NAR MLS/Client Data Standards Whitepaper ("Summary")

RETS Version 1.0
April 14, 1999

Larry Colson
Moore Data Management Services

Introduction

The National Association of REALTORS® (NAR) recognizes that technology has the ability to dramatically change the way that real estate transactions occur. The explosion of interest in the Internet has given rise to many web services that have increased the need for data interchange, and has created a number of questions in the minds of both REALTORS® and clients regarding the impact of the Internet on the real estate industry.

NAR recognizes that the REALTOR® is the critical part of the real estate transaction, and is committed to keeping the REALTOR® at the center of that transaction. To that end, NAR has begun the development of a set of data interchange standards that will facilitate the free flow of information between various users of the real estate transaction data.

In February 1999, a task force began to address these data standards. This group included representatives of MLS vendors, client application vendors, MLS operators and industry consultants, and collectively represented a vast level of real-world experience in design, development and implementation of real estate information systems. This group's mission was defined simply and succinctly:

> **Define detailed technical standards for Internet based communication and exchange of real estate property information.**

The proposed technical standards have been documented in detail in the **Real Estate Transaction Specification (RETS)**, which is intended as the definitive source for vendors or users who wish to use the transaction set. For each transaction, RETS discusses the parameters along with the expected behavior or results that clients and hosts should expect from one another, including error codes. In describing certain features of the transactions, the specification uses the definitions of MUST, SHOULD and MAY that have been widely accepted within the Internet standards document community to designate features that are required (MUST), recommended (SHOULD), and optional (MAY). For full compliance with the standard, implementations must adhere to all of the MUST features, and any SHOULD and MAY features included must conform to the RETS requirements. The standard is rather specific and very technical, due to the nature of the topic and the need for clarity and completeness.

The members of the task force included

Larry Colson	Moore Data Management Services
Tom Curtis	Metro MLS
Kevin Knoepp	GTE Enterprise Solutions
Tom McLean	Resolution Software Consulting, Inc.
Dan Musso	WyldFyre Technologies, Inc.
Tony Salvati	Grant Thornton
Errol Samuelson	RealSelect, Inc.
Alan Shapiro	Interealty
Dale Stinton	National Association of REALTORS®
Bruce Toback	OPT, Inc.

Terminology

For purposes of this discussion, the following definitions will be used.

Host — A computer that provides data to clients. All MLS systems would be considered hosts. A particular MLS host, in certain cases might also be considered a client. A "host" may also be referred to as a "server".

Client — A computer that requests data from a host. A client may be a personal computer running an application such as MLS access software or a CMA program, or it may be another server such as a Web site, a mortgage or title system or another MLS host.

DTD — Document Type Definition. This is used in connection with a particular type of XML document (such as one relating to a real property listing) to describe the structure of the data, for example, the names of the fields and the order in which they should appear on the display.

Meta-Data — Data in a defined format whose sole purpose is to describe other data. As a simplified example, the meta-data could describe a "school district" field as being searchable, not more than 32 alphabetic characters in length and governed by a table of valid entries.

Tag — When used in the context of an XML document, this refers to the name of a particular data item. Tags in XML are enclosed in '<' and '>'. Examples include <PROPERTY> and </PROPERTY>

TCP/IP — The communications protocol that is used on the internet to ensure that data sent arrives at the destination error free and in the same order in which it was sent.

XML — eXtensible Markup Language. An industry standard, hierarchical way of representing and adding meaning to data.

Solution Criteria

The task force realized that the issue of standards in real estate was a large one. The group realized that while the vision was large, the initial scope needed to be *limited* without being *limiting*. To this end, the group established the following set of criteria against which all proposed solutions would be measured.

- Must be **based on Internet (TCP/IP) standards**
- Must **support XML**
- Must be **goal oriented** – able to solve key business problems faced today
- Must be **extensible** – able to change to handle tomorrow's business problems
- Must be **implementable at reasonable cost**
- Must be **operate in conjunction with most current systems**
- Must **support established business rules** – able to assure the security of MLS data
- Must **create an environment conducive to increased competition** among software vendors
- Must **allow for query and transfer of more than property data**, such as agent, office and tax data

As the group considered the above solutions criteria, it realized that the issue of data interchange could be categorized into the following areas.

- Data Acquisition
- Data Format
- Data Meaning/Interpretation

Clearly, these problems must be tackled in order. There is obviously no need to understand the format of data that has not yet been acquired. Likewise, there is no need to interpret data unless that same data can be parsed. Based on this, it was decided to break the problem down into the following areas and attack each in order.

- A transaction set for client-host communication (Acquisition)
- A delivery format for search results (Data Format)
- A meta-data, or data dictionary, structure (Data Interpretation)
- A limited or minimum subset of common fields (Data Interpretation)

At first glance, it may seem that most, if not all, of these issues must be defined and standardized before any real value is gained. However, with further thought, it should be clear that solving or standardizing each in order incrementally adds immense value. For instance, it should be clear that the establishment of a standard transaction set which can be used by Real estate software solves some huge problems. REALTORS® should no longer have to deal with the problem of their favorite client software not being able to communicate with and acquire data from their MLS system. Client vendors will no longer have to spend development time custom coding interfaces for each host, and can instead concentrate on enhancing the features of the product, which will benefit the REALTOR®. Host vendors will no longer have to support multiple interfaces for each

particular client that comes on the market, and can also spend that time on adding features. Competition will be increased, as vendors will have to compete on the merits of their products and services, rather that falling back on being one of a few choices in a particular market due to the existence of specialized interfaces. A standardized transaction set is a win-win situation for REALTORS® and vendors alike.

Once clients and hosts have a common way to communicate, the next most valuable item is a definition of the format of the data to be transferred. The format of the data can take a number of different forms, some of which provide meaning to the data and others that do not. As the reader will see, the task force chose two different forms for data transfer, a compact format and an XML based format. The compact format will be most useful to clients that need fast, efficient transfer of large quantities of data, and that need to access every data field the host has available. This format is also necessary for clients that need the data to be represented in the exact way that it is stored on the host. Client software put out by MLS vendors for use with a particular host might be likely to use this format. The XML format will be most useful to clients that prefer to deal with the standardized subset of fields described by the chosen XML DTD. These clients will not generally require every piece of data available on any particular host and will be able to deal with some potential variation of that data for the benefit of being more generic. Sample applications that might benefit from XML data transfer would include CMA and flyer generation programs, web site population and data transfer for the purpose of printed advertisements. It is expected that many third-party (non-MLS vendor) created tools will use the XML data transfer format.

When clients have a way to communicate with hosts, make queries and transfer the results of those queries, they may then need a way to interpret that data. The standards task force has addressed this need in three complementary ways.

- The definition of a subset of common fields with common names or 'tags'
- The definition of a standardized format for meta-data, or a 'data dictionary', which can describe any or all fields on a particular MLS system
- The creation of an XML DTD which can be used in conjunction with the XML data transfer format to add meaning to the data

Each of these will be briefly described in the rest of this summary document. The reader is referred to the corresponding technical specification for more specific information.

Transaction Set

Today, developers of both host and client software for the real estate industry are faced with the daunting task of custom coding an interface for each combination of client and host software that wish to communicate. Client vendors will often support interfaces for each host that they wish to talk to, and host vendors are faced with implementing specific interfaces for clients that are requested by the MLS boards. In most cases, the operations that are executed between the client and the host are similar. The client must first log in

to the host and authenticate itself, usually by means of a user name/password combination, an answerback code, or some combination of the two. Then, the client issues a search or query, requesting some set of data from the host. The host then responds with the requested data. The client might optionally go back for supplemental data, such as pictures. Other queries for more data might follow, based on the particular client application. When done, the client informs the host that the host's services are no longer required in this session by issuing a log off.

In many cases, for numerous reasons, clients have chosen to implement this communication with the host via a method called 'screen scraping'. When screen scraping, the client reads the prompts coming from the host and responds like a user would who was looking at a screen and responding via the keyboard. While screen scraping can be an easy method of getting communication going between a client and a host, it is highly prone to errors, either from transmission errors or from subtle changes in the host user interface.

In other cases, a host vendor may provide an interface, or the client may have a predefined interface which a host vendor needs to implement, but these interfaces are not standard and often do not allow ready adaptability to other host or client systems. This lack of standardization leads to a lot of non-value-added work on the part of software vendors, and can prevent software from interacting as well as it should, to the distinct disadvantage of the end user.

Clearly, there is a need for standardization in this area. It should also be clear that, regardless of what data is transmitted, or in what format the data is transmitted, a standard method of communicating with all hosts and issuing requests as described above is paramount. If such a standard were defined and implemented, host and client vendors could be ensured of the ability to code the communications part of an interface once, and concentrate on the use of and interpretation of the data that is returned.

The task force decided upon a set of HTTP-based transactions, or operations, that would allow clients to perform the necessary steps needed to acquire data from a host. An HTTP based system was selected for a number of reasons:

- HTTP is TCP/IP based, which provides a reliable, error-free communications link that is widely and inexpensively available to users
- HTTP is a well understood, well documented internet standard
- HTTP can pass through most firewalls without additional setup
- HTTP is well supported by many third party tools, making initial development easier and less prone to errors
- An HTTP based transaction set lends itself to easy testing and experimentation in a browser environment, which will lead to quicker understanding and hence better acceptance in the vendor community

For those readers who do not wish to read the specification in its entirety, the following sections gives a listing and brief overview of each of the transactions that have been defined in Version 1.0 of the specification.

LOGIN
The LOGIN transaction is the first transaction that a client must issue during a session with a host. It identifies to the host the username and password of the human user, giving the host the opportunity to reject the login request or apply business rules as needed. The LOGIN transaction uses a simple challenge-response method called Digest Authentication in order to prevent sending the user's password over the Internet in clear text form, providing a minimal level of security. The LOGIN transaction may also initiate processing by the host to help prevent piracy of client software.

In the response, the host provides the client software with a number of valuable pieces of information, the most important of which is a list of URLs which give the client access to the various transactions and other capabilities of the host. Besides the standard (required) transactions, the host can optionally provide additional URL links for additional capabilities, which help to ensure that the standard is not only extensible in future standards, but is also extensible by private agreement. Also provided in the response is information such as the current meta-data version and account and user information.

LOGINCOMPLETE
The LOGINCOMPLETE transaction is the second part of the LOGIN transaction that enables software copy protection security/piracy prevention. It allows the client and host to synchronize the information kept to prevent piracy. Technically, it initiates the second phase of a two-phase commit algorithm. In layman's terms, it confirms for the host that the client has completed its check of security, and requests the host to update its software security data for that particular client software package.

GETOBJECT
The GETOBJECT transaction is used to retrieve additional information that hosts store related to specific data objects. This transaction is used primarily to retrieve multimedia types associated with host records (house pictures, agent photos, sound clips, 3D tours, etc). It also allows the client to download the latest copy of the meta-data, thereby enabling the client to configure itself to the host and adapt to changes.

LOGOUT
The LOGOUT transaction terminates a particular client/host session. In effect, the LOGOUT transaction tells the host that the client has completed all processing that is going to be done in this session, and allows the host to free up any information it had been keeping on behalf of the client. It also ensures that the host logs the user out of the system, and for systems that prevent simultaneous login of the same username, ensures that the username can log in again in the future.

SEARCH
The SEARCH transaction is the primary workhorse transaction of RETS. It enables a client to request a set of data from the host, and gives the host a number of additional parameters with which to work. These parameters include things like the type of data to be returned, the query itself, the data fields to return and the format for the data returned. The standard query language is also defined as part of the search transaction. For instance, a client can request **"All active properties in area 512 with a list price of between $200,000 and $250,000 with 3 or more bathrooms, with the return data set of no more than 50 records to be formatted in COMPACT-DECODED format"**. The host executes the query and returns the data or an error code to the client for further processing. The SEARCH transaction is very detailed, and the reader is referred to the standards document for further information on this transaction.

GET [File]
The GET [File] transaction simply allows the client to get an arbitrary file from a specific URL on the host. It is not specified how the client can obtain the name of the file, nor is it specified what business rules might need to be enforced. GET [File] allows additional host-specific functionality to be built into specific clients. This transaction might be used to acquire additional host information or to download client software updates.

For further details on the above transaction set, the reader is referred to the RETS 1.0 specifications document.

Data Transfer Format

The transaction set by itself can significantly increase the interoperability between client and host software used in the real estate industry. However, the main purpose for the transaction set is the acquisition of data from the host. Although each host could define this format independently, it should be clear that a standardized format for the data being sent to the client from the host would greatly simplify the interaction.

The task force partitioned data acquisition clients into two distinct groups: those that tend to need most if not all of the fields on the host, and those that generally work with a common, more generic subset of fields on the host. In order to meet both of these requirements, the task force designed two different data transmission formats, each having traits that make it more suitable for one of the above groups of client software. It should be noted that the standard does not specifically designate use of one format for one particular client type – it simply drew upon it's collective expertise in the real estate software industry and attempted to accommodate both groups.

The COMPACT Format
The COMPACT format is a simple and efficient way for a client to receive data from the host. For each query, the host returns a set of ASCII text lines, each delimited by a

CR/LF (0x0D/0x0A) combination. The first such line is a tab[1] (0x09) delimited set of field names. This line must be provided by the host and specifies to the client the order of the fields for each subsequent line. Following this field name line, each data record is returned. As with the field name line, an ASCII TAB character separates the data fields for each record. This format is commonly known as "tab delimited ASCII data with a header record", is familiar to many developers, and is very easy to parse. It is also fairly compact, in that the delimiters in the data are a very small proportion of the overall data stream.

The following is a very simple example of a set of properties being returned from a query. The delimiting TAB character is designated by the '➔' character. Please note that the choice of fields, the field names and the data have been arbitrarily selected for the sake of this example – they in no way should be interpreted as being required or otherwise condoned by the standard or the task force.

EXAMPLE 1:

```
MLNUM➔STATUS➔AREA➔ADDR➔LISTPR➔BEDS➔BATH
99040102➔ACT➔101➔123 MAIN➔105000➔3➔2
99030204➔ACT➔101➔100 WASHINGTON SQ➔550000➔5➔4
99030410➔SLD➔345➔26200 SW 95TH➔345000➔4➔2.5
```

The COMPACT form has an optional 'DECODED' mode that can be requested by the client. If this mode is enabled, the host must translate any 'coded' fields into human readable text. This option is useful for clients that do not store coded values, and only display text for the user. Please note the differences between the next two examples.

EXAMPLE 2: (DECODED = FALSE)

```
MLNUM➔STATUS➔AREA➔WATER
99040102➔ACT➔101➔A
99030204➔ACT➔101➔BD
99030410➔SLD➔345➔BDF
```

EXAMPLE 3: (DECODED = TRUE)

```
MLNUM➔STATUS➔AREA➔EXTERIOR
99040102➔ACT➔101➔WELL
99030204➔ACT➔101➔CITY WATER,CITY SEWER
99030410➔SLD➔345➔CITY WATER,CITY SEWER,SOFTENER
```

The XML Format

XML (eXtensible Markup Language) is one of the most exciting technologies for data interchange to come along. XML is a hierarchical, tagged markup format similar to

[1] The standard does not specifically designate that an ASCII TAB (0x09) character be the delimiter for fields. The standard specifies that the host must provide the client with a single byte value that is used as a field delimiter. For the sake of simplicity, this document assumes that this delimiter is an ASCII TAB.

HTML[2] and is excellent at providing meaning to data that is transferred between disparate systems. The key to XML's ability to attach meaning to data items is the corresponding XML DTD. The DTD gives both meaning and order to data. If two different computer systems utilize the same DTD, regardless of how the data is actually stored on each system, they can exchange data. The following is a very simple example of how an XML data transfer of two property records might look[3].

EXAMPLE 4:

```
<PROPERTY>
      <MLNUM>99040102</MLNUM>
      <STATUS>ACT</STATUS>
      <AREA>101</AREA>
      <ADDR>123 Main</ADDR>
      <LISTPR>105000</LISTPR>
</PROPERTY>
<PROPERTY>
      <MLNUM>99030204</MLNUM>
      <STATUS>ACT</STATUS>
      <AREA>101</AREA>
      <ADDR>100 WASHINGTON SQ.</ADDR>
      <LISTPR>550000</LISTPR>
</PROPERTY>
```

XML data transfer is more verbose than the COMPACT format, because of the starting and ending tags around each data item. However, this disadvantage may be overcome by the advantage the client gains in receiving the *description* of the data along with the data itself. Using Example 4, it should be easy to see that a piece of software that downloads and formats web pages for properties from a wide variety of host systems might benefit greatly from the ability of all hosts to transmit data in this format.

The XML format does not have a corresponding 'DECODED' flag. Data transferred in the XML format must always be decoded, because given a particular set of data formatted in XML, the client would have no way to determine if the data for a field was a coded or an actual representation. Were this rule not in place in the following example, the client would have no way of knowing if the '1' in the **GARAGE** field represents a one-car garage, or is instead some coded value that might represent 'no garage'[4].

EXAMPLE 5:

```
<PROPERTY>
      <MLNUM>99040102</MLNUM>
      <GARAGE>1</GARAGE>          Without a rule, it is unclear what '1' represents
</PROPERTY>
```

[2] In actuality, HTML and XML are distinct but compatible implementations of SGML. The specifics and details of this similarity is beyond the scope of this document.

[3] Please note that this example is only a very simple example and should not be viewed as an example of the specification itself.

[4] On some systems, the garage field might be stored as a numeric value, representing the number of car spaces. On other systems, the garage field might be stored as a coded field, with 1=no garage, 2=carport, 3=one car garage, 4=2 car garage, etc. It is because of this difference in storage and interpretation that the 'always decode in XML' rule is so important.

When using the XML data transfer format, hosts and clients must adhere to the standard DTD that has been defined by the task force. Please note that the task force has specifically *not* attempted to provide a tag or meaning to every possible data field that host systems store. The task force deemed this problem to be extremely complex, of little actual business value, and was certain that any attempt to solve this problem along with the release of the first version of the standard would almost certainly delay the release and implementation of the standard. Therefore, clients choosing to transfer data in XML format will have to restrict themselves to a common subset of fields that has been standardized in the first version of the XML DTD for real estate data transfer. Again, for many applications that choose to use XML, this restriction may not be significant. If a client needs access to data fields that are not defined in the DTD, that client will have to revert to use of the COMPACT data transfer format.

Meta-Data/Data Dictionary Structure

Meta-data is simply data that describes other data. It is often referred to as the 'data dictionary'. During the standards process, the task force realized that simply transferring data from host to client, while providing value, was not sufficient. It was also very useful to provide some way for a host to describe to a client *what* kind of data resided on the host (property, tax, history, etc) and for each type of data, a description of each field in that data. Given this information, an intelligent client could reasonably adapt itself to the capabilities of the host, and display or interpret data in a way that users of that host expect. Storing meta-data on each host, and making it accessible to the client via the defined transaction set has a number of benefits.

- It will reduce or eliminate the need for client vendors to do host specific setup
- It will reduce or eliminate the need for vendors to distribute, and users to acquire and install, host specific software distributions
- It will reduce or eliminate data errors and/or the need for software updates when host systems change

In defining the meta-data content, the task force provided not only the most basic and necessary information about the host data fields (name, printable name, length, data type), but also a number of descriptions that provide the client with 'hints' on how the data should be displayed for the user. The client need not use these hints, but they can go a long way toward making the client product more consistent with the intent of the host system designers with regard to data layout. The end user will see data formatted and displayed consistently in all of the applications that use the same host data.

The following table was taken from the specification and simplified for inclusion in this overview. The reader is referred to the specifications document for additional information.

Table 1 – Meta-data Content

Metadata Field	Description
SystemName	The name of the field as known to the native host
StandardName	The name of the field as known in the Real estate DTD
LongName	The name of the field as known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined.
ShortName	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined.
MaximumLength	The maximum length of the field, in characters or digits as appropriate.
DataType	The kind of data that this field holds. Examples include character, date, time and number
Precision	The number of digits to the right of the decimal point when formatted.
Signed	A true/false value indicating whether the field is signed.
Interpretation	An indication to the client as to how this numeric value should be interpreted. This includes as a number, as currency and as a lookup value.
Alignment	A hint to the client as to how the value of the corresponding field should be aligned (left, center, right, justified) when displayed.
UseSeparator	A true/false value which indicates that the numeric value SHOULD be displayed with a thousands separator.
Picture (optional)	A string which can be used by the client to perform simple insertion editing on a text or numeric field.
LookupData	The name of the LookupData section containing the lookup data for this field. Required if Interpretation is Lookup.

Host systems are under no obligation to provide meta-data for each and every field on their system. However, the usefulness of meta-data that does not completely and accurately describe the system is limited.

Common Field Definitions

The data stored regarding properties in various areas of the country is quite diverse, as is the terminology used to describe features of those properties. Whereas in Colorado, data storage regarding access to ski areas may be critical, similar data in Florida is probably not so important. A home might also be correctly referred to as a ‘ranch’, a ‘rambler’ or a ‘one-story’ in different parts of the country. The task force surmised that any attempt to commonly define each and every piece of data that is or might need to be stored regarding a property was an immensely difficult and time-consuming process. The task force also postulated that the majority of client applications did not actually need ‘meaning’ attached to most data items. The task force recognized that access to,

descriptions of and display hints for each and every data item stored had great value, but that for many client applications, a common meaning or interpretation need only be attached to a *limited subset* of the data items.

For instance, CMA applications would, at a minimum, need to know which data fields represented the MLS Number, the Status, the Address, the Price, and the number of Bedrooms and Bathrooms[5]. There are many other data items that readers of the CMA might wish to see, but for most of those additional fields, the client software need only know that the field exists, the name of it and how to display it.

Another example might be a large regional broker who wishes to acquire all of their own listings from the MLS systems in the areas they serve and publish them on a web page. Even within this smaller radius, the kind of data stored will vary greatly from system to system. Without definition of some common data fields and some way to attach meaning to those fields, the broker is faced with the task of creating and maintaining a 'mapping' of fields on each system to the fields in the local data store. With a set of common data fields, this problem is virtually eliminated, and the broker can easily consolidate some of the most important fields from multiple MLS systems.

This solution to this latter issue is very similar to that addressed with the XML data transfer format described above. Indeed, the two solutions intersect. The fields described by the XML DTD are simply the defined common data fields used as XML tags.

The task force recognizes that the set of fields chosen will not initially address each and every problem faced by the real estate community. However, it is expected that the definition, acceptance and implementation of the initial set of fields will solve many of those problems, and will be an imminently better situation than that which exists today. The task force expects that the number of fields in this 'common' set will grow slowly over time. The defined standards change control process will manage additions to this set. It is not expected that this list will grow to encompass every possible data field for the reasons explained above.

[5]The author requests that the reader grant him some leeway for the simplicity of this example. This example only serves to illustrate the concept, and does not attempt to exactly mimic the issues that CMA programs face.