



RETS Meeting Agenda

December 10-11, 2002

December 10

Time	Discussion Item
12:00 - 1:00	Lunch
1:00 - 1:30	Opening and introductions
1:30 - 2:45	Discussion: implementation experience with RETS 1.5; collection and discussion of required changes; compliance issues and recommendations; approval or setting approval date.
2:45 - 3:00	Break
3:00 - 4:00	Change proposal discussion; timing and possible dual support for 1.6 and 2.0.
4:00 - 4:20	New RETS user information site
4:20 - 5:00	Additional demonstrations

December 11

8:00 - 8:30	Breakfast
8:30 - 10:15	RETS 2.0 Objectives/Data Standard
10:15 - 10:30	Break
10:30 - 12:00	RETS 2.0 Architecture
12:00 - 12:45	Lunch
12:45 - 2:00	RETS 2.0 Architecture/Specification Planning/Schedule

Dress code for the meeting is Business Casual

Main
Mailing Lists
Developer
Documents
Comments
News
Resources

RETS Change Proposal 014: Metadata Backward Compatibility

Author: Alex Lancaster

Organization: Rapattoni Corporation

Telephone Number: (800) 722-7338

Email: alex@rapattoni.com

Status: Proposal

Date: November 18, 2002

Version: 1.5

1. Synopsis

This proposal adds the following:

- Optional Client Request Header Field: Metadata-Version to be included for all RETS Transactions.
- Additional RETS Reply Code: 20038, Meaning: Metadata Version ##.##.#### is expired and cannot be supported
- Changes the way the Metadata Version Tags section on pg. 4-4 of the RETS 1.5 spec can be returned in the Capability URL List.
- Allows a Server to be capable of supporting multiple Metadata Versions simultaneously.

2. Rationale

This proposal deals with keeping Metadata for existing Servers that have existing Clients in sync and provide an easy way to update/extend Metadata for an existing RETS installation without breaking any Clients.

During the last 18 months it has been our experience that updating a Server's Metadata while at the same time preventing all Clients using that Server from breaking isn't as easy as it sounds. Currently the RETS 1.5 spec says all RETS Clients must check Metadata Version on Login and "This is the minimum version of the metadata that the host will support. If the version of the metadata being used by the client is less than this version the client MUST retrieve the newer metadata from the host."

While this seems logical the reality of the situation is most Clients don't check the Metadata Version on Login and most are not equipped to deal with Metadata changes on the fly. This has caused us a considerable amount of difficulty with Clients who are not updating their Metadata and adjusting their product accordingly. The position of the RETS Spec is very server-centric and we at Rapattoni Corporation think it should be more client-centric.

It would be highly beneficial if there was a way for a Client to provide a Metadata Version parameter on Login which basically says "This is my current Metadata Version, can you support me?" If the Server can support the Client's current Metadata Version then it will say "Yes I can support that version." or "No, this is the minimum version I support: ##.##.####."

At this point, if the Server says "Yes I can support that version.", the Client can continue initiating RETS Transactions with its current Metadata version meaning every aspect of the Server will function as if it were on that version. This functionality includes returning default queries of only those fields on that specific Metadata version. If the Server says "No, this is the minimum version I support: ##.##.####." then the Client knows it MUST get new Metadata and make adjustments accordingly.

We at Rapattoni Corporation think this will be a significant improvement to the RETS Spec and it will help prevent existing Clients from breaking when Metadata changes take place on the Server. Additionally, it will make it easier to update Metadata on the Server and keep it separate from older (but still valid) Metadata versions. Rapattoni Corporation upgrades its Internet MLS product quarterly and new fields are always becoming available. In order for the RETS Server Metadata to stay current, Metadata changes must be easy to make and most of all, not break any existing Clients.

3. Proposal

This proposal adds an optional Client Request Header Field: Metadata-Version for all RETS Transactions. The Metadata-Version value is a standard RETS Metadata Version number (i.e. in the form ##.##.####). A Client MAY add this field to its HTTP Headers and a Server MAY use this value to adjust behavior internally to function as if it were working on the supplied Metadata Version number. If the Client does not supply a Metadata-Version Header value or if the Server does not support this optional argument, then the Server will operate according to its current Metadata Version as usual. If the Client supplies a Metadata Version value less than what the Server is willing to support, the Server MUST respond with an error message in the form:

```
<RETS ReplyCode=20038 ReplyText=quoted-string > CRLF
```

```
<MetadataVersion=##.##.#### ExpirationDate=YYYY-MM-DD /> CRLF
<MinMetadataVersion=##.##.#### ExpirationDate=YYYY-MM-DD /> CRLF
</RETS> CRLF
```

This proposal adds an additional RETS Reply Code: 20038, Meaning: Metadata Version ##.##.#### is expired and cannot be supported. This proposal changes the way the Metadata Version Tags section on pg. 4-4 of the RETS 1.5 spec can be returned in the Capability URL List after Login. An ExpirationDate value MAY be supplied immediately after the (Min)MetadataVersion tags as shown below:

```
MetadataVersion=##.##.#### ExpirationDate=YYYY-MM-DD
MinMetadataVersion=##.##.#### ExpirationDate=YYYY-MM-DD
```

Since a Server can now be capable of supporting multiple Metadata Versions simultaneously, a Server MAY explicitly list ALL the Versions it currently supports in the Capability URL List along with their ExpirationDates (if applicable) in order to avoid any ambiguity. A MetadataVersion that has no ExpirationDate value means that version is not currently set to expire. Example:

```
MetadataVersion=01.50.4000
MetadataVersion=01.50.3011 ExpirationDate=2003-06-01
MetadataVersion=01.50.2050 ExpirationDate=2003-03-01
MinMetadataVersion=01.50.1000 ExpirationDate=2003-01-01
```

4. Development Impact

This proposal only deals with an optional response argument added to the HTTP Headers for all RETS Transactions. A Server MAY choose to implement this proposal and it is still considered compliant if it does not.

If a Server does choose to implement this proposal, it will need to keep track of different sets of Metadata internally and have built in functionality that allows it to switch between different Metadata versions on the fly. This can be accomplished by adding a Version column to the Metadata tables internally that don't already have it, and modifying the queries against the database to include this column in the WHERE clause.

5. Compatibility

The RETS 1.5 spec allows for additional Optional Request Arguments as specified on pg. 4-2 of the RETS 1.5 spec. A Client MAY include the additional Metadata-Version HTTP Header for all RETS Transactions and a Server MAY make use of this value to adjust system behavior on the Server side to accommodate that Metadata version. This additional argument is optional and therefore will not cause any compatibility problems with Clients or Servers that choose not to support this feature.

It is also optional to include the ExpirationDate value after the MetadataVersion tag when the Server returns the Capability URL List after Login.

Last changed 25 November, 2002 by [Webmaster](#).

RETS Change Proposal 015: Allow Ordering of DMQL Query Responses

Author: Pete Clarno

Organization: Software & Vine, Inc.

Address: 825 Riverside, Suite 13, Paso Robles, CA

Telephone Number: (805) 227-1545

Email: pete@swvine.com

Status: Proposal

Date: July 24, 2002

Version: 1.5

1. Synopsis

This proposal adds a new order by argument to the Search Transaction, a new sortable field to content metadata and a new error code to query responses.

2. Rationale

Unlike the rich client model where bulk data is replicated on the client in a batch manner and specific queries are processed locally on demand, a thin client model assumes that a RETS server acts as a central data repository and delivers smaller amounts of data from more specific queries, on demand, during interactive sessions. While current DMQL syntax offers limit and offset parameters that would help reduce the dataset returned, it does not offer a sorting capability. This negates much of the usefulness of the limit and offset functionality, since results may be returned in any arbitrary order. By adding an order by parameter similar to that used in SQL, the client is assured of being returned the most meaningful data (according to client criteria) first. This substantially reduces the amount of data that must be served up, transmitted and consumed, thus substantially improving the user experience in a thin client environment.

3. Proposal

3.1 Add a new "order by" clause to the Optional Request Arguments of Section 7

A new subsection entitled "Order By" should be added to the Optional Request Arguments of Section 7.4. The OrderBy argument requests the server to provide the results of the query in a specified sequence.

```
Order By ::= (field [Asc|Desc]) *(,field [Asc|Desc])
```

This parameter is used to determine which field(s) are to be used to determine the sequencing of results returned by the query. The field(s) specified MUST be a system name or standard name field (as determined by the StandardNames flag) for which the Sortable Field in the Metadata Content Table is set to true. (See proposal 3.2 below for additional information on the Sortable Field.)

An optional directional indicator MAY be used to determine whether the list is sorted in Ascending (Asc) or Descending (Desc) order. If omitted, ascending order is assumed.

If the Server indicates that a field is Sortable in the metadata content, it MUST honor the client request for sorting by a single field. However, if the client requests multiple sorts columns which may create an undo burden on the server, the Server MAY return an error code or simply sort on the first field requested. If the field requested is not a Sortable field, the Server MUST return an error code. (See proposal 3.3 for suggested changes to error codes.)

: The collation sequence is implementation specific and not controllable by the client.

Example: Order By=ListPrice Desc

Instructs the server to return the result set with the highest priced property at top of the list.

3.2 Add a new "sortable" value to the Table metadata

Field Name	Content Type	Descriptions
Sortable	Boolean	A truth-value which indicates that the field is sortable

3.3 Add two new Reply Codes to Section 7.8

Reply Code Meaning

TBD	The requested Order By field is not a sortable value
TBD	The sort requested is too complex to be honored

4. Development Impact

The impact on Server developers varies from modest (adding a value to the table metadata) to rather extensive (developing an efficient sort mechanism for their server) depending upon their level of support for the proposed change. At minimum server may elect to simply revise the metadata to indicate that no fields are sortable, thus indicating that the OrderBy clause is not supported. Client developers wishing to take advantage of the sort capabilities would need to revise their program accordingly.

5. Compatibility

This feature is envisioned as a dot release enhancement to the DTD. Servers that do not support this feature would return a metadata version tag indicative of that fact. Clients wishing to support multiple metadata versions would need to adapt to this information.

Last changed 26 November, 2002 by [Webmaster](#).

RETS Change Proposal 016: Lookup Type Meta Update

Author: Wantao Zhou

Organization: Interealty Corporation

Address: 7018 Lougheed Highway, Burnaby, BC V5A 1W2

Telephone Number: (604) 422-2439

Email: wantao.zhou@interealty.com

Status: Proposal

Date: December 4, 2002

Version: 1.0

1. Synopsis

This proposal makes the following changes in RETS 1.5 draft 1, section 11.4.3:

- Expand size of the LongValue field.
- Add usage guidance to the LongValue and ShortValue fields in Lookup Type metadata.
- Add a new optional identity field.

2. Rationale

The current standard defines three fields in Lookup Type metadata: LongValue, ShortValue, and Value. The standard further clearly specifies the Value field is "[t]he value to be sent to the server when performing a search". However, the standard only noted that the use of both LongValue and ShortValue fields are implementation-defined. The vagueness of the description for the two fields caused confusion on whether the values SHOULD be unique, and choices for selection and display. Also the current standard specifies both LongValue and ShortValue have maximum of 32 characters. The size of LongValue is too restrictive in practice and need to be expanded.

Client vendors are also looking for a persistent field in Lookup Type meta that can keep its value regardless changes of other fields when the meaning of the Lookup Type item doesn't change. This field serves as an identity of the Lookup Type item. None of the three fields currently specified in the standard is the best choice because of their potential volatility in the present usage scenario.

3. Proposal

The Content Type of the LongValue field be modified to 1*100TEXT.

The Description of LongValue field be modified to read: "The value of the field as it is known to the user. This is a localizable, human-readable string. The values of this field do not need to be unique within one Lookup. A sample use of the field is for display in report."

The Description of the ShortValue field be modified to read: "An abbreviated field value that is also localizable and human-readable. The value SHOULD be unique within one Lookup. A sample use of this field is for selection in user interface."

Add a new optional identity column for Lookup Type:

Field Name	Content Type	Descriptions
ID	1*32ALPHANUM	The field identifies the Lookup Type item. The value of the field MUST be unique within one Lookup, and SHOULD not change as long as the meaning of the item doesn't change regardless changes of other fields.

4. Development Impact

The increase of size restriction on LongValue allows server vendors to send more descriptive information about the lookup type item. Client vendors need to increase the string size for the LongValue field in code and UI layout.

Server vendors may need to modify the internal field mapping based on the additional description on LongValue and ShortValue fields.

Server vendors may need to return a unique ID field. Client vendors need to use this field to identify Lookup Type items if this field is available.

5. Compatibility

Both server and client are required to support RETS 1.5.

Last changed 08 December, 2002 by [Webmaster](#).

RETS Change Proposal 017: RETS Namespace

Author:Bruce Toback

Organization:RETS Document Editor

Telephone Number: (602) 996-8601

Email: btoback@optc.com

Status: Proposal

Date: December 8, 2002

Version: 1.0

1. Synopsis

The RETS DTD should define a RETS namespace.

2. Rationale

The RETS DTD does not currently define a namespace. By adding all standard elements to a "rets" namespace, local extensions will be cleaner and interoperability with other standards will be enhanced.

3. Proposal

The name of each field in the RETS DTD should have "rets:" prepended.

4. Development Impact

Parsers will need to be changed to recognize the namespace, but no structure changes will be required.

5. Compatibility

Compatibility issues can be resolved during DTD version negotiation between clients and servers.

Last changed 08 December, 2002 by [Webmaster](#).