



RETS Meeting and Plugfest, February 1-20, 2003

Home

Documents

Resources

Mailing Lists

Working Groups

Comments

Links

1. [Agenda](#)
2. [Plugfest Participant Guide](#)

Agenda

February 19 (Plugfest)

- | | |
|----------|---|
| 8:00 AM | Breakfast |
| 8:30 AM | Opening session:
Introduction/Orientation
Demonstration of compliance tools |
| 9:00 AM | Technical working session
Break refreshments available at 10:00 AM |
| 11:30 AM | Interim checkpoint and progress reports |
| Noon | Lunch |
| 1:00 PM | Resumption of working session
Break refreshments available at 2:30 PM |
| 1:00 PM | RETS Marketing Interest Group meeting |
| 4:00 PM | Wrap-up |

February 20 (General Meeting)

- | | |
|----------|---|
| 8:00 AM | Breakfast |
| 8:30 AM | Opening session:
Introduction
Demonstration of compliance tools
Status report from plugfest
Discussion of results |
| 9:15 AM | Compliance Workgroup Report/Discussion |
| 9:45 AM | Metadata Workgroup Report/Discussion |
| 10:30 AM | Break |
| 10:45 AM | Transport Workgroup Report/Discussion |
| 11:15 AM | Marketing Workgroup Report/Discussion |
| Noon | Lunch |
| 1:00 PM | Discussion of outstanding change proposals |
| 3:00 PM | Break |
| 3:15 PM | Discussion of outstanding change proposals |
| 4:00 PM | Wrap up |

Plugfest Participant Guide

The RETS Plugfest is a technical working session, in the mold of such events for other standards such as [LDAP](#) and [Serial ATA](#). This means it's an event intended primarily for software developers working on RETS-compliant clients and servers.

What will be provided. We will have Ethernet 10BaseT local area network connections, with direct high-speed access to the Internet. In addition, we will have a system set up with the RETS compliance test platform (an NAR-sponsored software system written by [Avantia](#)), and a protocol analyzer that can be used to find and document interoperability problems.

What to bring. If you're developing a client, you'll get the most benefit from the Plugfest if you bring your code on your

laptop, and come prepared to discuss your implementation with other developers. If you have user-level documentation, it may help to have that along as well. If your laptop requires a card or cables to connect to 10BaseT Ethernet, bring those too.

If you're a server developer, you should come equipped with one or more login names for a test RETS server running your server software. You can access your system through the Internet connection at the Plugfest; we will provide firewall configuration information as soon as it's available (which may not be until the morning of the event).

What to expect. If the Plugfest is 100% successful, by the end of the day, every client will have successfully exchanged data with every server. A list of suggested tests will be posted here before the start of the event, and will be distributed at the event as well. At the midday checkpoint session, just before lunch, participants will be given the opportunity to note additional tests that should be performed as determined by their own experiences, and to start collecting areas of the standard that need clarification.

If a client/server pair encounters difficulty, they can use the protocol analysis tools or the compliance test system to help iron out the difficulties. The test system is scriptable, so if you discover a new and interesting incompatibility, a new script can be created in order to help others avoid encountering the same problem. Representatives from Avantia, who created the system, will be available to answer questions and assist in using it.

By the end of the day, participants should have a good idea of how close we are to the goal of universal interoperability. (We may have a prize for the client and server that have managed to communicate with the largest number of other products.) During the wrap-up, we will finish collecting experiences that will help to correct the language of the standard. This report will be presented to the general meeting on the following day.

Last changed 6 February 2003 by [Webmaster](#).

RETS Change Proposal 20: Special Search Token .ALL.

Author: Brita Brodin

Organization: WyldFyre Technologies

Telephone Number: (408) 369 - 8900

Email: Brita@wyldfyre.com

Status: Proposal

Date: December 9, 2002

Version: 1.0

1. Synopsis

Currently, if a lookup field is required, the RETS search request has to contain every value defined for the lookup in the metadata. The RETS Query Language should contain a special search token such as **.ALL.** which means "all values defined for the lookup".

2. Rationale

If a lookup has a long list of acceptable values, the query string gets to be very long and some RETS servers are not capable of parsing such long search requests. Also, it is inefficient for a server to have to parse such a request since sending all values for a lookup is logically equivalent to not sending the field at all. However, if the field is defined as required in the metadata, the client must send it and therefore, it would be much more efficient to use a search token that signifies all values. For example, if Area is defined as required, it would be much more efficient to send a request such as `QUERY=(Area=.ALL.)` than `QUERY=(Area=A|B|C|D|E|F...)`.

3. Proposal

Section 7.7 Query Language should be amended to show the following:

Lookup := <any legal ALPHANUM value for the field as defined in the metadata>.**.ALL.**

4. Development Impact

RETS parsers will need to be changed to recognize the **.ALL.** token.

5. Compatibility

RETS 1.5 Servers would be required to support the **.ALL.** token to be compatible.

Last changed 10 December, 2002 by [Webmaster](#).

RETS Change Proposal 22: Boolean Data Type

Author: Maggie T. Diaz

Organization: WyldFyre Technologies

Telephone Number: (408) 369 - 8900

Email: mdiaz@wyldfyre.com

Status: Proposal

Date: December 9, 2002

Version: 1.5

1. Synopsis

This change request is to clarify specification of Boolean data type in Metadata-Table

This description should clarify via an example, that the MaximumLength is the maximum visible length. The maximum internal storage length should NOT be provided.

2. Rationale

Although RETS 1.0 describes Boolean data type as "A truth-value, stored as 1 for true and 0 for false," there's some confusion on specifying such data type in metadata-table. Should value be Yes/No, Y/N, True/False, T/F, or 0/1?

3. Proposal

If a field is specified as Boolean Data Type, the Interpretation MUST be "Lookup" in Metadata-Table. The corresponding Lookup_Type MUST have the following structure:

For example, a LookupMulti would have

```
<COLUMN> LongValue ShortValue Value </COLUMN>
<DATA> (AnyValue) (AnyValue) 1 </DATA>
<DATA> (AnyValue) (AnyValue) 0 </DATA>
```

The Value column in the example above references the description of stored value 1 for true and 0 for false.

4. Development Impact

Development impact depends on each 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 1.5 MUST follow RETS 1.5 standard.

5. Compatibility

None.

6. Proof/Need of Concept Examples

None.

Last changed 10 December, 2002 by [Webmaster](#).

RETS Change Proposal 23: Update MaxChoice

Author: Maggie T. Diaz

Organization: WyldFyre Technologies

Telephone Number: (408) 369 - 8900

Email: mdiaz@wyldfyre.com

Status: Proposal

Date: December 9, 2002

Version: 1.5

1. Synopsis

This proposal is to include a new column in Update_Type section for maximum choice allowed in update transactions.

2. Rationale

A new column is needed in Update_Type section for RETS clients and servers to specify the maximum allowed entries for Update. RETS vendors have no way of specifying the maximum choice and use another column in Metadata-Table that actually serves a different purpose already.

3. Proposal

Create a new column called MaxChoice in Update_Type. RETS vendors MUST use this column to specify the total number of entries that can be selected in a LookupMulti fields.

4. Development Impact

Development impact depends on each current RETS client and server implementations, but should not have any impact on future RETS client and server implementation. Each existing RETS server and client prior to passing of this proposal has the option to change accordingly to be RETS compliant. Future RETS server and client implementation after approval of RETS 1.5 MUST follow RETS 1.5 standard.

5. Compatibility

None.

6. Proof/Need of Concept Examples

None.

Last changed 10 December, 2002 by [Webmaster](#).

RETS Change Proposal 25: Update Validation Flag

Author: Maggie T. Diaz

Organization: WyldFyre Technologies

Telephone Number: (408) 369 - 8900

Email: mdiaz@wyldfyre.com

Status: Proposal

Date: December 9, 2002

Version: 1.5

1. Synopsis

This proposal is to include a new validation flag for update transactions.

2. Rationale

RETS 1.0 provides insufficient validation requirement between RETS clients and servers. The current validate-flag options are either 1 or 0. A new validation flag is required to allow servers to better validate all fields sent by clients.

3. Proposal

Add a new flag, `validate-flag=2`, for servers to validate all fields. This will allow RETS clients and servers to use the following validate-flags:

If `validate-flag = 0`, RETS servers MUST update record on server; there are no errors. If `validate-flag = 2`, RETS Servers MUST populate and return data for all fields with Auto-pop attribute. If `validate-flag =1`, RETS servers MUST check for all errors and return any errors to client.

4. Development Impact

Development impact depends on each 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 1.5 MUST follow RETS 1.5 standard.

5. Compatibility

None.

6. Proof/Need of Concept Examples

None.

Last changed 10 December, 2002 by [Webmaster](#).

RETS Change Proposal 25: Update Response Body Format

Author: Maggie T. Diaz

Organization: WyldFyre Technologies

Telephone Number: (408) 369 - 8900

Email: mdiaz@wyldfyre.com

Status: Proposal

Date: December 9, 2002

Version: 1.5

1. Synopsis

This change proposal is to add clarification on Update Response Body Format.

2. Rationale

Section 10.5 shows the body of the update response format, which includes transaction-id-tag, column-tag, compact-tag, and error block. However, a clarification is needed on the syntax and usage of each tag, similar to whatOs described in Section 7.6 Search Response Body Format.

3. Proposal

3a.

Include in RETS 1.5 a description for each tag. Let column-tag and compact-tagOs definition and syntax (optional or required notation) be the same as in Section 7.6 Search Response Body Format.

3b.

Provide a description and clarification on syntax of transaction-id-tag and error block. In ERRORBLOCK, use a space delimiter within <ERRORDATA> to separate each error information.

Example: 1*(<ERRORDATA> field error-num *SP error-offset *SP error-text</ERRORDATA>CRLF)

4. Development Impact

This proposal is an enhancement on the specification and should not impact current or future RETS client and server implementations.

5. Compatibility

None.

6. Proof/Need of Concept Examples

None.

Last changed 10 December, 2002 by [Webmaster](#).

RETS Change Proposal 27: Special Search Token .NULL.

Author: Brita Brodin

Organization: Wyldfyre Technologies

Telephone Number: (408) 369 - 8900

Email: Brita@wyldfyre.com

Status: Proposal

Date: December 9, 2002

Version: 1.0

1. Synopsis

The RETS Query Language should contain a special search token such as **.NULL.** to search for fields that have no explicitly assigned value or have a blank value.

2. Rationale

Some MLS system fields are populated based on the status of the listing. For example, an active property would not have a sold date value.

For example, if a user wants all active properties along with any closed properties from the last 2 years in area 201, the following RETS query could be used:

```
Query=(Area=201),(liststatus=ACT)|((liststatus=CLOSD),(closeddate=2001-01-01-2001-06-01))
```

The problem with this type of query is that the RETS client must know which date field is associated with which status. This type of information is not available in the RETS metadata.

Alternatively, the RETS client could submit a request such as:

```
Query=(area=201),(liststatus=|ACT,CLOSD),((closeddate=2000-12-09-2002-12-09)|(closeddate=.NULL.))
```

In this example, the RETS client does not have to know which dates are associated with which status. The client has to know that a date field is status dependent but not with which status. Furthermore, the RETS client could be configured to always submit a NULL value for all standard date fields.

3. Proposal

Section 7.7 Query Language should be amended to show the following:

```
field-value := lookup-listrange-liststring-list..NULL.
```

4. Development Impact

RETS parsers will need to be changed to recognize the **.NULL.** token.

5. Compatibility

RETS 1.5 Servers would be required to support the **.NULL.** token to be compatible.

RETS Change Proposal 028: Omnibus Metadata Update

Number: 28

Authors: Metadata Workgroup

Status: Proposal

Date: January 27, 2003

Version: 1.0

1. Synopsis

This proposal augments the RETS metadata mechanism in several ways in order to improve the effectiveness of metadata updates and reduce the burden on clients when updating metadata.

2. Rationale

Experience to date has shown that the current metadata management scheme sometimes results in problems in deployment situations. These include failures of clients to operate because of metadata changes, and an inability of servers to make minor metadata changes because of client lags. These proposals are intended to address this difficulty.

It has also been noted that several of the length restrictions on metadata values are too small and no longer needed in the current computing environment.

3. Proposal

3.1 Specification Changes

Several changes are proposed to metadata tables and metadata management specifications.

3.1.1 Version Management

Currently, RETS specifies the use of version numbers for version management. Version numbers are required to increase when any element changes, with ripples through the hierarchy. The version numbering mechanism is specified with a <major>.<minor>.<release> convention, but the convention is not binding.

It is proposed that version numbering as a control mechanism be deprecated, and that metadata advertising and acceptance be based instead on the modification timestamp for the metadata. Section 4.7.3 will be modified to include `MetadataTimestamp` and `MinMetadataTimestamp` values; these are to be compared with the `Date` attribute of the `System` metadata to determine whether the client has an acceptable version of metadata.

Section 4.7.3 of the specification will be further amended to indicate that a client MAY retrieve the newer metadata version from the host if it finds that its stored version is earlier than the `MinMetadataTimestamp`. If the client chooses not to obey this advisory, then query results are not guaranteed to be correct and may result in errors or incorrect information being supplied to the client.

Section 4.4 will be amended to include an optional parameter, `SavedMetadataTimestamp`. The value is strictly advisory for the server; the server MAY use this to adapt to a metadata version earlier than its advertised `MinMetadataTimestamp`, but is not required to do so. It may also use the information for logging or any other purpose.

3.1.2 Metadata Comparisons

In order to facilitate metadata comparisons, a known-invariant field, named `MetadataEntryID`, is to be added to every metadata row in every table except `System`, `Resource` and `Class`. This field, a 32-character string, is to be maintained by the server and can be used by the client to indicate that a particular row has the same logical meaning despite changes in its fields. For example, the invariant field may be used to detect changes in the `SystemName` in the `Table` metadata.

It is not the intent of the specification to force servers to create a new logical data item. If one of the metadata fields already serves this purpose, its value can be duplicated in the `MetadataEntryID` field. The client MUST treat this field as opaque, and not try to interpret its contents.

3.1.3 Length changes

In general, the objective is to avoid having the standard impose arbitrary limits on human-readable fields.

Table	Field	Old	New	Rationale
METADATA-CLASS	VisibleName	32	64	Old length was inadequate for a prose description.
	Description	64	128	Old length was inadequate for a prose description.
METADATA-TABLE	ShortName	24	64	Old length was inadequate for a prose description.
	LongName	32	256	The field should be able to contain arbitrary text.
METADATA-OBJECT	MIMEType	24	64	Old length was inadequate for some MIME type designations.
	VisibleName	32	64	Old length was inadequate for some prose descriptions.
	Description	64	256	The field should be able to contain arbitrary text.
METADATA-LOOKUP_TYPE	LongValue	32	128	The field should be able to contain arbitrary text.
METADATA-SEARCH_HELP	Value	128	1024	The field should be able to contain arbitrary text, including examples.
METADATA-UPDATE_HELP	Value	128	1024	The field should be able to contain arbitrary text, including examples.

4. Development Impact

This proposal includes changes that are mandatory and may have development impact, particularly on servers. The requirement to create an invariant field for metadata may not be consistent with metadata management procedures in some server environments.

5. Compatibility

This change proposal includes several changes that may result in incompatibility. Clients that implement this change may be unable to operate with an unchanged server because of the lack of the mandatory `MetadataEntryID` field. Servers that implement these changes may be incompatible with clients that do not because of metadata version management issues, and because the new server may take advantage of the longer field lengths available with this change proposal.

Last changed 27 January, 2003, 0700 MST by [Bruce Toback](#).

RETS Change Proposal 29: Distributed Database Enhancement

Author: Steve Clarke

Organization: Interealty

Telephone Number: (408) 369-8900

Email: Steve.Clarke@interealty.com

Status: Proposal

Date: February 13, 2003

Version: 1.1

1. Synopsis

This proposal provides a new RETS transaction, specifically for the purpose of Distributed Database (DDB) support.

2. Rationale

A "Distributed Database" client is a fairly common class of real estate software that is preferred by many real estate professionals. This class of client software fundamentally requires the ability to maintain an up-to-date, local copy of the entire (or a cross section) of the MLS database in order to support off-line processing and searching. Currently the RETS standard does not specifically deal with this type of functionality. Under the current standard, a "DDB" client must make certain assumptions about the server's schema and/or take advantage of proprietary server implementations in order to provide the fundamental functionality of a distributed client. Furthermore, DDB implementations that use combinations of the traditional RETS Search transaction may be causing serious inefficiencies on the server side.

- A DDB client must make the assumption that there exists a field like "LastTransactionDate" in the RETS servers' schema, in order to fetch data incrementally. However, *LastTransactionDate* is NOT a required field in the RETS specification.
- A DDB client must make the assumption that there exists a field like "LastImageTransactionDate" in the RETS servers' schema, in order to fetch images incrementally. However, *LastImageTransactionDate* is NOT a required field in the RETS specification.
- Since deleted records are not returned by a typical RETS query, a DDB client may be extremely inefficient in determining which records have been deleted from the system or reclassified to statuses that fall outside of the DDB criteria.
- When downloading into a distributed database, a DDB client often will violate a server's "*maximum search result*" limit that has been established by the MLS board. In section 7.4.3 the RETS spec states: '*Client implementers should be aware that some server implementations might not honor the request to disable the limit.*' This proposal would require servers that do support the DDB transaction set **MUST NOT** enforce record limits on the results. Servers would, however, have some other means of limiting DDB requests.

This proposal is intended to address the above difficulties as well as provide a far more efficient means for supporting a DDB client.

3. DDB Transaction Proposal

3.1 Request Format

3.1.1 Search Type and Class

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

SearchType ::= ResourceID

The type of search to perform as discussed in Section 7.1 and defined in the Metadata (see section 11.2.2).

*Class ::= 1*32ALPHANUM*

This parameter is set to a value that represents the class of data within the *SearchType*, taken from the *Class* metadata (section 11.3.1). 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.

3.1.2

The specification consists of the query itself together with a designation of the query language.

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

This is the query that specifies the subset of the server's database that is being maintained in the DDB for this Class. The query is specified in the language described in Section 7.7.

QueryType ::= DMQL2

An enumeration giving the language in which the query is presented. The only valid value for RETS/1.5 is "DMQL2" which indicates the query language described in Section 7.7

3.1.3 Last Successful DDB Update Timestamp

This optional argument specifies the time of the last successful DDB update. If this argument is provided, then the response will only include records that have actually been affected or changed since that timestamp. If this argument is not provided, then the response will include all appropriate records, suitable for the complete initialization of the DDB. It is **HIGHLY RECOMMENDED** that clients actually provide the *LastUpdateDate* whenever possible. This will increase efficiency on the server and may avoid server-side throttling mechanisms.

LastUpdateDate ::= <the date of last previous successful DDB update>

3.1.4 Field Delimiter

A server may designate a particular OCTET to be used as a delimiter for separating entries in the DATA returned by the response. The octet should be chosen to avoid the need to escape data within a record. A tab character (an octet with a binary value of 9) is the default delimiter unless another is specified as part of the transaction.

field-delimiter ::= HEX HEX

Response Format

3.2.1 <DDB-ACTIVITY> tag

This tag wraps the DDB transaction response.

Attributes:

- Class - Simply echoes the requested class.
- Date - tells the client the date through which the DDB activity is being reported. This date SHOULD be stored on the client and SHOULD be submitted as the LastUpdateDate in the subsequent DDB transaction request. The server MUST guarantee that if the client submits this date upon its subsequent DDB request, that NO transactions will be missed.
- ReplyCode - Result status of the request. See section 3.2.6.

3.2.2 <DDB-TRANSACTION> tag

This tag may be repeated up to four times, each for different types of transactions. Each instance of this tag will also contain a <DATA> tag, which will contain the appropriate KeyField values. The KeyField values in the <DATA> section will be delimited by the field-delimiter as specified in the request (see section 3.1.4). If a particular section (Type) of DDB activity contains no records, then that instance of the <DDB-TRANSACTION> tag may be omitted completely.

Attributes:

- Count – Identifies how many records (KeyFields) are contained within this section.
- Type – Specifies what type of DDB transaction is being returned:
- ReplyCode - Result status of the request. See section 3.2.6.

DeletedRecord Records that have been deleted from the system, since the given LastUpdateDate parameter, given the specified Query.

ChangedRecord Records that have been modified in the system, since the given LastUpdateDate parameter, given the specified Query.

ChangedImage Records that have had changes (additions/modifications/deletions) to their associated images, since the given LastUpdateDate parameter, given t-e specified Query.

NoLongerMatch Records that no longer match the criteria specified in the Query, since the given LastUpdateDate parameter. This means records that have "fallen out" of the specified criteria since the provided date.

3.2.3 Sample DDB Response

```

<DDB-ACTIVITY ReplyCode = "0" Class = "RESIDENTIAL" Date="Sat, 20 Mar 2002
12:03:38 GMT" >
<DDB-TRANSACTION Type= "DeletedRecord" Count = "4">
<DATA>123111|123222|123333|123444</DATA>
</DDB-TRANSACTION>
<DDB-TRANSACTION Type= "ChangedRecord" Count = "3">
<DATA>234111|234222|234333</DATA>
</ DDB-TRANSACTION>
< DDB-TRANSACTION Type = "ChangedImage" Count = "3">
<DATA>111111|111222|111333</DATA>
</ DDB-TRANSACTION>
< DDB-TRANSACTION Type = "NoLongerMatch" Count = "3">
<DATA>222111|222222|222333</DATA>
</ DDB-TRANSACTION>
</DDB-ACTIVITY>

```

3.2.4 Limitations

The DDB response will only provide the KeyField of the relevant records. Subsequent interactions would be required to fetch the data or images based on the KeyField values.

3.2.5 Throttling

The use of this new transaction would imply that the client wishes to exceed the search result limitations of the server. The server MUST either allow this, or must fail the transaction altogether. Other forms of throttling would be allowed on the server. For example, a server may restrict the frequency of DDB transactions from a particular client, or may restrict such transactions during certain hours of the day. Perhaps DDB requests that do not provide a LastUpdateDate argument would be more restricted than those that do provide a LastUpdateDate.

Note: See section 3.2.6 for reply codes that reflect throttling. There may be a need for an additional transaction for the client to query the server for details of DDB transaction restrictions.

3.2.6 Reply Codes and Errors

Reply Code Meaning

0	Operation Successful
20801	Server does not support DDB transaction.
20802	Unable to complete DDB transaction at this time (throttling restriction, try again later).
20803	DDB transactions not supported for this resource type.
20804	Invalid Search Criteria Field.
20805	No DDB Activity since specified LastUpdateDate.

4. Development Impact

The changes in this proposal would be OPTIONAL for implementation by server vendors. Client applications could continue to support their DDB functionality by using combinations of the regular search transaction. However, with the implementation of this new DDB transaction, server vendors may choose to enforce search result record limits designed into the regular search transaction. Doing so may render any client DDB application that uses the regular search transaction non-operational.

This proposal is an enhancement on the specification and should not impact current or future RETS client and server implementations.

5. Compatibility

If this approach is adopted, there should be a transitional time period where the regular search transaction can still be used for DDB support, without enforcement of record limits.

Last changed 13 February, 2003 by [Webmaster](#).

RETS Change Proposal 30: Hotsheet

Author: Maggi Diaz

Organization: WyldFyre

Address: 900 East Hamilton Ave., Suite 500, Campbell, CA 95008

Telephone Number: (408) 369-8900

Email: maggie@wyldfyre.com

Status: Proposal

Date: January 20, 2003

Version: 1.5

1. Synopsis

This change proposal is to (1) add Hotsheet as another Resource in the metadata structure, (2) add Hotsheet as another Search Type, and (3) add Hotsheet to the RETS DTD.

2. Rationale

RETS vendors need a standard to implement Hotsheet that is currently not in RETS

3. Proposal

3a. Include Hotsheet as another Search Type in Chapter 7, Section 7.1 Search Types, and describe as follows:

Hotsheet A Hotsheet Search is a search against a property class database/table.

3b. Include Hotsheet as another Resource in Section 11.2.2. Resource, Table 11-1 Well-Known Resource Names as follows:

Resource	Name	Purpose
Hotsheet		A resource that contains information about property listings with price and status changes from a resource property class table.

3c. Include Hotsheet in Real Estate Data (REData) Transaction DTD, Chapter 1, Section 1.4.8 Packaging Elements, as follows:

```

REData
  REProperties?
    ResidentialProperty*
    CommonInterest*
    LotsAndLand*
    MultiFamily*
    TaxData*
  REOffices?
    REOffice+
  REAgents?
    REAgent+
  REOfficeRosters?
    REOfficeRoster+
  REActivities?
    Activity+
  REProspects?
    Prospect+
  REPUBLICRecords?
    RETax+
  REHistories?
    REPropHistory+
    REPropEntry+
  REHotsheet?
    REHotsheetRecord+

```

3d. Include Hotsheet in Real Estate Data (REData) Transaction DTD, Section 1.5 Comments on Field Usage, as follows:

The ListingID is the intended container for the data in the KeyField of the Hotsheet Resource as defined in the Metadata.

3e. Include Hotsheet in Real Estate Data (REData) Transaction DTD, Chapter 2 DTD, as follows:

```
<!-- PACKAGING ELEMENTS -->
<!ELEMENT REHotsheet (REHotsheetRecord+) >
```

3f. Include Hotsheet in the Real Estate Data (REData) Transaction DTD, Chapter 2 DTD, as follows:

Note1: The REHotsheetRecord utilizes the 'PropertyRecord' element defined in the 'Listing History' section. That element should be moved from the 'Listing History' section to the general 'Compound Elements' section.

Note2: The REHotsheetRecord utilizes the 'ChangeType' element defined in the 'Listing History' section. That element should be modified as shown below and moved from the 'Listing History' section to the general 'Compound Elements' section.

```
<!ELEMENT ChangeType (#PCDATA) >
  <!ATTLIST ChangeType (#PCDATA) >
    PriceChange (Increase | Decrease | None)
    StatusChange (Yes | No)
    System (Yes | No) "No" >

<!-- Hotsheet Elements -->
<!ELEMENT PreviousListPrice (#PCDATA) >
<!ATTLIST PreviousListPrice (#PCDATA) >
  Type (INTEGER | FLOAT) "INTEGER"
  CurrencyCode CDATA "USD" >

<!ELEMENT PreviousStatus (#PCDATA) >
<!ATTLIST PreviousStatus (#PCDATA) >
  Status (Active | Closed | Expired | OffMarket | Pending) >

<!-- Compound Hotsheet Elements -->
<!ELEMENT REHotsheetRecord (ListingID, ChangeType, PreviousListPrice?,
PreviousStatus?, PropertyRecord?, ModificationTimestamp?)>
```

3g. Include Hotsheet in Real Estate Data (REData) Transaction DTD, Chapter 3 Sample Data, as follows:

```
3.7 Hotsheet
<REData>
  <REHotsheet>
    <REHotsheetRecord>
      <ListingID>4839424</ListingID>
      <ChangeType PriceChange="Increase" StatusChange="No">PriceIncrease</ChangeType>
      <PreviousListPrice>489000</PreviousListPrice>
      <PreviousStatus Status="Active">New</PreviousStatus>
      <PropertyRecord>
        <ResidentialProperty>
          <Listing>
            <StreetAddress>
              <StreetNumber>720</StreetNumber>
              <StreetDirPrefix>E</StreetDirPrefix>
              <StreetName>Vineyard</StreetName>
              <StreetSuffix>Ln</StreetSuffix>
            </StreetAddress>
            <ListingData>
              <ListPrice>479000</ListPrice>
            </ListingData>
            <SalesData>
              <ClosePrice>0</ClosePrice>
            </SalesData>
            <MLSInformation>
              <ListingStatus Status="Active">Active</ListingStatus>
              <StatusChangeDate>2003-01-31T8:00:00</StatusChangeDate>
              <ModificationTimestamp>2003-01-31T15:25:40.897</ModificationTimestamp>
            </Listing>
          </ResidentialProperty>
        </PropertyRecord>
      <ModificationTimestamp>2003-01T15:25:40.897</ModificationTimestamp>
    </REHotsheetRecord>
  </REHotsheet>
</REData>
```

4. Development Impact

This proposal adds another Resource and Search Type in RETS. There should not be an impact on RETS clients and servers.

This proposal is an enhancement on the specification and should not impact current or future RETS client and server implementations.

5. Compatibility

None

5. Proof/Need of Concept Examples

None

Last changed 13 February, 2003 by [Webmaster](#).