

# Agenda

<b>Date</b>	<b>Time</b>	<b>Discussion</b>	
June 18	1:00 PM	Introduction/Agenda review	
	1:10 PM	RETS Working Group status (Dale Stinton)	
	1:25 PM	Governance issues: presentation and discussion	
	2:00 PM	Standards release schedule discussion	
	2:30 PM	Errata review and resolution	
	3:00 PM	Break	
	3:15 PM	Change proposal review and discussion	
	4:45 PM	Wrap-up and review/revisit 6/19 agenda	
	June 19	8:30 AM	Change proposal review
		9:30 AM	Change proposal version assignment and scheduling
10:00 AM		Break	
10:15 AM		RETS 2.0 directions – introduction	
10:30 AM		Demonstration	
11:00 AM		RETS 2.0 discussion	
12:00 PM		Lunch	
12:30 PM		XML directions: XML schema, or...	
1:00 PM		Strategies for developing a more comprehensive DTD/schema	
1:45 PM		Wrap-up	

## Governance Notes

- Participation open to anyone who wants to participate (and who is willing to pay the travel expenses and a nominal meeting fee).
- All meeting attendees may vote.
- Semiannual meetings proposed.
- Consultation via email list and, when necessary, conference calls.
- Change proposals require at least 80% “yes” votes at the meeting. Objective is consensus.
- Modeled on IETF, ANSI.
- NAR Board will endorse this process and the changes it produces.
- Endorsement is revocable.
- NAR participates as do other members.

## Submitted Errata

### 1. Undefined XML return format for "History" Search

**Reporter:** michael.delgudio@mrisc.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 7

**Description:**

The RETS Standard documentation does not provide information describing the appropriate information that should be returned when conducting a "History" Search as defined in Section 7. Specifically the Standard is silent to the XML structure that should be returned. The Standard provides no guidance as to what constitutes History -- either by listing number, property address or tax ID number. Nor, does it indicate what type of historical information one may expect when conducting a "History" Search.

**Proposed Solution:** a solution to the errata (as appropriate)

### 2. RETS Standard / Web Site Missing the RETS-mmddyyy.dtd document

**Reporter:** michael.delgudio@mrisc.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 7

**Description:**

Section 7 of the RETS standard refers to a "RETS-mmddyyy.dtd" document that should be used to validate STANDARD-XML data. However, the dtd supplied by RETS-wg is "REData-mmddyyy.dtd", and is missing certain elements required by section 7.

**Proposed Solution:** MRIS has, and is using, a revised DTD containing missing tags, to be compliant with section 7.

### 3. Standard Names for Search Transactions (Other than property)

**Reporter:** michael.delgudio@mrisc.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 4 of DTD, Section 7 of Standard

**Description:**

The RETS Standard provides a list of standard names for the Property Search type in the Search transaction. However it does not supply Standard Names for the remaining Search types. For an early release of our system we created standard names for the Agent, ActiveAgent and Office Search types.

**4. Class for Class-less searches?**

**Reporter:** michael.delgudio@mris.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 7.3.1

**Description:**

The Standard requires that the "Class" attribute be passed in a Search transaction (Section 7.3.1). However several searches are "class-less". Particularly, Agent, ActiveAgent, Office (and potentially Prospect). The standard is silent on the appropriate value for class in these circumstances.

**Proposed Solution:** One offered solution, if class were to remain mandatory, that the class attribute contain the same value as the Search Type. I.e. both SearchType and Class might be ActiveAgent, or Office.

**5. REAgent tag missing brokerage affiliation info**

**Reporter:** michael.delgudio@mris.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 7.3.1

**Description:**

The REAgent tag suffers greatly in that there is no way to associate the Agent with his or her affiliated offices. In fact, the standard is vague on what address information belongs in the agent tag. Should it be the agent's Home Address? Office Address?

**Proposed Solution:** One interpretation: since Agents may be affiliated with many offices that the appropriate address may be the agent's Home Address. In that case, there needs to be a way to associate this agent with his or her offices to also display office address.

One solution may be to add an affiliations list to the REAgent structure:

```
<affiliations>
  <affiliation>
    <officeid>
```

```
<brokerid>  
</affiliation>  
<affiliation>  
<officeid>  
<brokerid>  
</affiliation>  
</affiliations>
```

## 6. Count Tag should contain a quoted value

**Reporter:** michael.delgudio@mrisc.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 7.6

### Description:

The Standard and XML validation rules are in conflict in Section 7.6, specifically the COUNT tag. The Standard indicates that the count tag should be represented as follows:

```
<COUNT Records=123/>
```

However XML requires that attributes be quoted as in:

```
<COUNT Records="123"/>
```

**Proposed Solution:** The Standard document, and DTD should be updated to reflect this. Currently, COUNT does not appear in the RETS-wg supplied DTD. The tag should be added to the DTD.

## 7. Special Capability URLs are bound to specific front-ends

**Reporter:** michael.delgudio@mrisc.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 4.13

### Description:

Section 4.13 requires that any special function created be bound to one or more specific versions of specific front end software. This requires the server to be aware of every version of every front-end that access the system, and special functions will not be available to new front-ends until they are registered with the server.

**Proposed Solution:** The MUST in this section should be changed to MAY, and the server given the ability to offer special functions to ALL front ends that may access the system. For example, in a

given system system there may be extended capability URLs available to all user agents. These functions could be extensions to an API beyond those offered by RETS.

### 8. Section 11.4.3 “Lookup Type” Error

**Reporter:** michael.delgudio@mrisc.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 11.4.3

#### **Description:**

In the Standard Document, in section 11.4.3 Lookup Type, there is an Attribute defined as follows:

```
lookup-name ::= 1*32ALPHANUM
```

However this attribute does not appear in the Definition immediately aboveit.

**Proposed Solution:** The Standards Doc should be revised as follows:

```
<METADATA-LOOKUP_TYPE Resource="resource-id"  
                        Lookup="lookup_name"  
                        Version="lookup-type-version"  
                        Date="lookup-type-date">
```

[ETC...]

The attribute Lookup was selected to remain consistent with the Sample compact reply in section 11.4.3

### 9. Purpose / Intention of REOfficeRosters and REOfficeRoster Tags

**Reporter:** michael.delgudio@mrisc.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 7

#### **Description:**

The standard is silent on the data to be returned by Agent, ActiveAgent andOffice searches, and is also silent on the purpose of certain structures within the XML data.

It is unclear the intended Purpose of the REOfficeRosters tag. It is our assumption that the Standard intends for an Agent/ActiveAgent search to return an REAgents structure and an Office search to return an REOffices structure. Therefore the REOfficeRosters structure would never be used.

**Proposed Solution:** It is our proposal that the REOfficeRosters structure be used as a return format for a new search called "Roster". The Search can be conducted using the OfficeID, BrokerID combinations as search criteria to return this data.

## 10. Meaning of certain search types

**Reporter:** michael.delgudio@mris.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** Section 7

### Description:

The Standard is intentionally vague on many items. However, since the SEARCH transaction is quite essential to RETS, it seems critical to understand what the Searches are intended to return.

Here are Examples:

1. Does the "Tour" Search return a series of properties that constitute an "Open House Tour", or does it represent the "Virtual tours" available for a property? What structure should the Search return?
2. Does the "Prospect" search represent "hot-sheet" updates for prospects, and therefore does it imply that there is some way to store saved criteria for that prospect. Or, does Prospect mean some type of contact management where the agent can use the server as an address book? This raises the same question how is this data added to the server, and what is the appropriate return structure.
3. Does "History" mean the changes that have occurred to the listing over its course, or the history of the property over the course of all its previous listings also? I believe it is the former, and not the latter, however it is unclear what the return XML structure should be to represent this.
4. How would one represent an open house using STANDARD-XML, as there are no XML tags that describe open houses in the standard.

## 11. RETS DTD orphans the ownership element

**Reporter:** michael.delgudio@mris.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** DTD

### Description:

The current RETS dtd does not properly place the ownership element within any other element structures. Due to this the element is orphaned, and can only be used as a root element. As the Own-

ership element is a simple text element and may not contain other elements -- it is rendered unusable. The Standard is silent on the intended use of this field.

**Proposed Solution:** We have interpreted ownership as follows and incorporated these changes into our current DTD:

Added missing element Ownership to ResidentialProperty, LotsAndLand, MultiFamily and TaxData. The implication being, if ownership appears in TaxData, it indicates the CURRENT ownership of the property and if present in the others, it indicates the offered ownership. I.e.

It is offered for sale as a Fee-Simple property. Ownership was appended to the list, as a 0 or 1 optional element that may contain PCDATA.

The selection of this location is arbitrary, since what "ownership" means is not defined in the spec. Our interpretation - as noted above - suggested the location of the element.

## Change Proposal #10: Extended Remarks

**Reporter:** michael.delgudio@mris.net

**RETS Document:** Real Estate Transaction Standard 1.0

**Location(s):** dtd

### Rationale:

Currently there are two types of remarks allowed in RETS, a Remarks and PublicRemarks. Not all systems are limited to these two. Systems with more than two types of remarks cannot publish those remarks using the standard XML output.

**Proposed Change:** The Remarks Element could be made more generic with a "Type" Attribute. The Type attribute would indicate the type of remark it contains. This may or may not deprecate "PublicRemarks", but it does allow for a more flexible remarks system. The DTD should be altered to allow for Zero or more remarks, or create a container element for the list of remarks.

```
<REMARKS>
  <REMARK type="Agent">Foo</REMARK>
  <REMARK type="Customer">Bar</REMARK>
  <REMARK type="Farm">Baz</REMARK>
  <REMARK type="Internet">Zing</REMARK>
</REMARKS>
```

The values permitted in the Type attribute could be agreed upon by the working group and published with the standard.

## Change Proposal #11: "Rooms" Element

**Reporter:** michael.delgudio@mris.net

**RETS Document:** Real Estate Transaction Standard 1.0

### Rationale:

The standard defines certain tags that represent specific rooms. These specific elements do not allow for the Standard XML version to properly model a given listing, nor does it take advantage of the structured nature of XML.

**Proposed Change:** This could be changed to a single, simple ROOM element, contained within a ROOMS element, that uses attributes to describe the room: for example:

```
<ROOMS>
  <ROOM name="Kitchen"
    length="24"
    width="18"
    height="9"
    Area="456"
    Dimensions="24x18"
    Fireplace="Yes"
    Flooring="Ceramic Tile">
  </ROOM>
</ROOMS>
```

This permits MLS's that maintain arbitrary lists of rooms to describe all the rooms in the listing without metadata.

This may or may not require the deprecation of other elements such as "LivingRoom", "DiningRoom", "FamilyRoom".

## Change Proposal #12: Valid XML for all response types

**Reporter:** michael.delgudio@mris.net

**RETS Document:** Real Estate Transaction Standard 1.0

### Rationale:

There is an apparent discrepancy in the way data should be returned from the server for different transactions. I was hoping the group could clarify my interpretation, and tell me if I am reading the standard properly.

In Section 4.9, the standard describes how information should be returned from a login attempt. As I interpret it, the return should look something like:

```
<RETS ReplyCode="0" ReplyText="Operation Successful">
  MemberName=Sally Superagent
  User=12345,1,Agent,12345
  MetadataVersion=1.0.0
  MinMetadataVersion=1.0.0
  OfficeList=BRK;01, BRK;02
  Balance=50
  TimeoutSeconds=600
  Expr=[date in RFC1123 format], 3
  Action=/action
  ChangePassword=/changepassword
  GetObject=/getObject
  [... Rest of capability URLs snipped out ...]
<RETS-STATUS/>
</RETS>
```

In this transaction, the "Tags" contained between the RETS tags are really just key/value pairs. Some of which, such as OfficeList and User, must be custom parsed because they may contain more than one piece of information.

However, in section 7.6, the standard describes how information should be returned from a search in COMPACT or COMPACT-DECODED format. Again, as I interpret it, the return should look something like:

```
<RETS ReplyCode="0" ReplyText="Operation Successful">
<COUNT Records="1"/>
<DELIMITER value=","/>
<COLUMNS>
col1, col2, col3
</COLUMNS>
<DATA>
val1, val2, val3
</DATA>
<MAXROWS/>
<RETS-STATUS/>
</RETS>
```

It appears that Section 7.6 is written using a different and more traditional use of "Tags". However, even though they are in a format that one could parse, and validate in XML format, discussion of this transaction has indicated that this is NOT in XML format:

Bruce Toback wrote on August 1, 2001:

"The compact formats aren't XML, so there's no DTD. Compact formats can be parsed by an engine that decodes XML, but there is no formal parent/child relationship as there is in XML."

My questions are:

1. Is my interpretation correct - are the sample returns above syntactically correct?
2. Is any thought being given to standardize these returns into a consistent format, so that parser code could be reused - or handed off to a readily available XML parsers?

It would be useful to standardize the responses across the transaction types and to use consistent language to describe "tags". I propose that we use the word "tag" to mean a marker delimiting data elements as is described in section 7.6. Specifically, tags only exist in the body of the response. Elements that exist in the HTTP header are called header fields.

Using this definition, the title of section 4.7.1 would change from "BrokerCode Tag" to "BrokerCode Header" or "BrokerCode Header Field".

A set of tags should be created for section 4.9 (and any other sections). This will allow a consistent parsing of the return data, whether in Compact, Compact-Decoded, Standard-XML or any other future format.

For backward compatibility, an option Request Argument can be added at section 4.7.2 Tagged-Response

```
tagged-response ::= TaggedResponse: ("true"|"false")
```

The responses can then be identified using correct tags, for example:

#### 4.10.3 Metadata Version Tags

```
metadata-ver-tag ::= <METADATA-VERSION Version="version"
  MinVersion = "version"/>
  version          ::= 1*2DIGITS.1*2DIGITS.1*5DIGITS
```

The rationale for this change is that it will allow simpler parsing of all of the transactions and will allow a consistent behavior for all transactions.

# RETS Change Proposal 001: Structured Data Exchange

**Number:** 1

**Authors:** Bruce Toback and Leo Bijl nagte

**Status:** Proposal

**Date:** December 25, 2001

**Version:** 1.0

## 1. Synopsis

RETS 1.0 does not provide any way for servers and clients to exchange information with internal structure except within the limited domain of a `STANDARD-XML` transaction. Structured Data Exchange adds a fourth data exchange type, to be designated `LOCAL-XML`, to allow structured information to be transmitted with a full metadata description. A `LOCAL-XML`-packaged query response is a well-formed XML document that can be parsed using any standard XML parser.

## 2. Rationale

RETS 1.0 specifies two methods for exchanging data between endpoints: `COMPACT`, which is a flat record format described by metadata, and `STANDARD-XML`, which is a structured record format that is described by a [DTD](#) and a [developer note](#). This means that endpoints can exchange all the information for a resource but with internal structure flattened, or limited information but with a rich internal structure. The obvious weakness here is that there is no way for endpoints to exchange full information with deep internal structure.

To address this need, developers have either been forced to flatten their data structures artificially (often at some cost in completeness), or to extend the standard RETS DTD. Since such extensions are nonstandard, they inhibit the interoperability of RETS implementations that use them. Structured Data Exchange provides a standard method for describing structure, eliminating the need for nonstandard extensions when resources have a complex internal organization.

In addition, some developers have expressed the need for an upload format that uses XML rather than the present HTML field encapsulation. Structured Data Exchange provides a framework within which that need can be accommodated, although this proposal does include modification of the Upload transaction.

## 3. Proposal

### 3.1 Specification Changes

This solution involves adding a new `LOCAL-XML` data return type, and several new fields to the Table metadata. The `LOCAL-XML` return type encapsulates the returned data as XML, using `<record>` as the root of each returned record. The return is thus a series of `<record>` elements within the `<DATA>` section of the search transaction response.

The server describes the `LOCAL-XML` document using six new metadata fields:

Name	Type	Description
<code>privateXMLTag</code>	Character	The XML tag or attribute name which encapsulates this field. If this element is omitted or blank, the system name is used as the XML tag.
<code>parentField</code>	Character	The system name of the field of which this field is a subfield -- or, in other words, the system name of the structure of which this field is a member. If this element is not included in the metadata or is blank, the field is not a subfield of any other field.
<code>isAttribute</code>	Boolean	If this metadata element is TRUE, the field's value is contained as an attribute of the parent field. If FALSE or omitted, the field is contained as an element of the parent field.
<code>isArray</code>	Boolean	If this metadata element is TRUE, the parent field may contain more than one of this field. If FALSE or omitted, there may be only one instance of this field within the parent. This element must be FALSE or omitted if <code>isAttribute</code> is TRUE.
<code>maximumElements</code>	Integer	The maximum number of elements if this field is an array. If the value is zero or omitted, there is no limit to the number of repetitions. However, if <code>isArray</code> is FALSE or omitted, this element must either be omitted or have a value of 0 or 1.

groupClassName	Character	A name that may be associated with this parent item. Meaningful only for fields that are designated as parent fields by one or more other fields. The value is advisory only; the client need not make use of the name provided.
----------------	-----------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

This scheme permits retrieved records to be XML-encoded complete with internal structure, yet still retains the flexibility offered by having the server provide metadata. Because it generates well-formed XML, the data can be parsed by off-the-shelf tools and easily assembled into flat records, if that's what's desired. In addition, because the XML structure is specified by the server's metadata, XML can be used for upload as well.

There is no requirement that either the XMLTag or the groupClassName be meaningful or human-readable, though some sites may wish to do this. Omitting any requirement for readability means that the tag names can be very short and even machine-generated to help reduce the size of the returned data. Since a client requesting the LOCAL-XML return format is presumed to have the metadata, there is no need to present the names to an end-user.

Since the metadata now includes structure information, and since the COMPACT data format has no way of designating structures, the server must decide for any array field which single instance will be transmitted. This will normally be the most important or most significant instance, since it will be the only one seen by a user whose client requests that data be returned in COMPACT format.

This will not represent a loss in information from earlier versions of the standard, since such decisions had to be made in any case when some field in a class's data represented an array. The usual case solution in that case was to provide multiple instances of fields in the metadata (for example, PHONE\_1, PHONE\_2, PHONE\_3 and so on); this strategy may still be used if desired.

### 3.2 Example of use

Assume table metadata for some resource that includes the following fragment:

SystemName	Type	length XMLTag	parentField	isAttribute	isArray	maximumElements	groupClassName
ListingID	Char	8 ListingID					
Commission	Numeric	6 Commission					
ListDate	Date	ListDate					
Address	Char	0 PropertyAddress		F	F		StreetAddress
StreetName	Char	20 Name	PropertyAddress	F	F		
StreetNumber	Char	6 Number	PropertyAddress	F	F		
ListingAgent	Char	0 ListingAgent			Y		Member
LAName	Char	40 Name	ListingAgent	F	F		
LANumber	Char	10 Number	ListingAgent	F	F		
LAPhone	Char	16 Phone	ListingAgent	F	T	4	
LAPager	Char	16 PagerNumber	ListingAgent	F	F		
SellingAgent	Char	0 SellingAgent			Y		Member
SAName	Char	40 Name	SellingAgent	F	F		
SANumber	Char	10 Number	SellingAgent	F	F		
SAPhone	Char	16 Phone	SellingAgent	F	T	4	

Then a RETS client requesting LOCAL-XML with all fields selected would receive data similar the following:

```

<record>
  <ListingID>18331402</ListingID>
  <Commission>6</Commission>
  <ListDate>05 DEC 2002</ListDate>
  <PropertyAddress>
    <Name>Downing</Name>
    <Number>10</Number>
  </PropertyAddress>
  <ListingAgent>
    <Name>Bill Ding</Name>
    <Number>735310487</Number>
    <Phone>888 666-1432</Phone>
    <Phone>614 234-5678</Phone>
    <PagerNumber>800 759-7243</PagerNumber>
  </ListingAgent>
  <ListingAgent>
    <Name>Rusty Nail</Name>
    <Number>638310107</Number>
    <Phone>888 555-3388</Phone>
    <Phone>604 888-5553</Phone>
    <PagerNumber>800 759-7243</PagerNumber>
  </ListingAgent>
  <SellingAgent>
    <Name>Ford Prefect</Name>
    <Number>36392837</Number>
    <Phone>877 444-3238</Phone>
    <Phone>619 888-3410</Phone>
    <PagerNumber>800 759-7243</PagerNumber>
  </SellingAgent>
</record>

```

The COMPACT format for the same record would be (using "+" to indicate tab characters):

```

<COLUMNS>ListingID+Commission+ListDate+Address+StreetName+StreetNumber+
ListingAgent+LANumber+LAPhone+LAPager+SellingAgent+
SANumber+SAPhone&lt;COLUMNS>
<DATA>
18331402+6+05 DEC 2002++Downing+10++Bill Ding+735310487+888 666-1432+
800 759-7243++Ford Prefect+877 444-3238+800 759-7243+
</DATA>

```

Note that the arrays `ListingAgent` and `Phone` (within `ListingAgent`) have been reduced to a single instance. Although the "container" elements are explicitly transmitted in this example, they need not be. This is left as an option for the server, since container elements are empty of any textual content.

### 3.3. Interaction between LOCAL-XML and SELECT

Because the `SELECT` argument can include any arbitrary subset of fields from the server's Table metadata, the server must modify its interpretation of `SELECT` when a client requests the `LOCAL-XML` return format. In particular, the server must include tags for all containing elements when the client requests only a contained element. For example, if a client operating with the above metadata fragment were to request only the `LAPhone` field in its `SELECT` argument, the server's response would look like:

```

<record>
  <ListingAgent>
    <Phone>1-800-SELLNOW</Phone>
  </ListingAgent>
</record>
<record>
  <ListingAgent>
    <Phone>312 555-1212</Phone>
    <Phone>206 555-1212</Phone>
  </ListingAgent>
</record>

```

The client must be prepared in this case to receive element names that it had not explicitly selected, since they are required to preserve the containment hierarchy.

### 3.4 Using Structured Data Representation for Upload

Because of the substantial changes required to modify the Upload transaction to accept an XML representation -- as well as to allow references to structured data in the validation language -- this proposal does not address using Structured Data

Representation for upload. A separate change proposal will address this issue.

#### 4. Development Impact

Since the functionality described in this Change Proposal is optional, there is no mandatory development impact. If a client chooses to implement Structured Data Exchange, it will need to accept and interpret the new metadata fields described in Section 3.1, and will need to incorporate an XML parser to parse the returned information. Its human-interface components, if any, will also need to be modified to accept and display multiple instances of a field, and may be changed so as to make the hierarchy explicit.

A server implementing this proposal will need to transmit the additional metadata fields, and will need substantial changes to extract and package array and contained elements.

A site using a server that implements this proposal will need to create and populate the additional metadata fields. Note that if the site is using an essentially flat data representation to begin with, the population of the additional metadata tasks is trivial, since all of the default values are acceptable.

If a site wishes to provide as much flexibility as possible to clients, it should insure that the `groupName` attributes it chooses match across all resource classes. This allows clients to implement cross-resource analysis on these entities.

#### 5. Compatibility

A RETS 1.0 client communicating with a server that implements Structured Data Exchange will not recognize the new metadata elements and will therefore not be able to see when it has requested (via the `SELECT` clause) elements with internal structure. Section 3.1 of the proposal addresses this situation, requiring the server to include the most important or most visible instance of the element, together with its internal structure. This is essentially what happens now, since servers must arbitrarily flatten deep data representations in order to present a `COMPACT` view of the data.

A RETS 1.0 server communicating with a client that implements Structured Data Exchange will not send any of the new metadata elements. It will thus be impossible for the client to formulate a query or a data request that the server does not understand. No additional effort is required in client implementations to insure forward compatibility.

*Last changed 25 December 2001, 1520 MST by [Bruce Toback](#).*

## RETS Change Proposal 002: Compression Negotiation

**Number:** 2

**Authors:** Bruce Toback

**Status:** Proposal

**Date:** December 29, 2001

**Version:** 1.0

### 1. Synopsis

RETS 1.0 does not provide a standard method for client and server to negotiate data compression for query responses. This proposal suggests and standardizes the use of the HTTP `Accept-Encoding` and `Transfer-Encoding` headers so that a client and server can negotiate a compression method.

### 2. Rationale

There is currently no standard method within RETS for a client and server to negotiate a compression method. Since the response to a RETS query can be very large, this represents a deficiency in the current specification. Compression would be even more important if the a `LOCAL-XML` return type is added to the specification.

By using the HTTP `Accept-Encoding` header, a client can indicate its readiness to accept a compressed response in an HTTP-compliant way, and the server can similarly use the standard `Content-Encoding` header to indicate that the response is compressed, and by what method. This preserves normal HTTP processing at both ends.

### 3. Proposal

#### 3.1 Specification Changes

An `Accept-Encoding` header following RFC 2616 will be added to Section 3.5 of the [RETS protocol specification](#), describing the current set of optional request headers:

```
Accept-Encoding: encoding [,encoding]
```

where *encoding* is the MIME type of the encoded data, for example, `application/gzip`. A client desiring a compressed return SHOULD request (and support) at least `application/bzip`.

A corresponding `Content-Encoding` optional response header will be added to Section 3.8 of the specification:

```
Content-Encoding: encoding
```

where *encoding* is the MIME type of the encoded data. A RETS-compliant server intending to support compression SHOULD support at least `application/bzip`.

The specification does not require either the client or the server to support compression. The server is permitted to ignore the client's `Accept-Encoding` header and supply the response unencoded.

#### 3.2 Compression Methods

As noted above, RETS endpoints intending to support compression SHOULD accept and/or generate `bzip` encoding. `bzip` is similar to the popular `gzip` compression method, but has a less restrictive license than the [GNU `gzip`](#) library. (A [Java implementation of `bzip`](#) is also available under the [LGPL](#).) Clients and servers are, of course, free to use any other encoding methods.

### 4. Development Impact

Because the changes described in this proposal are optional, there is no forced development impact. A client or server writer wishing to implement this proposal will need to integrate the `bzip` library or a suitable substitute; this will not be difficult

except where no library can be found for the development platform. In addition, both client and server will need to be changed to emit and process the new headers. This is generally minor.

## **5. Compatibility**

This proposal relies only on optional HTTP/1.1 header semantics, and servers are permitted to ignore the client's request for compression. Thus, there are no compatibility issues: if both client and server implement this extension, then server responses may be compressed. If either does not implement the extension, responses will be sent uncompressed, as now.

*Last changed 31 December 2001, 2120 MST by [Bruce Toback](#).*

## RETS Change Proposal 003: Global Structure Information

**Number:** 2

**Authors:** Bruce Toback

**Status:** Proposal

**Date:** January 1, 2002

**Version:** 1.0

### 1. Synopsis

This proposal adds several fields to the RETS 1.0 metadata specification to allow a client to find and exploit relationships between resources offered by a server.

### 2. Rationale

RETS servers frequently make available resources that are interrelated. A simple example is the inclusion of agent information on a listing. Currently, there is no way for a client to understand these relationships except by guessing on the basis of RETML tag names and well-known resource names. This proposal adds fields to the `TABLE` metadata to make these relationships explicit.

The proposal does not require that clients make use of the information, nor does it require that servers make it available. However, a client that makes use of the information may be able to provide a richer data set to the user or optimize fetches from the server.

### 3. Proposal

#### 3.1 Specification Changes

The following metadata fields are added to the `TABLE` metadata (Table 11-4 in the RETS protocol specification):

Field Name	Content Type	Description
ChildResourceID	Character	The ResourceID (Table 11-2) of the resource that for which this field functions as a foreign key (if any). The name given <b>MUST</b> appear in the <code>METADATA-RESOURCE</code> table.
ChildResourceClassName	Character	The name of the resource class for which this field functions as a foreign key. This name <b>MUST</b> appear in the <code>RESOURCE-CLASS</code> table for the given ChildResourceID.
ChildFieldSystemName	Character	The <code>SystemName</code> 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 <code>SystemName</code> in the <code>METADATA-TABLE</code> section of the metadata for the resource class, and the named item must have its <code>Searchable</code> and <code>Unique</code> attributes set to <code>TRUE</code> .

If the server does not transmit values for all three of these fields, the client **MUST** operate as if it had received none of them.

#### 3.2 Implementation Notes

When displaying data from a particular resource, a client **MAY** use the child resource information in the metadata to retrieve and display related records from other resources. It does so by creating a search on the specified resource and resource class using a simple equal-to query. There is no restriction on the content of this query, however, so the server should not anticipate any particular sequence of operations when it supplies child resource metadata.

### 4. Development Impact

Because the changes described in this proposal are optional, there is no forced development impact. Server developers wishing to implement this change proposal need only transmit the new metadata fields. Client developers wishing to take advantage of the additional structural information may do so as they see fit.

## **5. Compatibility**

Because this change involves only optional metadata, there is no forward or backward compatibility impact from this change. Clients that implement this proposal but do not receive the optional metadata fields simply display or operate on the transmitted search results without further enhancement.

*Last changed 2 January, 2002, 1715 MST by [Bruce Toback](#).*

## RETS Change Proposal 004: Global Foreign Key Information

**Author:** Stuart Schuessler

**Organization:** MarketLinx

**Address:** P.O. Box 24119, Knoxville, TN 37933-2119

**Telephone Number:** (865) 218-3606

**Email:** [sschuessler@marketlinx.com](mailto:sschuessler@marketlinx.com)

**Status:** Proposal

**Date:** January 24, 2002

**Version:** 1.0

### 1. Synopsis

This proposal adds a new metadata type call `METADATA-FOREIGNKEYS` to the RETS 1.0 metadata specification to allow a server to advertise relationships between resources offered.

### 2. Rationale

RETS servers frequently make available resources that are interrelated. A simple example is the inclusion of agent information on a listing. Currently, there is no way for a client to understand these relationships except by guessing on the basis of RETML tag names and well-known resource names. This proposal adds a new metadata type that makes these relationships known.

The proposal does not require that clients make use of the information, nor does it require that servers make it available. However, a client that makes use of the information may be able to provide a richer data set to the user or optimize fetches from the server.

### 3. Proposal

#### 3.1 Specification Changes

A new metadata type called ForeignKeys needs to be added to the specification Chapter 11 section 4. The following is the description of `METADATA-FOREIGNKEYS` and its purpose:

Foreign Keys allows the operator of a particular server to advertise its supported parent child relationships. The Foreign Keys metadata starts with a `<METADATA-FOREIGNKEYS>` tag with Version and Date attributes. This is followed by a `<COLUMNS>` section, which contains the name of the fields as defined in Table 1 followed by the `<DATA>` section, which contains the actual field information. The Foreign Keys metadata has the following format:

```
<METADATA-FOREIGNKEYS Version="foreignkey-version" Date="foreignkey-date">
<COLUMNS>foreignkey-field * (•foreignkey -field) •</COLUMNS>
<DATA> foreignkey -data*(• foreignkey-data) •</DATA>
</METADATA-FOREIGNKEYS>
```

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

This is the version of the ForeignKeys metadata. The convention used is a "`<major>.<minor>.<release>`" numbering scheme. Every time any element of the ForeignKey metadata changes the version number **MUST** be incremented.

*foreignkey-date ::= DATE* The latest change date of the ForeignKey metadata.

*foreignkey-field ::= <Field Name from Table 1>* *foreignkey-data ::= <valid value as defined in Table 1>*

An example ForeignKey section follows:

```

GetMetadata request:
Type: METADATA-FOREIGNKEYS

Compact Reply:
<METADATA-FOREIGNKEYS Version="1.00.000000" Date="Wed, 23 Jan 2002 12:37:38 GMT">
<COLUMNS>PARENT_RESOURCE_ID•PARENT_CLASS_ID•PARENT_SYSTEMNAME• CHILD_RESOURCE_ID•CHILD_CLASS_ID•CHILD_SYSTEMNAME•</CO
<DATA>Property•RES•MLSNUM•TAX•TAX•MLSNUM•</DATA>
<DATA>Property•RES•MLSNUM•History•History•MLSNUM•</DATA>
<DATA>Property•RES•MLSNUM•OpenHouse•OpenHouse•MLSNUM•</DATA>
<DATA>Property•RES•ListingAgentID•Agent•Agent•AgentID•</DATA>
<DATA>Property•RES•COListingAgentID•Agent•Agent•AgentID•</DATA>
<DATA>Property•RES•SellingAgentID•Agent•Agent•AgentID•</DATA>
<DATA>Property•RES•COSellingAgentID•Agent•Agent•AgentID•</DATA>
<DATA>Property•RES•ListingOfficeID•Office•Office•OfficeID•</DATA>
<DATA>Property•RES•SellingOfficeID•Office•Office•OfficeID•</DATA>
</METADATA-FOREIGNKEYS>
    
```

Field Name	Content Type	Description
Foreign_Key_ID	1*32ALPHANUM	A Unique ID that represents the foreign key combination.
Parent_Resource_ID	1*32ALPHANUM	The ResourceID (Table 11-2) of the resource that for which this field functions as a foreign key . The name given MUST appear in the METADATA-RESOURCE table
Parent_Class_ID	1*32ALPHANUM	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 Parent_Resource_ID
Parent_SystemName	1*32ALPHANUM	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 Parent_Class_ID, and the named item must have its Searchable attribute set to TRUE.
Child_Resource_ID	1*32ALPHANUM	The ResourceID (Table 11-2) of the resource that for which this field functions as a foreign key . The name given MUST appear in the METADATA-RESOURCE table.
Child_Class_ID	1*32ALPHANUM	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 Child_Resource_ID.
Child_SystemName	1*32ALPHANUM	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 Child_Class_ID, and the named item must have its Searchable attribute set to TRUE.

If the server does not transmit values for all six of these fields, the client MUST operate as if it had received none of them.

Each combination of the seven fields MUST be unique.

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.

### 3.2 Implementation Notes

When displaying data from a particular resource, a client MAY use the child resource information in the metadata to retrieve and display related records from other resources. It does so by creating a search on the specified resource and resource class using a simple equal-to query. There is no restriction on the content of this query, however, so the server should not anticipate any particular sequence of operations when it supplies child resource metadata.

This is not meant to replace the lookup metadata in 11.4.2 or lookuptype metadata in 11.4.3. The purpose of the new metadata is to advertise the foreign key relationships between the major tables of a system.

## 4. Development Impact

Because the changes described in this proposal are optional, there is no forced development impact. Server developers wishing to implement this change proposal need only transmit the new metadata type. Client developers wishing to take advantage of the additional structural information may do so as they see fit.

## 5. Compatibility

Because this change involves only optional metadata, there is no forward or backward compatibility impact from this change. Clients that implement this proposal but do not receive the optional metadata fields simply display or operate on

the transmitted search results without further enhancement.

## 6. Proof/Need of Concept Examples

All of the following examples assume the client is requesting data in the COMPACT format and is familiar with NORMALIZED data

### 6.1 Agent Listings

The RETS Client (client) request data from the server for the property listing and needs to have UP TO DATE Agent information as part of the property data. The client retrieves the new Foreign Key metadata to lookup up the RETS Well Known Resource and Class Name for Agent and gets the field name that links to the Well Known Residential Resource and Class name field. The following record is retrieved from the metadata:

```
<DATA>Property•RES•ListingAgentID•Agent•Agent•AgentID•</DATA>
```

The client proceeds to download all agent records modified TODAY+ once per hour and all Property Records modified TODAY+ once per hour

Because of the knowledge the client has obtained from the Foreign Key metadata it can bring up a listing and have all the Agent information instead of just a few fields normally listed in the flat record

OR

Because the CanChildBeParent is TRUE, the client can bring up an Agent and show all property listing this agent has.

This is all accomplished automatically for the client and can easily change should the Metadata version be incremented.

### 6.2 Property History and Agent Information

The RETS Client (client) request data as needed from the RETS Server. It does not maintain a local copy of the database, however it does maintain the Full Metadata structure.

The following is the data for this example in the Foreign Key metadata.

```
<DATA>Property•RES•SellingAgentID•Agent•Agent•AgentID•</DATA>  
<DATA>Property•RES•MLSNUM•History•RES•MLSNUM•</DATA>
```

The client receives a request from the user to download a particular property record. The server delivers the record to the client.

The client then receives a request from the user to download the History of this property. The client Looks up the History relationships to the property and determines the key field. It then generates the appropriate request to the RETS Server and the server returns the correct data.

Now the user would like to see the Selling Agent for a property. The client Looks up the Agent relationship to the property for SellingAgentID and generates the appropriate request to the server. The server responds with the data.

This is all accomplished automatically for the client and can easily change should the Metadata version be incremented.

### 6.3 The unknown (Why there should not be limitations on nesting)

Let's say an MLS System wanted to introduce something new to the market that has never been seen before. For the sake of this example let's say it is a Transaction System.

We have a MLS System that has published its Transaction Resource and Class Types.

We have the following foreign key metadata:

```
<DATA>Property•RES•MLSNUM•TRANSACTION•RES•MLSNUM•</DATA>
<DATA>TRANSACTION•RES•TRANSACTION_ID•TRANSACTION•HISTORY•PARENT_TRANSACTION_ID•</DATA>
<DATA>TRANSACTION•HISTORY•PARENT_TRANSACTION_ID•TRANSACTION•HISTORY•CHILD_TRANSACTION_ID•</DATA>
<DATA>TRANSACTION•HISTORY•NEXT_TRANSACTION_ID•TRANSACTION•HISTORY•CHILD_TRANSACTION_ID•</DATA>
```

The transaction client has the task of obtaining the transaction history for a particular property. The data structure for the Transaction Database is recursive in nature using nodes to follow paths of transactions. The client will be required to recurse those nodes to their completion to display the complete transaction history. The client is also capable of adding it's own transaction nodes.

This example albeit incomplete demonstrates that given the Foreign Key metadata, a client capable of using this new metadata type and traversing the tree that it represents would be prepared for most future expansions of the RETS Specification. It also opens a whole new world up to RETS. Nesting should not be limited in this proposal except for well known RETS Names.

*Last changed 15 February, 2002, 1000 MST by [Webmaster](#).*

# RETS Change Proposal 005: Forward-Compatible Metadata Extension Mechanism

**Author:** Leendert M. "Leo" Bijnagte

**Organization:** Fidelity National Information Solutions

**Address:** 100 Washington Ave. South #900, Minneapolis, MN 55401

**Telephone Number:** (612) 661-1087

**Email:** leob@vistainfo.com

**Status:** Proposal

**Date:** January 24, 2002

**Version:** 1.0

## 1. Synopsis

A change to the metadata DTD is proposed to allow for some types of future extensions without requiring further changes to the DTD.

## 2. Rationale

The current RETS Metadata DTD is specific to Version 1.0 of the standard. When new metadata fields are added in future versions of the standard, clients using validating parsers will break because they will encounter unexpected elements in the DTD. This proposal adds an `<extension>` tag to the existing DTD so that future extensions will not change the document type definition.

## 3. Proposal

### 3.1 Specification Changes

A new element is added to the Metadata DTD:

```
<!ELEMENT extension ANY>
<!ATTLIST extension
  Name CDATA #REQUIRED
  Type CDATA #REQUIRED>
```

This element is added to the top-level definition of each of the existing metadata types.

### 3.2 Implementation Notes

Future extensions to any Metadata type will add no new elements to the DTD. Instead, the extension will be included as an element of type `Extension`, with its name and type given as attributes of the `Extension` element.

This proposal does not include a mechanism for adding new metadata types to the DTD. Since the addition of new metadata types generally implies significant changes to program logic, and since entirely new metadata types are expected to be added only rarely, allowing top-level extensibility is not viewed by the program authors as justifying the additional implementation effort required.

## 4. Development Impact

The extension proposed amounts to a new way of representing metadata within the DTD. This will require changes to the semantic processing of the metadata DTD in order to access any future metadata extensions. However, this will be a one-time change and there will be no further development impact with future metadata extensions.

## 5. Compatibility

This change involves a change to the metadata DTD. Clients that check metadata returns for validity as well as well-formedness will report that the metadata from a server implementing this change proposal is incorrect. It is not clear how many client implementations actually check for a valid metadata return in addition to requiring a well-formed return; the proposal author believes the number to be small or zero.

The purpose of this change is to prevent such incompatibilities in the future; however, there is no way to avoid at least one incompatible change during the transition.

*Last changed 15 February, 2002, 1030 MST by [Webmaster](#).*

## RETS Change Proposal 006: Search Standard Queries

**Author:** Stuart Schuessler

**Organization:** MarketLinx

**Address:** P.O. Box 24119, Knoxville, TN 37933-2119

**Telephone Number:** (865) 218-3606

**Email:** [sschuessler@marketlinx.com](mailto:sschuessler@marketlinx.com)

**Status:** Proposal

**Date:** 2/21/2002

**Version:** 1.0

### 1. Synopsis

This proposal adds a new metadata type called `METADATA-STANDARD_QUERIES` to the RETS 1.0 metadata specification to allow a server to advertise searches that are optimized and follow the boards business rules.

### 2. Rationale

RETS Clients frequently make the same queries to the RETS Servers over and over. An example is all the active listing modified TODAY. By allowing the RETS Server to advertise the searches most RETS Clients would be interested that follow the business rules established by the board it provides the following advantages:

1. A server could optimize for well known searches.
2. ACTIVEAGENT could be removed as a RESOURCE/CLASS type from well known names and provided as a Standard Query. The server would now longer need to duplicate metadata simply to represent a defined search.
3. The MLS Vendor could provide the 3rd parties with proper searches that meets most of their requirements and follows the Boards business rules for downloading data. Again, the reason ACTIVEAGENT exists is because the board defines what it means to be an active agent.
4. It lowers the barrier of entry for 3rd parties. They would simply need to look at the available Standard Queries to have an understanding of searching the RETS Server.

### 3. Proposal

#### 3.1 Specification Changes- Class Metadata

In order to add a new Metadata the version and Date MUST be added to the Class Metadata. The following Additions are needed to Table 11-3 (Metadata Content: Resource Class):

Metadata Field	Content Type	Description
Standard_Query_Version	1*2DIGITS.1*2DIGITS.1*5DIGITS	The latest version of the Standard Query metadata for this Class. The convention used is a "<major>.<minor>.<release>" numbering scheme. Clients MAY rely on this data for cache management. A blank version indicates no Standard Query is available for this Class.
Standard_Query_Date	DATE	The date on which the any of the Standard Query metadata for this Class was last changed. Clients MAY rely on this date for cache management. A blank date indicates no Standard Query is available for this Class.

#### 3.2 Specification Changes- New Metadata

A new metadata type called `METADATA-STANDARD_QUERIES` needs to be added to the specification Chapter 11 section 3 (Metadata Format for Class Elements). The following is the description of `METADATA-STANDARD_QUERIES` and its purpose:

Standard Queries allows the operator of a particular server to advertise its known and optimized queries. The Standard Queries metadata starts with a <METADATA-STANDARD\_QUERIES> 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 1 followed by the <DATA> section, which contains the actual field information. The Standard Queries metadata has the following format:

```
<METADATA-STANDARD_QUERIES Resource="resourceID" Class="ClassID" Version="Standard Query-version" Date="Standard Query-date">
<COLUMNS>Standard Query-field * (*Standard Query -field) *</COLUMNS>
<DATA> Standard Query-data*( * Standard Query-data) *</DATA>
</METADATA-STANDARD_QUERIES>
```

*Standard Query-version ::= 1\*2DIGITS.1\*2DIGITS.1\*5DIGITS*

This is the version of the Standard Query metadata. The convention used is a "<major>.<minor>.<release>" numbering scheme. Every time any element of the Standard Query metadata changes the version number **MUST** be incremented.

*Standard Query-date ::= DATE* The latest change date of the Standard Query metadata.

*Standard Query-field ::= <Field Name from Table 1>*

*Standard Query-data ::= <valid value as defined in Table 1>*

An example METADATA-STANDARD\_QUERIES section follows:

```
GetMetadata request:
Type: METADATA-STANDARD_QUERIES

ID: Property:RES

Compact Reply:
<METADATA-STANDARD_QUERIES Resource="Property" Class="RES" Version="1.00.000000" Date="Wed, 23 Jan 2002 12:37:38 GMT">
<COLUMNS>*STANDARD_QUERY_ID*SELECT*QUERY*STANDARD_QUERY_LABEL*STANDARD_QUERY_DESCRIPTION*REFRESHRATE*</COLUMNS>
<DATA>ACTIVE_LISTINGS*(ListStatus=Active)*Active Listings>Returns All Residential Active Listings*30*</DATA>
<DATA>ACTIVE_LISTINGS_MODIFIED_TODAY*(ListStatus=Active),(Modified=TODAY)*Today's Modified Active Listings>Returns all
<DATA>ACTIVE_LISTINGS_MODIFIED_TODAY_SHORT*MLSNUM,MODIFIED,PHOTOCOUNT*(ListStatus=Active),(Modified=TODAY)*Today's Modified
</METADATA-STANDARD_QUERIES>
```

Metadata Field	Content Type	Description
Standard_Query_ID	1*32ALPHANUM	A UniqueID for Resource:Class that represents the Standard Query
Select	field*(,field)	Same as Select defined in 7.4.5
Query	<The Query to be executed by the server>	Same as Query defined in 7.3.2, except it <b>MUST NOT</b> contain another Standard_Query_ID
Standard_Query_Label	1*32ALPHANUM	A Client User Readable label so that the client can provide the search as an option to the user.
Standard_Query_Description	1*1024ALPHANUM	A description of the search and it's purpose.
Refresh_Rate	1*2DIGIT	The rate the data is available to be refreshed. Some RETS Servers <b>MAY</b> choose to only provide the data refreshed a different intervals to avoid abuse by the client software. If the Refresh_Rate is 0 or NULL then the data is provided as "live" data with no delay. Refresh Rate is in Minutes.

### 3.3 Specification Changes- Section 7.3.2

The Query specification **MUST** be rewritten to include Standard\_Query\_ID. The changes are in **Bold**

The specification consists of the query itself **or a Standard\_Query\_ID** together with a designation of the query language

Query ::=<The query to be executed by the server> **OR** <Standard\_Query\_ID>

The query is specified in the language described in Section 7.7.

**The Standard\_Query\_ID is defined in** METADATA-STANDARD\_QUERIES

### 3.4 Implementation Notes

The server SHOULD optimize the queries transmitted in the `METADATA-STANDARD_QUERIES` Metadata. On SQL Databases with support for Stored Procedures a stored procedure could be created for each of the standard queries. The client software SHOULD advertise to the user the available standard queries to allow the user to take advantage.

### 4. Development Impact

Because the changes described in this proposal are optional, there is no forced development impact. Server developers wishing to implement this change proposal need only transmit the new metadata type. Client developers wishing to take advantage of the additional query information may do so as they see fit.

### 5. Compatibility

Because this change involves only optional metadata, there is no forward or backward compatibility impact from this change. Clients that implement this proposal but do not receive the optional metadata fields simply display or operate on the transmitted search results without further enhancement.

### 6. Example

Retrieve the Metadata:

```
http://rets.someserver.com/rets/Getmetadata?type=Metadata-Standard_Queries&id=PROPERTY:RES
```

and perform the search:

```
http://rets.someserver.com/rets/Search?SearchType=PROPERTY&Class=RES&Query=ACTIVE_LISTINGS&Format=COMPACT
```

*Last changed 4 March, 2002, 1000 EST by [Webmaster](#).*

## RETS Change Proposal 007: Login Reply Code

**Author:** Stuart Schuessler

**Organization:** MarketLinx

**Address:** P.O. Box 24119, Knoxville, TN 37933-2119

**Telephone Number:** (865) 218-3606

**Email:** [sschuessler@marketlinx.com](mailto:sschuessler@marketlinx.com)

**Status:** Proposal

**Date:** 3/1/2002

**Version:** 1.0

### 1. Synopsis

This proposal adds a new Login Reply Code to support the denial of multiple logins

### 2. Rationale

RETS Servers need to give the RETS Clients more information should a denial of multiple logins occur. Because there are special actions a client may take should this error occur there needs to be something more than just 20036 Miscellaneous Login Error

### 3. Proposal

#### 3.1 Add Login Error Code

The Following Error Code SHOULD be added to Table 4-3

Reply Code	Meaning
20004	Another user using the same User ID is currently logged in. Login Denied

#### 3.4 Implementation Notes

When a client receives this error message it SHOULD inform the user of the error condition.

### 4. Development Impact

Because the changes described in this proposal are optional, there is no forced development impact. Server developers wishing to implement this change proposal need only transmit the ReplyCode. Client developers wishing to take advantage of the additional error information may do so as they see fit.

### 5. Compatibility

Because this change involves only an optional ReplyCode, there is no forward or backward compatibility impact from this change. Clients that implement this proposal but do not receive the optional ReplyCode MAY assume the feature is not supported by the server.

### 6. Example

```
<RETS ReplyCode="20004" ReplyText="Another user using the same User ID is currently logged in. Login Denied">
```

*Last changed 19 March, 2002, by [Webmaster](#).*

# RETS Change Proposal 008: Query Language (Search Syntax) Modifications

**Author:** Leendert M. "Leo" Bijnagte

**Organization:** Fidelity National Information Solutions

**Address:** 100 Washington Ave. South #900, Minneapolis, MN 55401

**Telephone Number:** (612) 661-1087

**Email:** leob@vistainfo.com

**Status:** Proposal

**Date:** March 18, 2002

**Version:** 1.0

## 1. Synopsis

Changes are proposed to the Query language to clarify usage and add some missing functionality.

## 2. Rationale

The specification of the syntax for searches defined in version 1.0 of the spec is not only confusing but in spots inaccurate. Due to the omission of a NOT operator on query components, the current DMQL language does not allow for the generation of certain types of queries commonly used by clients and supported by servers. Usage of nesting and the AND and OR operators is shown in the examples but the production rules are not clear in the augmented BNF. The suggested revisions (tentatively DMQL2) are to the augmented BNF of the Boolean elements of the language and extensions to the definitions that make up field-criteria.

The suggested revision for defining the Query takes its form from that of the <search condition> found in the BNF of SQL92 ([www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt](http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt)) with the exception that the intermediate production <boolean test> and its <truth value> have been deprecated to always TRUE to retain compatibility for existing queries.

Separate but included revisions to the details of the field-criteria are three-fold. First, the addition of string-literal to field-value to provide searching for string field content that might include non-ALPHANUM characters. The string-literal definition matches that of SQL-92. Second, replacing the 1\*DIGIT production to the newly defined number to support decimal numbers. Third, a change to the definition of range to use number and add string-eq to permit ranges on all datatypes to avoid forcing complex query construction for the currently excluded datatypes.

With the use of the specified revisions, field-criteria becomes a subset of the SQL92 <predicate> options that provides equivalent syntax for basic <predicate comparison>, <predicate between>, and <predicate in> expressions without the need (or ability) to specify joins or sub-queries. This fits well with the flattened views embodied in the Resources and the way Metadata links Lookups to Fields.

## 3. Proposal

### 3.1 Specification Changes

1. A revised BNF for the Query language and instructions about string-char and number substitution in section 7.7:

```
Query ::= search-condition
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-content ")"
field-value  ::= lookup-list | string-list | range-list | period | number | string-literal
lookup-list  ::= lookup-or | lookup-not | lookup-and
lookup-or    ::= "|" lookup *( "," lookup )
lookup-not   ::= "~" lookup *( "," lookup )
lookup-and   ::= "+" lookup *( "," lookup )
lookup       ::= <any legal ALPHANUM value for the field as defined in the metadata>
string-list  ::= 1*( 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 1*"?*" *ALPHANUM
string_literal ::= <"> 1*( *( PLAINTEXT except <"> ) *( 2<"> ) *( PLAINTEXT except <"> ) ) <">
range-list   ::= 1*( range *( "," range ) )
range        ::= between | greater | less
between      ::= ( period | number | string-eq ) "-" ( period | number | string-eq )
greater      ::= ( period | number | string-eq ) "+"
less         ::= ( period | number | string-eq ) "-"
period       ::= (date | datetime | time)
number       ::= 1*DIGIT [ "." *DIGIT ]
date         ::= (year "-" month "-" day) | TODAY
datetime    ::= (year "-" month "-" day "T" hour ":" minute ":" second [ "." fraction ]) | NOW
time         ::= (hour ":" minute ":" second [ "." fraction ])
fraction     ::= 1*3DIGIT
second       ::= 2DIGIT
minute       ::= 2DIGIT
hour         ::= 2DIGIT
day          ::= 2DIGIT
month        ::= 2DIGIT
year         ::= 4DIGIT
NOW          ::= "NOW" <the host should substitute this with its current date/time>
TODAY        ::= "TODAY" <the host should substitute this with its current date>

```

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, a server MAY round as follows, down for the bottom of ranges or values less than .5 and round up for the top of ranges or values .5 or greater..

## 2. The addition of DMQL2 as a Query Type in section 7.3.2:

```

QueryType   ::= DMQL | DMQL2

```

An enumeration giving the language in which the query is presented. The valid values are "DMQL" for RETS/1.0 compliance and "DMQL2" which indicates the query language described in Section 7.7.

### **3.2 Implementation Notes**

This proposal minimizes the changes to the query syntax while filling some important holes. The author hopes to have balanced the implementation effort that will be required by servers with the need for more SQL like syntax for the clients.

## **4. Development Impact**

The proposal requires server's search mechanisms to support the additional functionality of the NOT operator. If such functionality is not currently available, servers would need to provide at minimum a way to break down a DMQL2 query into multiple DMQL queries and perform Boolean logic on the result sets to be compliant with the new version.

The proposal for string literal support includes an option for servers that could limit the impact to parsing for quote characters and doing ? substitution for some range of characters.

The proposal for allowing decimals in numbers includes an option for servers to do rounding that could limit the impact to parsing for fractional elements and rounding.

The proposal for including string-eq in ranges will require servers that do not have greater than or less than search functionality on character data to provide such functionality.

Clients will not be required to change in any way to be compliant.

## **5. Compatibility**

Existing DMQL client query syntax would continue to conform while servers would need to provide processing for the new version to be compatible with the new release of the spec. Servers could elect to remain at version 1.0 of the spec; clients wanting to make use of the new syntax may want to retain some accommodation for connecting to servers at version 1.0 of the specification.

*Last changed 22 March, 2002 by [Webmaster](#).*

**Subject: Comment on RETS Change Proposal 008; Enhancement**  
**From: [Michael DelGaudio](#)**  
**Date: Thursday, June 13, 2002 7:45:07 AM**

Submitted By:

Mark Sleeman  
Templates for Business Inc.  
Email: [Mark.Sleeman@t4bi.com](mailto:Mark.Sleeman@t4bi.com)  
Phone: (604) 434-5967

AND

Mike DelGaudio  
Metropolitan Regional Information Systems, Inc.  
9420 Key West Ave, Suite 200  
Rockville, MD 20850  
E-Mail: [michael.delgaudio@mriss.net](mailto:michael.delgaudio@mriss.net)  
Phone: (301)838-7152

Subject:

Enhancement to RETS Change Proposal 008:  
Allow a search string to contain string-literals enclosed by single quotes.

Reference:

RETS Change Proposal 008 (As proposed by Leendert M. "Leo" Bijngate):  
Query Language (Search Syntax) Modifications  
(<http://www.rets-wg.org/comments/rcp008.html>)

Introduction:

In Section 3.1 Specification Changes it has been proposed  
that the use of the string-literal be added to the DMQL Specification:

```
string_literal ::= <"> 1*( *( PLAINTEXT except <"> ) *( 2<"> ) *( PLAINTEXT  
except <"> ) ) <">
```

As stated in section 2. this addition will accomplish the following:

"...the addition of string-literal to field-value to provide searching for  
string  
field content that might include non-ALPHANUM characters. The string-literal  
definition matches that of SQL-92."

For example:

```
(OfficeName="Smith && Smith, Inc.")
```

would represent a query with a String Literal.

Remaining Difficulty:

However, this proposal still does not allow the client to create queries similar to the example listed below where DMQL delimiters are actually part of the search string:

```
(OfficeName='Smith & Smith, Inc.', 'Anderson (Barney), Inc.', '*nderson, Inc.', 'Anderson*', '*nderson*')
```

Proposal:

As an enhancement to RETS Change Proposal 008, modify the grammar to allow strings which are composed of string-literals and, additionally, use single quotes instead of the double quotes specified:

1) Create new definition of SEARCHTEXT, similar to PLAINTEXT:

```
SEARCHTEXT ::= <any OCTET except CTLs, "*", ">">
```

2) Modify definition of field-value:

```
field-value ::= lookup-list | string-list | range-list | period  
| number
```

3) Modify definition of string-literal

```
string-literal ::= <'> 1*( *( SEARCHTEXT except <'> ) *( 2<'> ) *(  
SEARCHTEXT except <'> ) ) <'>
```

4) Add definitions for new versions of string-literal, similar to string-eq etc.:

```
string-literal-eq ::= 1*SEARCHTEXT  
string-literal-start ::= 1*SEARCHTEXT "*" "  
string-literal-contains ::= "*" 1*SEARCHTEXT "*" "  
string-literal-char ::= *SEARCHTEXT 1*"?" *SEARCHTEXT
```

5) Modify definition of string to include the new string-literals:

```
string ::= string-eq | string-start | string-contains |  
string-char | string-literal-eq | string-literal-start |  
string-literal-contains | string-literal-char
```

With these additions to the specification the example, introduced above, can now be successfully interpreted by the DMQL Parser:

```
(OfficeName='Smith & Smith, Inc.', 'Anderson, Inc.', '*nderson, Inc.',  
'Anderson*', '*nderson*')
```

Additionally, previous search strings are still considered to be valid:

(OfficeName=Smith\*, \*Barney\*)

---

## RETS Change Proposal 009: Client Software Validation

**Number:** 9

**Author:** Bruce Toback

**Organization:** OPT, Inc.

**Address:** 11801 N. Tatum Blvd. Ste. 142, Phoenix AZ 85028

**Telephone Number:** (602) 996-8601

**Email:** [btoback@optc.com](mailto:btoback@optc.com)

**Status:** Proposal

**Date:** June 16, 2002

**Version:** 1.0

### 1. Synopsis

This proposal replaces the current authorization scheme with one built on RFC 2617.

### 2. Rationale

The current authorization specification in RETS contains several ambiguities that have caused problems between client and server vendors. In addition, RETS servers currently do not have a method of validating authorized client software on a given system. The only current check of a User Agent's identity is the USER-AGENT in the http header, but client software can spoof the User-Agent header and use a valid UserID and Password to gain access. There is thus no secure way of allowing an MLS to enforce business rules regarding authorized user-agents.

RFC 2617 can solve both these problems: it is a widely-implemented industry standard and has a facility for client authentication. This change proposal specifies the replacement of the existing authorization and authentication with RFC 2617, and specifies the method of computing the `nonce` element in the `Authorization` header to allow for client authentication.

### 3. Proposal

#### 3.1 Specification Changes

Sections 4.3 - 4.5 of the specification are removed and the text is changed to refer to RFC 2617.

The server MAY require the client to authenticate itself using the `nonce` parameter on the `Authorization` header.

The `nonce` is calculated as:

```
nonce ::= <H(unquoted nonce-value ":" client-password-value)>
```

#### 3.2 Implementation Notes

Servers SHOULD implement this new feature as a transition by supporting clients both with and without client passwords.

### 4. Development Impact

Because the changes described in this proposal are optional, there is no forced development impact.

### 5. Compatibility

Because this change involves a change in the method gaining access, RETS clients that do not support the client password

will not be able to access a RETS server that requires it.

*Last changed 15 June, 2002, by [Webmaster](#).*

Comments on RCP009  
Leo Bijnagte

Client passwords as a method to allow hosts to enforce business rules regarding the use of the RETS interface by Client product independent of the Username/Password makes sense. The proposed solution uses a single variant (nonce) in the construction of its User-Agent Authorization; at minimum a more secure key (patterned after the proposed cnonce creation) should be used. However, it is not absolutely necessary to add yet another proprietary header. Instead I would propose that we formally include support for hosts to request the MD5 Digest Authorization type `qop=auth` as specified in RFC2617 and that we define a method for creating the cnonce used in the response to allow clients to incorporate a masked client password. Specifically,

### **Authentication Specification Change**

In section 4.3 change the definition of the WWW-Authentication Response Header to that of RFC2617:

If the server receives a Login request and an acceptable Authorization header (as defined in section 4.4) is not sent, the server MUST respond with a 401 Unauthorized status code and a WWW-Authenticate header as defined in RFC2617.

Hosts MUST request the Digest scheme.

Client-Passwords

Hosts that support a Client-Password MUST include the `auth qop=option`; hosts that do not require Client-Passwords MUST support responses both with and without a `qop` value.

### **Authorization Specification Change**

In section 4.4 change the definition of the Authorization Request Header to that of RFC2617:

When the client receives a 401 Unauthorized status code it is expected to retry the request, passing an Authorization header as defined in RFC2617.

Client-Passwords

If a client supports a Client-Password and the host includes the `auth qop=option`, it MUST respond with a value `qop=auth` and construct its cnonce as:

MD5 (User-Agent:Client-Password:RETS-Request-ID:nonce)

The RETS-Request-ID may be the empty string if the client does not use this optional header.

If a client does not support a Client-Password, it MAY use the option to ignore the qop-options and respond in the format retained from RFC2069 for backward compatibility.

Hosts that do not require Client-Passwords but that do allow qop-options MAY ignore the embedded information in the cnonce.

Establishment of Client-Passwords and the relationship to User-Agents are independent of the standard.

#### Reply Codes Specification Change

In section 4.14 add the reply code:

20037            Client-Password Invalid

#### **Impact**

RFC 2617 retains backward compatibility with the current implementation for both hosts and clients. Support for Client-Passwords remains optional (independently) for both hosts and clients.

C