Submitter Name	Mike Wurzer
Submitter Organization	FBS
Submitter Email	mwurzer@flexmls.com
Co-submitter Name	Troy Davisson
Co-submitter Organization	FBS
Co-submitter Email	tdavisson@fbsdata.com

Document Name	GetObject URL as Default Location
Document Version	RETS 1.8.0
Date Submitted	2014-11-23
Status	**ADOPTED**
Status Change Date	2015-04-10

Synopsis

This proposal modifies the expected behaviour of a GetObject request to default to returning a URL for the object.

Rationale

Common practice with static resources on the web is to use content delivery networks or other systems to reduce network bandwidth for objects that change infrequently. These systems default to delivering binary data like images by URL that is then used when the resource is demanded rather than delivering the object as the response. Returning a URL should be the preferred practice because it allows more flexible and efficient delivery of the information.

Proposal

Specification Changes

The following sections detail each area of the existing specification that needs to be changed or clarified and provides reasoning related to each change. Each area of the change will be listed according to the section of the specification using numbering in italics.

This proposal modifies the GetObject Transaction and affects server support.

1. Section 5.4.1 Location

The following paragraph is modified from:

> *If Location is set to "1" the server MAY return a URL to the given object. The default is "0". The server MAY support this functionality (Location=1) but MUST support Location=0. In other words, some servers may store the objects in a database or generate them dynamically. Therefore, it may not be possible for those servers to return a URL to the requested object. In these cases the server MAY choose not to support Location=1. However, all servers MUST support a method to get the object and therefore, MUST support the case where Location=0*

To:

> *If Location is set to "1" the server **should** return a URL to the given object. The default is "**1**". The server **SHOULD** support this functionality (Location=1) but **MAY** support Location=0. In other words, some servers may store the objects in a database or generate them dynamically. Therefore, it may not be possible for those servers to return a **permanent** URL to the requested object. In these cases the server MAY choose not to support Location=1. However, all servers MUST support a method to get the object and therefore, MUST support **either** the case where Location=0 or **Location=1**. Client vendors should be prepared to handle HTTP 410 responses when the URL expires or is removed and should not depend on the URL existing for all time.*

2. Section 5.13 Error Codes

An additional error code will be added.

Reply Code	Meaning
20415	Objects in the response body, Location=0 is not supported. The server does not support retrieving Objects in the response body, only by URL

Compatibility

RETS 1.8.0 and higher.

Document History

Date	Version	Author	Description
October 23, 2014	1	Michael Wurzer	Initial Release
April 9, 2015	2	Paul Stusiak	Clarification and additional changes

Version 1.9.0

Change proposals that were added to the RETS 1.9.0 version:

RCP 103 - Geospatial Search

Submitter Name	Matthew McGuire
Submitter Organization	CoreLogic Inc
Submitter Email	mpmcguire@corelogic.com
Co-submitter Name	
Co-submitter Organization	
Co-submitter Email	

Document Name	Geospatial Search
Document Version	1.8.0
Date Submitted	2015-04-12
Status	**ADOPTED**
Status Change Date	2015-04-12

Synopsis

The current RETS 1.8.0 specification does not provide a means to search geographical areas using geospatial coordinates or shapes.

This proposal resolves this by adding a new field data type to represent a geospatial location. This proposal also adds new standard name to identify a geospatial location field that can be used to search geospatial areas. Finally this proposal details how to search geospatial data using polygon and radial search criteria data.

Rationale

Geospatial Search means searching for properties located in an area specified by geographical coordinates. With the growth of mobile and widespread use of geospatial data in the mobile space, RETS needs a means to represent geospatial data and criteria within metadata and search criteria respectively. This also needs to be discoverable so that data vendors may control access to geospatial functionality. This proposal attempts to resolve these needs by adding a new field data type and interpretation to metadata, adding a new standard name to identify a geospatial location field, and details how to search geospatial data using polygon and radial search criteria data.

Proposal

In order to support geospatial data and search features the following changes are needed in the specification.

New Data Type and Interpretation for geospatial field metadata

In Table 11-15 (Metadata Content – Tables), add following line to the list of Data Types:

Point	Geospatial coordinates represented as a space-delimited pair of decimal numbers, showing first the longitude then the latitude of a point on the Earth’s surface.

In the same table, add following line to the list of Interpretations:

Location	A Point value representing the longitude and latitude of a point on the Earth’s surface.

New Search Criteria DMQL for representing geospatial search criteria

In section 7.6.1 (Query language BNF), modify the definition of field-value by adding an area token:

field-value	::= lookup-list | string-list | range-list | area | period | number | string-literal | .Empty.

In the same section, add following definitions before the definition of ‘period’:

area	::= circle-by-radius | circle-by-points | rectangle | polygon
circle-by-radius	::= map-point “:“ number [length-unit]
circle-by-points	::= map-point “:” map-point
rectangle	::= map-point “, ” map-point
polygon	::= map-point 2*(“, ” map-point)
map-point	::= number “ ” number
length-unit	::= “M” | “K”

In section 7.6.2 Query parameter interpretation add following text above the description of the string field:

> *An* area *value represents an area in which a Point value should be searched for. The area may be shaped as a circle or a polygon.*
>
> *A circular area may be described by its center point and a length of radius, which can be specified in miles (if M is used, default) or kilometers (if K is used). A circular area may also be described by its center and an arbitrary point on the circle’s circumference.*
>
> *A polygonal area is described by a list of points. Each pair of adjacent points defines a side of the polygon. If the last point in the list is not the same as the first one, another side is defined by the first and last point in the list. Any two sides of the polygon MUST NOT intersect. A special case of a polygonal area is a rectangle, which can be defined by just two points where the points MUST be opposite vertices of the rectangle.*
>
> *Client and Server implementations SHOULD use Spherical Mercator for search criteria content and return results using WGS84 if possible.*

This text describes the query functionality for geospatial areas and how to use the search criteria format for all of the supported geospatial search features.

New “Location” Standard Name for indicating a standard geospatial field

In the Real Estate Transaction XML DTD (as mentioned in the Description of the StandardName field in Table 11-15), add a StandardName “Location” to represent the location of the property using the Point type.

Validation Expressions are not supported

This change proposal does not specify any changes in the Validation expression syntax (11.4.9.1). That means, ultimately, that without other changes Point fields cannot be used in the validation expressions.

Impact

<<blank>>

Compatibility

This proposal adds functionality that is only available if the server provides matching metadata. This new metadata may break older clients so the following rules are applied.

Servers MUST NOT present fields of the Point data type to any client with RETS Version 1.8 or lower.

Servers MUST NOT present fields of the Location interpretation to any client with RETS Version 1.8 or lower.

If a server does expose geospatial metadata an older client may not recognize the new data type and therefore may not submit accurate criteria for the advertised geospatial field. In this case the server MUST return an Invalid Search Criteria error to the client.

RCP 104 - StandardValue for Enumerations

Submitter Name	Gregory Lemon
Submitter Organization	RESO
Submitter Email	GregL@reso.org
Co-submitter Name	
Co-submitter Organization	
Co-submitter Email	

Document Name	Real Estate Transaction Specification
Document Version	1.8.1
Date Submitted	2015-06-24
Status	ADOPTED
Status Change Date	2015-07-09

Synopsis

This proposal adds an additional Metadata element for enumerations. StandardValue is a new element for the Metadata LookupType. The value of StandardValue MUST match a value from the Data Dictionary enumerations or MUST be empty.

Rationale

The existing Metadata LookupType does not provide a place for the Data Dictionary enumerations on a RETS 1.x server. An implementation must use a non-standard way to show the value, for example using the enumeration LongValue to match a Data Dictionary defined enumeration.

Adding the field StandardValue will allow existing LongValue, ShortValue, and Value enumeration descriptions to be used for their current defined meaning while providing for the Data Dictionary values.

Proposal

A new element will be added and the LOOKUP_TYPE definition will be modified to include this element.

11.4.3 Lookup Type

A new row will be added to Table 11-27 Metadata Content: Lookup Type as described below.

Field Name	Content Type	Description
StandardValue	1*128PLAINTEXT	The value of the field as it is known in the RESO Data Dictionary. This value SHOULD be used for a COMPACT-DECODED or STANDARD-XML format response when StandardName=1 is used in the RETS Query. If this value is empty, then the LongValue MAY be used in the response. If the system does not support the Data Dictionary, the StandardValue will be the empty string. The following paragraphs describe the behaviour where the system supports the Data Dictionary. If the LOOKUP is not in the Data Dictionary, the StandardValue will be the empty string. If the LOOKUP is in the Data Dictionary, if the LOOKUP_TYPE is not in the Data Dictionary, the StandardValue will be the empty string. If the LOOKUP and the LOOKUP_TYPE are in the Data Dictionary, the StandardValue will match that of the Data Dictionary. A special case exists where the LOOKUP as defined in the Data Dictionary has a 'closed' enumeration. That is, all values of the enumeration are defined in the Data Dictionary and no additional values are permitted. In this case, if the LOOKUP is in the RESO Data Dictionary, the StandardValue MUST match a value in the Data Dictionary and CANNOT be empty. Please consult the appropriate Data Dictionary version to determine the set of 'closed' enumerations.

rets-metadata-content-1_8_1.dtd

```
<!ELEMENT LookupType (MetadataEntryID, LongValue, ShortValue, Value, StandardValue, EXTENSION* , PROPRIETARY*)>


<!ELEMENT StandardValue (#PCDATA)>
```

Compatibility

This change is not compatible with previous versions.

This additional method of describing the enumerations will allow existing RETS 1.x implementations to maintain LongValue,ShortValue, andValue enumeration descriptions. Those who previously consumed the RETS data may choose to use an existing LongValue or the new StandardValue based on their query preferences.

RCP 105 - Update Transaction Response Format Correction

Submitter Name	Paul Stusiak
Submitter Organization	Falcon Technologies Corp
Submitter Email	pstusiak@gmail.com
Co-submitter Name	
Co-submitter Organization	
Co-submitter Email	

Document Name	Real Estate Transaction Standard

Document Version	1.8.1
Date Submitted	2015-07-08
Status	ADOPTED
Status Change Date	2015-07-08

Synopsis

This proposal changes the order of the RETS-STATUS, ERROR block and WARNING block on the Update Transaction response body.

Rationale

The existing DTD for this transaction places the ERROR and WARNING block of responses where there are errors or warning on an update after the RETS-STATUS tag. This is different from any other transaction. Placing the ERROR and WARNING block before the RETS-STATUS will make the Update response match the form of the other transactions.

Proposal

The Update transaction DTD will be modified to re-order the ERROR and WARNING block elements to occur before the RETS-STATUS.

rets-update-1_8_1.dtd

The element RETS will be changed to

```
<!ELEMENT RETS (TRANSACTIONID,DELIMITER?,COLUMNS,DATA*,ERRORBLOCK?,WARNINGBLOCK?,CHILDDATA*,RETS-STATUS?)>
```

Compatibility

This change is not compatible with previous versions. This change affects RETS 1.8.1 and later.

Vendors who have implemented the Update transaction will need to modify their implementation to reorder the response body.

RCP 106 - Client cookie support for RFC 6265

Submitter Name	Steve Martin
Submitter Organization	MRIS
Submitter Email	steve.martin@mris.net
Co-submitter Name	
Co-submitter Organization	
Co-submitter Email	

Document Name	Real Estate Transaction Standard
Document Version	1.8.1
Date Submitted	2015-07-22
Status	ADOPTED
Status Change Date	

Synopsis

The RETS Message Format requires that clients support cookie handling per RFC 2109. This cookie RFC has been superseded twice, by RFC 2965 and most recently by RFC 6265. The language should be adjusted to permit and encourage a client to support RFC 6265.

Rationale

Most client API's provide support for RFC 2109 and now many support RFC 6265. Some implementations of RETS may have an individual requirement for clients to use RFC 6265. This is most commonly seen when a security appliance is in place and requires implementation of the current RFC 6265 for cookie handling. Changing the requirement for clients to implement the most recent cookie RFCs may cause an undo burden on these clients and client API's. It is suggested that the language in Section 3 Message Format is modified to permit clients to support RFC 6265 if an implementation requires it.

Proposal

It is suggested that the language is changed to the following:

A compliant RETS client MUST implement cookie handling as specified in RFC 2109. Although not required, a client may choose to implement the latest cookie RFC 6265 so that it is compatible with servers that may choose to require it.

Impact

This change would provide a suggestion over a required change and would therefore have no impact on any existing RETS clients.

Compatibility

This change proposal is intended to provide full backwards compatibility with previous RETS versions.

RCP 107 - IETF HTTP RFC Updates to references in the RETS specification

Submitter Name	Geoff Rispin
Submitter Organization	Templates 4 Business, Inc.
Submitter Email	grispin@t4bi.com
Co-submitter Name	Paul Stusiak
Co-submitter Organization	Falcon Technologies Corp.
Co-submitter Email	pstusiak@falcontechnologies.com

Document Name	IETF HTTP RFC Updates
Document Version	1.9.0
Date Submitted	2016-05-02
Status	ADOPTED
Status Change Date	2016-08-25

Synopsis

The RESO standards should encourage the adoption of current internet standards at the time of release. The RETS1x Document references obsolete RFC standards documents. This change proposal replaces the references to the obsolete HTTP IETF documents with references to the current IETF RFC standards.

The updated IETF RFC standards include errata, language clarifications, updated security, bug fixes and backwards compatible feature enhancements.

Rationale

The RESO community is building its standards on top of those standards already in use on the internet. The RESO standards should reflect that in our documentation by using those standards that are actively in use. Most RETS implementations that use current external libraries and frameworks may already meet these IETF RFC standards since those external libraries have adopted these IETF standards some time ago. However, to improve interoperability and clarify the RETS document, the RETS document should reference the newer IETF RFC documents so those implementing custom libraries know to update their implementations.

Proposal

Updates all HTTP RFC references in the WebAPI transport document to their latest specification

- HTTP/1.1 RFC 2616 -> RFC 7230-7237
- HTTP/1.1 RFC 2617 -> RFC 2617 + RFC 7235
- HTTP/2 (Not Referenced) -> RFC 7540

Changes:

Changed text is in blue.

Section 3 - Message Format

3.1 General Message Format

3.1.1 RETS HTTP Encapsulation

RETS messages are encapsulated as the bodies of HTTP/1.1 and optionally newer requests and responses. The request body may be null, depending on the request. The response body is never null

Note that, per the HTTP Specifications [2], [23] , keywords in header key-value pairs are not case-sensitive. The values, however, may be case-sensitive depending on context.

3.2 Request Format

A RETS request is either an HTTP GET request or an HTTP POST request. In the case of the GET-request the Argument-List is appended to the Request-URI after a delimiting question mark ("?"). For the post-request the Argument-List is sent as the first entity body for the POST method.

get-request	::= **GET** *Request-URI* [**?** *Argument-List*] *HTTP-Version CRLF* **message-header* *CRLF*
post-request	::= **POST** *Request-URI HTTP-Version CRLF* **message-header* *CRLF* [*Argument-List*]

The *Request-URI*, *HTTP-Version* and *message-header* are defined in HTTP Specifications [2], [23] . The detailed construction of the *Argument-List* is defined in HTML 4.01.

3.3 Required Client Request Header Fields

The HTTP header of any messages sent from the client MUST contain the following header fields:

User-Agent	This header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations, as well as providing enhanced capabilities to some user-agents. All client requests MUST include this field. This is a standard HTTP header field as defined in HTTP Specifications [2], [23] .
User-Agent	::= **User-Agent:** 1* *product*
product	::= *token* [**/** *product-version*]
product-version	::= *token*

Example: `User-Agent: CMAZilla/4.00`

Product tokens should be short and to the point: use of them for advertising or other non-essential information is explicitly forbidden. Although any token character may appear in a product-version, this token SHOULD only be used for a version identifier (i.e., successive versions of the same product SHOULD only differ in the product-version portion of the product value). For more information about User-Agent see HTTP Specifications [2], [23] .

A server MAY advertise additional capabilities based on the client's User-Agent, and MAY refuse to proceed with the authorization if an acceptable User-Agent has not been supplied. A server MAY also choose to authenticate the client's identity cryptographically using the RETS-UA-Authorization header; see section 3.4 for additional information.

`RETS-Version`	*::= 1*DIGIT "." 1*DIGIT "." 1*DIGIT* The client MUST send the RETS-Version.The use of the *RETS-Version*as a maker for versioning metadata may require additional digits to correctly represent the version of metadata. Specifically, implementers should be permissive in the use of *RETS-Version* and should accept values where there are more than a single digit for the release or minor positions.The convention used is a numbering scheme similar to the HTTP Version in HTTP Specifications [2], [23] . The version of a RETS message is indicated by a RETS-Version field in the header of the message.
`Cookie`	The client MUST implement cookie handling as specified in RFC 2109. Servers and clients are encouraged to implement cookie handling as specified in RFC 6265 [15] and MAY implement cookie handling as specified in RFC 6265. Future versions of this standard will obsolete the use of 2109 as permitted. If any server response has included a valid Set-Cookie header, and the cookie in that header has not expired, the client MUST return the corresponding Cookie header. See RFC 2109, 6265 [15] for the full specification.

3.4 Optional Client Request Header Fields

`Authorization`	Authorization header field as defined in the HTTP Specifications [2], [23] . See 4.1, "Security", as well as the HTTP Specifications [2], [23] , for additional information.
`RETS-Request-ID`	A character string of printable characters which the client can use to identify this request. The contents are implementation-defined. If this field is included in a request from the client then the server MUST return it in the response.
`RETS-Request-ID`	*`::= 1*64ALPHANUM`*
`Accept-Encoding`	A comma-separated list of MIME types indicating the content encoding schemes that the client is willing to accept. This is intended to support the use of compression in data returns; see section 3.8 for additional information.
`Accept-Encoding`	*`::= 1*64ALPHANUM/1*64ALPHANUM *[,1*64ALPHANUM/1*64ALPHANUM...]`*
`RETS-UA-Authorization`	A client MAY support authentication of its User-Agent value by including the RETS-UA-Authorization header. Servers MAY require this header with a valid value before providing services.
`RETS-UA-Authorization`	`::= ua-method ua-digest-response`
`ua-method`	`::=` **`Digest`**
`ua-digest-response`	`::= "*LHEX "` See section 3.10 for the method of computing the `ua-digest-response` value. The client MAY send this header under any circumstances. It need not send this header if the server has not indicated that it requires user-agent authentication by responding to a transaction with a RETS error code of 20037.

In addition to the header fields listed here, the client may send any header compliant with the HTTP Specification [2], [23] in use.

3.5 Response Format

The general server response to a request is either a well-formed XML document returning RETS-encapsulated data or error information, or, for the `Get` transaction and for successful `GetObject` transactions, the content of the requested object in the format given in the response's HTTP Content-Type header. Note that this is an ordinary HTTP response per RFCs [2],[23].

The more common HTTP *`Status-Codes`* are provided in Section 3.9 , though any status code defined in RFC [2],[23] is permissible. Servers MUST use appropriate predefined status codes when communicating with the client.

3.7 Optional Server Response Header Fields

Example: `RETS-Server: AcmeRETS/1.0`

`Set-Cookie`	The server MAY use HTTP cookies to maintain state information. See RFC 6265 [15] for the format of the `Set-Cookie` header. A cookie having a name of `RETS-Session-ID`defines the RETS session ID, which is used in calculating the RETS User-Agent Authentication (section 3.10). Cookies with other names have no special meaning in RETS but MAY be used when necessary.

In addition to the header fields listed here, the server may send any header compliant with HTTP Specifications [2],[23].

3.9 General Status Codes

Any of the following status codes (in addition to the others provided in the HTTP Specifications [2], [23]) may be returned by a server in response to any request:

4.1 Security

4.1.1 User Authentication

While this specification does not require the use of security — it is permissible, for example, to operate a publicly-accessible RETS server — most operators of RETS servers will wish to authenticate users. A server that requires that users be authenticated MAY implement RFC 7235 [2] , HTTP Authentication. The use of at least digest authentication is strongly recommended.

4.1.2 Client Authentication

Client authentication may be performed through the use of the optional RETS-UA-Authorization header (section 3.4) . Prior versions of this specification used a specially-calculated cnonce value in the Authorization header to implement this function. A server implementing this version of the RETS specification MUST accept the RETS-UA-Authorization header for client authentication. It MAY accept RFC 7235 [2] style authentication as in prior versions of the RETS specification.

Section 8 - Get Transaction

Gets an arbitrary file from the server or performs an arbitrary action, specified by URI. This is a standard HTTP GET, per the HTTP Specifications [2], [23] . The file to get is passed as part of the Request-URI.
RETS servers need not support the GET transaction to any greater extent than is necessary to implement the functionality of the Action URL (see 4.10, "Capability URL List"). If a RETS server does not intend to include an Action URL in its login responses, it need not support the GET transaction.

Section 19 - References

Number	Reference
[1]	Braden, R., "Requirements for Internet Hosts — Communication Layers" STD 3, RFC 1123, IETF 1989.
[2]	R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014 R. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, June 2014 J. Reschke, "Initial Hypertext Transfer Protocol (HTTP) Authentication Scheme Registrations", RFC 7236, June 2014 J. Reschke, "Initial Hypertext Transfer Protocol (HTTP) Method Registrations", RFC 7237, June 2014
[3]	Rivest, R., "The MD5 Message Authentication Algorithm", RFC 1321, April 1992
[4]	Crocker, D., "Standard for ARPA Internet Text Messages", RFC 2822, IETF 2001

...

[14]	Kaliski, "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, IETF 1998
[15]	A. Barth, "HTTP State Management Mechanism", RFC 6265, April 2011
[16]	W3C, "HTML 4.01 Specification", W3C Recommendation 24 December 1999 (http://www.w3.org/TR/html401/)

...

[21]	Klyne, G. and Newman, C., "Date and Time on the Internet: Timestamps", RFC 3339, IETF 2002
[22]	Crocker, D. and Overell, P., "Augmented BNF for Syntax Specification: ABNF", RFC 2234, IETF 1997
[23]	M. Belshe, "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, May 2015

Impact

The impact should be minimal since most implementers are using common third party libraries and frameworks to implement the HTTP transport within their software. Implementations that have custom HTTP libraries will need to update their libraries to implement the current IETF RFC documents noted in this change proposal.

The HTTP/2 includes a separate handshake method that will not impact those that have not implemented it and downgrade gracefully.

Compatibility

The new IETF RFCs are backwards compatible with the old ones except where contradictions or ambiguity exists in the older RFCs. This change is not intended to be compatible with previous versions of the RETS standard, however, many implementations of previous versions may work with the updated IETF RFC documents and libraries.

RCP 108 - Migrate RETS specific HTTP Headers to HTTP User Space headers (X-*)

Submitter Name	Geoff Rispin
Submitter Organization	Templates 4 Business, Inc.
Submitter Email	grispin@t4bi.com
Co-submitter Name	Paul Stusiak
Co-submitter Organization	Falcon Technologies Corp
Co-submitter Email	pstusiak@falcontechnologies.com

Document Name	HTTP User Space Headers
Document Version	1.9
Date Submitted	2016-05-02
Status	ADOPTED
Status Change Date	2016-08-01

Synopsis

The RETS specific headers should be changed to the use the prefix "X-" to conform to industry/Internet best practices and make the adoption of security appliances easier for implementators.

Rationale

The current RETS specific headers are in the root name space within the HTTP header space. While this is allowed by the IETF RFC specifications, it has become accepted practice that all headers not explicitly defined in the HTTP RFC specifications should start with a "X-" to allow for the future expansion of the protocol and to keep the root namespace under the control of the IETF.

Many of the vendors of deep packet inspection solutions (Application Firewalls, denial of service prevention, etc.) follow this accepted practice and enforce only RFC defined HTTP root name space defined headers and only permit application specific headers that conform to the X-* naming convention be allowed to pass through the security appliance. The RETS 1X custom headers make adoption of these types of solutions more complex as exception rules must be written or security threshold decreased to allow RETS traffic to pass through the security appliance if that appliance supports exception rules.

Proposal

Prefix all RETS specific headers with the "X-" string so that they are in the HTTP Header user name space.

The following sections have replacement text as shown.

3.3 Required Client Request Header Fields

X-RETS-Version	*::= 1*DIGIT "." 1*DIGIT "." 1*DIGIT* The client MUST send the X-RETS-Version.The use of the *X-RETS-Version* as a maker for versioning metadata may require additional digits to correctly represent the version of metadata. Specifically, implementers should be permissive in the use of *X-RETS-Version* and should accept values where there are more than a single digit for the release or minor positions.The convention used is a numbering scheme similar to the HTTP Version in Section 3.1 of RFC 2616. The version of a RETS message is indicated by a X-RETS-Version field in the header of the message.

3.4 Optional Client Request Header Fields

X-RETS-Request-ID	A character string of printable characters which the client can use to identify this request. The contents are implementation-defined. If this field is included in a request from the client then the server MUST return it in the response.
X-RETS-Request-ID	::= *1*64ALPHANUM*

X-RETS-UA-Authorization	A client MAY support authentication of its User-Agent value by including the X-RETS-UA-Authorization header. Servers MAY require this header with a valid value before providing services.
X-RETS-UA-Authorization	::= ua-method ua-digest-response

3.10 Computing the RETS-UA-Authorization Value

The RETS User Agent Authorization digest response value is used in the `X-RETS-UA-Authorization` header specified in section 3.4 . It is computed as follows:

a1	::= MD5(*product* ":" *UserAgent-Password*)
ua-digest-response	::= LHEX(MD5(LHEX(a1)":” *X-RETS-Request-ID* ":" *session-id* ":" *version-info*))

where:

product	The first *product* value taken from the User-Agent header (section 3.3). Note that the *product* value consists of both the product token and version.
UserAgent-Password	::=*TOKEN* This value is a secret shared between the client and server.
X-RETS-Request-ID	::=*X-RETS-Request-ID* This value MUST be the same as that sent with the `X-RETS-Request-ID` header. If the client does not use the X-RETS-Request-ID header, this token is empty in the calculation.
session-id	::= If the server has sent a `Set-Cookie` header with a cookie name of `RETS-Session-ID`, `session-id` is the value of that cookie. If the server has not sent a cookie with that name, or if the cookie by that name has expired, this token is empty in the calculation.
version-info	::= The value of the `X-RETS-Version` header sent by the client with this transaction.

13.1 Required Request Header Fields

X-UpdateAction	::= *1*24ALPHANUM*	
	Add	Add a new Object
	Replace	Change an existing Object
	Delete	Delete an existing Object

X-Resource	::= *RETSID*	The ResourceID for the Resource that this Object belongs as defined in Table 11-6 Metadata: Resource Description Fields

X-ResourceID	<value from the KeyField>	A value from the KeyField of the Resource for which the Object is to be uploaded.
X-ObjectID	::= `1*5DIGIT`	
X-OrderHint	::= `1*5DIGIT`	See Section 13.2 for further information.
X-UID	::= `TOKEN`	A string identifying the single Object being posted

Section 13.2 Optional Request Headers

13.2.1 OrderHint

`X-OrderHint`	::= *1*5DIGIT*

Impact

Both Clients and Servers would need to be updated to accept the new headers. The change would be a small code change but it would require a release and the client and server must run the same version using the "X-" headers.

Compatibility

This change is not backwards compatible. The header name change will limit this to forward versions of RETS1 starting with RETS1.9.

RCP 109 - Update TLS specification references to current Internet/Industry Standards

Submitter Name	Geoff Rispin
Submitter Organization	Templates 4 Business, Inc.
Submitter Email	grispin@t4bi.com
Co-submitter Name	Steve Ledwith
Co-submitter Organization	Black Knight Financial Services
Co-submitter Email	stephen.ledwith@bkfs.com
Co-submitter Name	Paul Stusiak
Co-submitter Organization	Falcon Technologies Corp.
Co-submitter Email	pstusiak@falcontechnologies.com

Document Name	Update TLS references to current IETF RFC
Document Version	1.9.0
Date Submitted	2016-05-02
Status	ADOPTED
Status Change Date	2016-08-01

Synopsis

The RESO RETS specification should enforced the use of the current internet/industry standards and best practices for cryptography. The current specification contains the HTTP over TLS specification, RFC2818 (superseded by RFC7230) but does not include a specification for the implementation of TLS or any best practices around implementation or operation. Using obsolete standards and libraries presents complications with interoperability as cryptography is usually not backward compatible due to vulnerabilities in ciphers, implementations and algorithms found over time. Also as computational power increases, this weakens the strength of cryptography and larger keys or new cryptographic algorithms must be used to protect data.

Rationale

With the large number of vulnerabilities and weaknesses found in SSL and TLS that underpin the HTTPS standard, its very easy to have traffic sent encrypted but be completely un-secure as the transport, algorithm or implementation has been compromised. These announcements happen regularly for example, POODLE, HEARTBLEED, BEAST, etc.

There needs to be a common base line for implementors to target. Without a standard someone could use the SSLv2 standard and meet the letter of the implementation but no up-to-date client would be able to talk to that solution since SSLv2 has been compromised for over 10 years. The TLS/SSL specifications can create communication problems as the handshake is a lowest common denominator approach and will choose the highest transport and security of the lower of the two parties in the conversation. This can default to poor choices for protection or communications being rejected as no common approved ciphers exist.

For security minded implementations or those organizations with other security standards to meet, it is common to have a hardened security policy in place to disable all ciphers and transports that are deemed compromised or weak. This prevents communication from those that do not meet the standard but also protects against data compromise as well.

Proposal

Add the current standard cryptographic RFCs for HTTPS communication to the specification.

- TLS 1.2 -> RFC5246 https://www.ietf.org/rfc/rfc5246.txt

Add a references to security vendor best practices for the use of Encryption

- RFC 7525 Best Practices - https://www.rfc-editor.org/rfc/rfc7525.txt
- OWASP TLS implementation guide - https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
- https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices.pdf

Include security in the certification process. This step should be a requirement of validating a secure OAuth2 communication for the current specification as strong security is a requirement of the underlying authentication protocol. Extending it to the Transport protocol make sense Users get a false sense of security when they see a secure connection but the implementation is poor or flawed.

- Server test (free service) - https://www.ssllabs.com/ssltest/

The following section changes are recommended:

4.1.3 Data Security

The need for secure HTTP transactions cannot be met by authentication schemes. Vendors MAY require authentication or all transport to be secured. For those needs, HTTP-over-TLS (commonly known as HTTPS) is a more appropriate protocol[23]. A compliant server MAY support only HTTP-over-TLS. In this case, the server SHOULD listen on port 12109 rather than the standard RETS port, 6103.

16.1 Connection Establishment

A client initiates communication with a server by beginning a TCP connection on any mutually agreed TCP port, with the default being 6103 for unencrypted connections, and port 12109 for TLS-encrypted connections[23]. When the TCP connection has entered the Established state, the session proceeds to the start of the Authorization phase.

16.3 Session

Once the Authorization phase has been completed, both endpoints enter the Session phase. During the Session phase, clients may issue any combination of requests for which they are authorized. The first of these MUST be to issue a GET requests for the "Action" URL, if any, included in the Login response (Section 4.10). After this, clients may issue other transactions. Note that the transport level security requirements will determine the protocol (HTTP/HTTPS), the port (6103/12109) and the version of the protocol for a specific implementation. Vendors using transport level security SHOULD use the most current version of the transport standard and the supporting libraries to minimize security risks.

Section 19 - References

[23]	Dierks, T. "The Transport Layer Security (TLS) Protocol." *The Internet Engineering Task Force*. Internet Society. RFC 5246, IETF 2008 <https://www.ietf.org/rfc/rfc5246.txt>. **NOTE**: This RFC may be obsoleted by the IETF. Implementations where this standard applies (HTTP-over-TLS) MUST use the most recent Standards Track version of the IETF standard.

Impact

Depending on the age of the library, the impact should be minimal for those implementors tracking the common TLS libraries and frameworks. That is, those implementors who regularly rebuild their client or server using current libraries or frameworks. For implementors who have implemented a custom TLS stack, the library will need to be updated to the current standard.

All of the major software implementations (Mozilla NSS, Java SSE, OpenSSL, Microsoft SChannel/CryptAPI, GnuTLS, etc.) provide regular security updates for all issues and implement the latest feature sets. For those that are using out-of-date or un-patched software or custom cryptographic implementations, they will have more work involved to bring them up to a reliable state but their security is already compromised and they should be tracking the updates to their security libraries in their services.

There may be many clients using older libraries that have not been updated they should be encourage to move to non-encrypted communications or update as their implementation are providing no security benefit and making the providers less secure.

Implementations will still need validate their configurations secure their systems properly. The cryptographic software does allow for weak configurations as there are places that they are still valid. This is where the best practices references can help with giving the specification a target for MUST implement and SHOULD implement behaviours.

Compatibility

For those solutions that are maintaining patch levels of their software and frameworks, there is little or no impact.

The compatibility problem come from unmaintained solutions and the frequent security updates that happen. Due to the nature of cryptographic vulnerabilities and their announcement, those using un-secure methods would be broken by strict enforcement but those same solutions are have a false sense of security and are gaining no crpytograhic benefits in their implementation. This includes such software as librets.

RCP 110 - Deprecate and Replace GetPayloadList

Submitter Name	Matthew McGuire
Submitter Organization	CoreLogic Inc.
Submitter Email	mpmcguire@corelogic.com
Co-submitter Name	
Co-submitter Organization	
Co-submitter Email	

Document Name	Deprecate and Replace GetPayloadList
Document Version	1.9.0
Date Submitted	2016-05-28
Status	ADOPTED
Status Change Date	2016-08-25

Synopsis

This proposal removes the GetPayloadList transaction in favor of adding Payload information to the RETS Metadata as a shared metadata element.

Rationale

The GetPayloadList Transaction currently returns information that mimics the RETS metadata structure and describes potential metadata that the server supports. Rather than include a separate Transaction for this information, this proposal adds the Payload information to the RETS metadata as shared content so that RETS clients can discover and use it from the overall system metadata model.

Proposal

Add Payloads as a shared element below System metadata after Filter metadata in the metadata hierarchy diagram.

Add section 11.2.6 Metadata Payloads and move content from section 14.5 Payload Response Format to this new section. Reformat the response content to match the general RETS metadata structure.

Impact

Removes GetPayload Transaction in lieu of shared payload metadata.

Compatibility

This proposal explicitly moves content from one transaction to another so it is not backward compatible with previous versions.

RCP 112 - RETS Metadata Version Header

Submitter Name	Matthew McGuire
Submitter Organization	CoreLogic Inc.
Submitter Email	mpmcguire@corelogic.com
Co-submitter Name	
Co-submitter Organization	
Co-submitter Email	

Document Name	RETS Metadata Version Header
Document Version	1.9.0
Date Submitted	2016-05-28
Status	ADOPTED
Status Change Date	2016-08-01

Synopsis

This proposal provides an additional HTTP header to communicate the current RETS metadata identity and version for client applications.

Rationale

Current RETS clients must use the GetMetadata transaction in order to discover the current metadata identity and version that applies to the connection. For client systems that serve multiple users or groups, this is inefficient as it requires additional requests of GetMetadata to maintain the correct metadata version in use. Adding a RETS metadata version header to the general response header allows the client to immediately detect changes to metadata and act accordingly for the system. This removes the need to request GetMetadata repeatedly and reduces overall transaction costs for both the RETS Client and Server.

Proposal

3.1 Specification Changes

In Section 3.7 Optional Server Response Header Fields add the following optional header definition.

X-RETS-MetadataVersion ::= Identity:1*128ALPHANUM;Version:*rets-version-type*;

Add the following text to define the usage of the header:

The X-RETS-MetadataVersion header permits the server to communicate to the client what metadata identity and version is applied to the response of the transaction. This is primarily intended to communicate metadata change to the client in an immediately discoverable manner.

Impact

None

Compatibility

This is an additional header with no impact on the existing standard, therefore there are no backward compatibility concerns.

RCP 113 - Search Transaction Optional Format Argument - Add JSON

Submitter Name	Paul Stusiak
Submitter Organization	Falcon Technologies Corp
Submitter Email	pstusiak@falcontechnologies.com
Co-submitter Name	
Co-submitter Organization	
Co-submitter Email	

Document Name	RETS Standard
Document Version	1.9.0
Date Submitted	2016-05-31
Status	ADOPTED
Status Change Date	2016-08-25

Synopsis

This change proposal adds an optional format type to the optional FORMAT search parameter.

Rationale

The Data Dictionary workgroup supports the JSON format as a work item. Those systems that would like to use this work have no way to express this format in the existing standard

Proposal

Section 7.4.2 will have the additional argument values added of

```
Format  ::= COMPACT | COMPACT-DECODED | STANDARD-XML |
            STANDARD-XML:dtd-version | JSON | JSON:dd-version | JSON-DECODED | JSON-DECODED:dd-version | payload-name | payload-name":"payload-version
```

And the body text will read

"COMPACT" means a field list <COLUMNS> followed by a delimited set of the data fields <DATA>.

"COMPACT-DECODED" is the same as COMPACT except the data for any field with an interpretation of Lookup, LookupMulti, LookupBitString or LookupBitMask, is returned in a fully-decoded format using the LongValue. See Section 13 for more information on the COMPACT formats and section 11.4.3 for more information on the Lookup types.

"STANDARD-XML" means an XML presentation of the data in the format defined by the RETS Data XML DTD. Optionally, a dtd-version may be added to the literal "STANDARD-XML:" to indicate a specific version of the DTD.

"JSON" means a JSON presentation of the data in the format defined by the METADATA-TABLE information. This allows legacy systems to represent their metadata as JSON. The intent of the change is to allow the Data Dictionary *to be used within existing RETS1 implementations.* Optionally, *where the implementation supports multiple versions of the data dictionary, a version may be added to the literal "JSON:" to indicate a specific version of the data dictionary.*

A Server that supports Data Dictionary names does not have to support all valid Data Dictionary payload-name values for a specific payload-version.

Servers MUST support COMPACT, COMPACT-DECODED and Data Dictionary formats. A Server MAY support STANDARD-XML. A Server MAY support JSON, JSON-DECODED, Payloads, data dictionary versions or any valid combination of them. If the format is not specified, the server MUST return the Data Dictionary response body for the Resource/Class combination.

Servers MUST support COMPACT, COMPACT-DECODED and Data Dictionary formats. A Server MAY support STANDARD-XML.

Example: `Format=COMPACT-DECODED`

If the client requests STANDARD-XML, it MAY also append a preferred DTD version. Where the server supports STANDARD-XML, the server MUST support the current version and SHOULD additionally support at least the prior version.

Example: `Format=STANDARD-XML:1.0`

If the client requests a payload from the Data Dictionary, if MAY also append a preferred version of the Data Dictionary. Where the server supports Data Dictionary names, the server MUST support the current version and SHOULD additionally support at least the prior version.

A Client MUST only request the Format argument, and optionally the Select argument OR the Payload argument.

A Server MUST return an error code 20216 Invalid Argument Combination, Format and Payload when a Client submits a request with both Format and Payload optional arguments.

A Server MUST return an error code 20219 Invalid FORMAT when a Client submits a request with an optional format that is not supported on this Server.

7.7 Reply Codes

Table 7-1 Search Transaction Reply Codes

20219	Invalid FORMAT - The selected optional Format argument is not supported by this implementation.

20220	Requested Data Dictionary version unavailable - The selected optional Format argument data dictionary version is not supported by this implementation.

Impact

None.

Compatibility

RETS 1.9