

RETS Change Proposal 59: Changes for Metadata Driven Update Clients

Author: Sergio Del Rio

Organization: Templates 4 Business, Inc.

Telephone Number: (604) 529-1544

Email: Sergio.Del.Rio@t4bi.com

Status: Proposal

Date: March 18, 2005

Proposal Revision: 1.0

1. Synopsis

This change request consists of two main sections. The first section consists of a set of changes that are designed to enhance the functionality of the RETS 1.7 specification. These enhancements are geared towards enabling a fully metadata driven RETS Update client. The second section consists of a set of additional metadata tables that are designed to provide presentation guidelines to clients. These enhancements are geared towards enabling a fully metadata driven RETS Update or Query client.

2. Rationale

We have been striving to create RETS Update Client that is 100% metadata driven, yet still presents itself to users as a fully functional update client that presents data to them in a way that they are comfortable.

When looking at what is available in the specification as it stands today, we realized that this goal was not fully attainable unless some changes were made to the specification. With that in mind, we set out to build it.

With all the changes that are presented below, we were able to build a RETS Update Client that is as functional as we believe is necessary to provide users with a good user experience and allow them to enter their listing data in the most effective way possible. This client is now in alpha testing and has almost identical to its predecessor which was interacting directly with the MLS database.

3. Proposal

This proposal affects several sections of the RETS 1.7d5 document as follows.

3.1 Section 7 - Search Transaction Changes

When building the update client, we wanted to be able to guarantee to users that nobody else would change their data using another client. In order to make this possible, we wanted to provide servers that wished to implement this feature the ability to do so. As such, we propose the addition of the following Request Argument:

ForUpdate ::= 1 | 0

When set to 1, a client is telling the server that it is doing the query in order to update the data. If a server chooses to do so, it may lock this record. If a server chooses to lock the record, it is expected that if a second user attempts to select the record for update, a new response code would be provided as follows:

20301 – Record requested for update is already locked by another user.

Furthermore, when set to 1, a server reserves the right to limit the number of records that can be locked to 1 or, at the very minimum, to limit the number of records locked to a number smaller than the current select Limit.

When set to 0, normal search functionality is assumed.

3.2 Section 10 - Update Transaction Changes

It was discovered that there was an error in the definition of warning-response in the Required Request Arguments section of the specification.

Since it defined only the error-num as being passed up by the client for any responses, this would have required a server to actually save the error messages that were transmitted to the client since it is quite rare that a server would keep unique error messages across individual fields if the same error occurs on multiple fields.

For this reason, the following modification to the warning-response definition are proposed:

warning-response ::= field:warning-num = user-response

field ::= 1*32ALPHANUM

warning-num ::= 1*16DIGIT

user-response ::= *1024TEXT excluding delimiter

Additionally, it was realized that the error-num, warning-num and error-text fields are much to small for modern computer systems.

For this reason, the following modifications are being proposed to these fields in the Update Response Body format section:

error-num ::= 1*16DIGIT

error-text ::= *1024TEXT

warning-num ::= 1*16DIGIT

warning-text ::= *1024TEXT

We have also noted that there is a tag identified in the User Response Body Format called transaction-id-tag which is not defined anywhere in the documentation. We propose implementing this as follows:

transaction-id ::= <TRANSACTIONID value = “transaction-id-value”/>

transaction-id-value ::= 1*32ALPHANUM

3.3 Section 11.3.2 – Additional field to Table Metadata

In order for our client to provide the user with immediate feedback on data entry, we required a way to communicate the minimum acceptable data length for a field. In order to do this, we required the following additional field in Table 11-9 Metadata Content - Tables.

Table 11-9.1 Metadata Table – Extensions – Class

Metadata Field	Content Type	Description
MinimumLength	Numeric	The minimum length that this field is allowed to have. This is used in both Search and Update transactions.

3.4 Section 11.3.3 – Additional well known UpdateName

In order to allow clients to tell the server that they no longer want to have a record locked that was previously selected using the new ForUpdate Request Argument, we propose that the following UpdateName be added to the list of well known :Update Names in the Table 11-11 Metadata Content – Update as follows:

Metadata Field	Content Type	Description				
UpdateName	1*24ALPHANUM	Additional Values Required: <table border="1"> <thead> <tr> <th>Update Name</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>Release</td> <td>Releases the record that was previously checked out using the Search transaction with the ForUpdate=1 request argument.</td> </tr> </tbody> </table>	Update Name	Function	Release	Releases the record that was previously checked out using the Search transaction with the ForUpdate=1 request argument.
Update Name	Function					
Release	Releases the record that was previously checked out using the Search transaction with the ForUpdate=1 request argument.					

3.5 Section 11.3.4 – Additional fields to Metadata Update Type

As we designed and built the client, we realized that we needed several additional features in this table as follows:

Table 11-13.1 Metadata Content – Extensions – Update Type

Metadata Field	Content Type	Description									
Attributes	See Table 11-13 in RETS document.	Additional Values Required: <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>6</td> <td>AutopopRequired</td> <td>Indicates that this field is mandatory before calling Update transaction for Auto Population.</td> </tr> <tr> <td>7</td> <td>Hidden</td> <td>Indicates that this field may be used in ValidationExpressions but is to remain hidden from the user.</td> </tr> </tbody> </table>	Value	Meaning	Description	6	AutopopRequired	Indicates that this field is mandatory before calling Update transaction for Auto Population.	7	Hidden	Indicates that this field may be used in ValidationExpressions but is to remain hidden from the user.
Value	Meaning	Description									
6	AutopopRequired	Indicates that this field is mandatory before calling Update transaction for Auto Population.									
7	Hidden	Indicates that this field may be used in ValidationExpressions but is to remain hidden from the user.									
DisplayOrder	1*32ALPHANUM	The order that fields should appear in a default one-line search result that is executed in order to give the user a list of existing listings to select from. Fields that should not appear in the default one-line format should have a value of 0. Fields that should never be visible to the user should have a value of -1.									
OverrideOnInsert	Boolean	A truth value which indicates whether an Autopop, ValidationExternal or SET ValidationExpression may be overridden by a user for new data records.									
OverrideOnUpdate	Boolean	A truth value which indicates whether an Autopop, ValidationExternal or SET ValidationExpression may be overridden by a user when updating data records.									

3.6 Section 11.4.8 – Support additional Parent in Validation Lookup Type request

In order to provide similar functionality to the new proposed Lookup Type By Parent metadata (see section 3.11 below). We propose that the request to get Validation Lookup Type be allowed to specify an optional parameter that indicates the client only wants data for a specific parent.

GetMetadata request:

Type: METADATA-LOOKUP_TYPE_BY_PARENT
 ID Format: Resource : ParentLookup : ParentLookup Value
 ID Example: Property : STATES_OR_PROVINCES : MD

3.7 Section 11.4.9 – Additional fields to Metadata Validation Expression

As we tried to implement a full rule engine that handled rules specified as validation expression, we realized that for better process control and for a better user experience, we needed to extend this table as follows:

Table 11-36.1 Metadata Content – Extensions – Validation Expression

Metadata Field	Content Type	Description
ConditionalValue	1*512TEXT	A test expression that is to be executed before attempting to execute the Value expression. If this expression evaluates to TRUE, the Value expression is to be executed. If it evaluates to false, the Value expression is NOT to be executed.
Message	1*512TEXT	A message to be displayed in the following cases: 1. Expression Type of ACCEPT results in FALSE. 2. Expression Type of REJECT results in TRUE. 3. Expression Type of WARNING results in TRUE.

3.8 Section 11.4.9 – Additional Keywords to Validation Expression Types

In order to enable the functionality that is required for a fully functional and user-friendly update client, we propose that the following additions be made:

Table 11-32.1 Validation Rule Execution Types

Type	Description
SET_DISPLAY	Rules of this type are designed to allow a client to make fields visible or invisible based on the evaluation of an expression.
REJECT	Rules of this type are designed to evaluate an expression and reject the record if the expression returns true and accept the record if the expression returns false. The return data type of these rules MUST be Boolean.

WARNING	Rules of this type are designed to evaluate an expression and display a warning message to the user if the expression returns true. The return data type of these rules MUST be Boolean. Rules of this type may require that the user acknowledges the WARNING in some manner.
SET_REQUIRED	Rules of this type are designed to evaluate an expression and set the field the rule is being executed on to Required if the expression returns true and to Non Required if the expression returns false.
SET_READ_ONLY	Rules of this type are designed to evaluate an expression and set the field the rule is being executed on to Read Only if the expression returns true and to Updateable if the expression returns false.
RESTRICT_PICKLIST	Rules of this type are designed to return one or more LOOKUP values that are to be removed from the LOOKUP list that is defined for the field the rule is being executed on. This is always the entire set of values to remove from the lookup. In other words, if this returns a blank list or .EMPTY., the entire set of LOOKUP values is to be displayed.

3.9 Section 11.4.9 – Validation Expression Grammar

Upon examination of the Validation Expression Operators, it became evident that we needed to define a proper grammar that would handle the Validation Expression Operators in a proper manner. This was necessary to allow expressions to be properly parsed and evaluated.

Following is a BNF description of the grammar that we implemented for this purpose and that we propose is added to the specification:

Terminals

This section describes the Tokens or Terminals used in the Grammar. They include all the tokens specified in Section 11.4.9 of the **Real Estate Data Interchange Standard: Real Estate Transaction Specification Version 1.7d5** and two minor additions, the <STRING_CONCAT> operator “||” and the **.RECORDSTATUS.** Special Operand Token.

The <STRING_CONCAT> operator “||” allows for concatenation of strings within an expression. It is identical in function to the ANSI SQL concatenation operator "||" used by Oracle, DB2 and others. See the **ValueExpr** non-terminal below for usage.

The Special Operand Token **.RECORDSTATUS.** indicates the status of the current listing. The value may be designated as ‘NEW’, for a new listing, ‘UPD’ for an updated listing and ‘DEL’ for a deleted listing.

e.g. validation expressions:

```
(ListingStatus = 'Active' .AND. .RECORDSTATUS. = 'NEW')
```

```
(.RECORDSTATUS. != 'UPD')
```

For further descriptions of the tokens please refer to tables 11-33 and 11-34.

TOKEN : /* See Validation Expression Operators - Table 11-33 */

```
{
  < DIVISION: "/" >
  | < MULTIPLICATION: "*" >
  | < MODULO: ".MOD." >
  | < PLUS: "+" >
  | < MINUS: "-" >
  | < CONTAINS: ".CONTAINS." >
  | < STRING_CONCAT: "||" >
  | < LESS: "<" >
  | < LESSEQUAL: "<=" >
  | < GREATER: ">" >
  | < GREATEREQUAL: ">=" >
  | < EQUAL: "=" >
  | < NOTEQUAL: "!=" >
  | < AND: ".AND." >
  | < OR: ".OR." >
  | < NOT: ".NOT." >
}
```

TOKEN: /* Separators and operators */

```
{
  < OPENPAREN: "(" >
  | < CLOSEPAREN: ")" >
  | < SEMICOLON: ";" >
  | < COLON: ":" >
}
```

TOKEN : /* Validation Expression Special Operand Tokens - Table 11-34 */

```
{
  < TODAY: ".TODAY." >
  | < NOW: ".NOW." >
  | < ENTRY: ".ENTRY." >
  | < EMPTY: ".EMPTY." >
  | < OLDVALUE: ".OLDVALUE." >
  | < USERID: ".USERID." >
  | < USERCLASS: ".USERCLASS." >
  | < USERLEVEL: ".USERLEVEL." >
  | < AGENTCODE: ".AGENTCODE." >
  | < BROKERCODE: ".BROKERCODE." >
  | < BROKERBRANCH: ".BROKERBRANCH." >
  | < RECORDSTATUS: ".RECORDSTATUS." >
}
```

TOKEN: /* Literals */

```
{
  < INTEGER_LITERAL: ([ "0"-"9" ])+ >
  | < FLOATING_POINT_LITERAL:
    ([ "0"-"9" ])+ "." ([ "0"-"9" ])+ (<EXPONENT>)?
    | "." ([ "0"-"9" ])+ (<EXPONENT>)?
    | ([ "0"-"9" ])+ <EXPONENT>
    | ([ "0"-"9" ])+ (<EXPONENT>)?
  >
  | < #EXPONENT: [ "e", "E" ] ([ "+" | "-" ])? ([ "0"-"9" ])+ >
  | < STRING_LITERAL: "" (~[ "" ])* ( "" (~[ "" ])* ) * "" >
}
```

TOKEN: /* Identifiers */

```
{  
  < ID: (<LETTER>)+ ( "-" | "$" | "#" | <DIGIT> | <LETTER>)* >  
  | < #LETTER: ["A"- "Z", "a"- "z"] >  
  | < #DIGIT: ["0"- "9"] >  
}
```

TOKEN : /* Period - Date, DateTime, Time in ISO8601 format */

```
{  
  < DATE: (<DIGIT> <DIGIT> <DIGIT> <DIGIT> <MINUS> <DIGIT> <DIGIT> <MINUS> <DIGIT> <DIGIT>)  
>  
  | < DATETIME: (<DIGIT> <DIGIT> <DIGIT> <DIGIT> <MINUS> <DIGIT> <DIGIT> <MINUS> <DIGIT>  
<DIGIT> "T" <TIME>) >  
  | < TIME: (<DIGIT> <DIGIT> <COLON> <DIGIT> <DIGIT> <COLON> <DIGIT> <DIGIT>) >  
}
```

Non-Terminals

The BNF Grammar is specified as follows. Note that order of precedence of the operators is implied in the grammar and identical to that specified in Table 11.33 with the exception of the .NOT. operator.

Input	::= Expression (<EOF> <SEMICOLON>)
Expression	::= OrExpr
OrExpr	::= AndExpr (<OR> AndExpr)*
AndExpr	::= NotExpr (<AND> NotExpr)*
NotExpr	::= ((<NOT>))? EqualExpr
EqualExpr	::= CompareExpr (CompareExprRight)?
CompareExpr	::= (SumExpr (SumCompareExprRight)?) Empty
Empty	::= <EMPTY>
CompareExprRight	::= EqOp CompareExpr
SumCompareExprRight	::= CompareOp SumExpr ((<CONTAINS>) ValueExpr)
SumExpr	::= (ProductExpr (((<PLUS> <MINUS>)) ProductExpr)*)
ProductExpr	::= UnaryExpr ((((<MULTIPLICATION> <DIVISION> <MODULO>)))) UnaryExpr)*
UnaryExpr	::= (((<PLUS> <MINUS>)))? Term
Term	::= ((<OPENPAREN>) OrExpr (<CLOSEPAREN>)) (ValueExpr)
ValueExpr	::= Value ((<STRING_CONCAT>) Value)*
Value	::= Id StringLiteral SpecialOperandToken WellKnownNameSpecialOperandToken PeriodLiteral PeriodSpecialOperandToken NumberLiteral
SpecialOperandToken	::= (<ENTRY> <OLDVALUE>)
PeriodSpecialOperandToken	::= (<TODAY> <NOW>)
WellKnownNameSpecialOperandToken	::= (<USERID> <USERCLASS> <USERLEVEL> <AGENTCODE> <BROKERCODE> <BROKERBRANCH> <RECORDSTATUS>)
Id	::= <ID>
StringLiteral	::= <STRING_LITERAL>
PeriodLiteral	::= (<DATE> <DATETIME>)
NumberLiteral	::= (<INTEGER_LITERAL> <FLOATING_POINT_LITERAL>)
EqOp	::= (<EQUAL> <NOTEQUAL>)
CompareOp	::= (<GREATER> <GREATEREQUAL> <LESS> <LESSEQUAL>)

3.10 Section 11.4.11 – Additional fields to Metadata Validation External

In order to allow for additional restrictions of data that should be selected when using a Validation External, we propose the additional field be added:

Table 11-40.1 Metadata Content – Extensions – Validation Expression

Metadata Field	Content Type	Description
RestrictFields	1*1024PLAINTEXT	<p>A comma separated list of valid field pairs joined by = (equal) the first is a target field in the table being updated and the second is a source field in the table being searched. The fields use a SystemName from Section 11.3.2.</p> <p>If values have been entered in the table being updated for any of the above fields, the query executed on the source table must be restricted to the values entered.</p>

3.11 Additional Presentation Metadata

The final proposed change is to add all of the following new metadata to the specification in order to allow for fully dynamic clients to present data in a way that the MLS providing the RETS Server would like and that it's Realtors expect to see.

METADATA-LOOKUP_PARENT

This metadata defines a list of METADATA-LOOKUP objects that have parent lookup objects. Servers have the option of outputting a single record with a value of -1 for any METADATA-LOOKUP-TYPE requests for any METADATA-LOOKUP objects specified within this list. This is an indicator that the full lookup list is extremely long and that clients should use the METADATA-LOOKUP_TYPE_BY_PARENT metadata to get this information instead.

The Lookup Parent metadata starts with a <METADATA-LOOKUP_PARENT> tag with Resource, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-41, followed by the <DATA> section, which contains the actual field information. The Lookup Parent metadata has the following format:

```
<METADATA-LOOKUP_PARENT SP Resource="resource-id" SP
Version="lookup-parent-version" SP Date="lookup-parent-date"> ↵
<COLUMNS>→lookup-parent-field *(→lookup-parent-field)→</COLUMNS> ↵
*(<DATA>→lookup-parent-data *(→lookup-parent-data)→</DATA> ↵)
</METADATA-LOOKUP_PARENT> ↵
```

*resource-id ::= 1*32ALPHANUM*

This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Lookup Parent objects belong.

*lookup-parent-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS*

This is the version number of this Lookup Parent metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. Every time this Lookup Parent metadata changes the version number should be increased.

lookup-parent-date ::= DATE

The latest change date of this Lookup Parent metadata.

lookup-parent-field ::= <Field Name from Table 11-41>

lookup-parent-data ::= <valid value as defined in Table 11-41>

An example Table section follows:

GetMetadata request:

Type: METADATA-LOOKUP_PARENT
ID Format: Resource
ID Example: Property

Compact reply:

```
<METADATA-LOOKUP_PARENT Resource="Property" Version="1.2.100" Date="Wed,
12 Jan 2005 17:38:19 GMT">
<COLUMNS>→ParentName→ChildName→Version→Date→</COLUMNS>
<DATA>→COUNTIES_OR_REGIONS→CITIES→1.2.70→Wed, 12 Jan 2005 17:38:19
GMT→</DATA>
<DATA>→STATES_OR_PROVINCES→COUNTIES_OR_REGIONS→1.2.70→Wed, 12
Jan 2005 17:38:19 GMT→</DATA>
<DATA>→COUNTRIES→STATES_OR_PROVINCES→1.2.70→Wed, 12 Jan 2005
17:38:19 GMT→</DATA>
<DATA>→AGENCY_TYPES→SUB_AGENCY_TYPES→1.2.70→Wed, 12 Jan 2005
17:38:19 GMT→</DATA>
</METADATA-LOOKUP_PARENT>
```

Table 11-41 Metadata Content – Lookup Parent

Metadata Field	Content Type	Description
ParentName	1*32ALPHANUM	The lookup-id for the METADATA-LOOKUP that contains all the valid Parent values for the METADATA-LOOKUP identified by ChildName.

ChildName	1*32ALPHANUM	The lookup-id for the METADATA-LOOKUP that has parent values.
Version	1*2DIGIT . 1*2DIGIT . 1*5DIGIT	The latest version of the Lookup Type By Parent metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. The version number is advisory only.
Date	DATE	The date on which any of the Lookup Type By Parent metadata child elements were last changed. Clients MAY rely on this date for cache management.

METADATA-LOOKUP_TYPE_BY_PARENT

This metadata defines a list of METADATA-LOOKUP_TYPE objects for a specific value of a METADATA-LOOKUP_PARENT.

Note: Servers have the option of outputting a single record with a value of -1 for any METADATA-LOOKUP-TYPE requests for any METADATA-LOOKUP objects specified within this list. This is an indicator that the full lookup list is extremely long and that clients should use the METADATA-LOOKUP_TYPE_BY_PARENT metadata to get this information instead.

The Lookup Type By Parent metadata starts with a <METADATA-LOOKUP_TYPE_BY_PARENT> tag with Resource, Parent, Child, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-42, followed by the <DATA> section, which contains the actual field information. The Lookup Type By Parent metadata has the following format:

```

<METADATA-LOOKUP_TYPE_BY_PARENT SP Resource="resource-id" SP
Parent="parent-lookup-id=parent-lookup-value" SP Child="child-lookup-id" SP
Version="lookup-type-by-parent-version" SP Date="lookup-type-by-parent-
date"> ↵
<COLUMNS>→lookup-type-by-parent-field *(→lookup-type-by-parent-
field)→</COLUMNS> ↵
*(<DATA>→lookup-type-by-parent-data *(→lookup-type-by-parent-
data)→</DATA> ↵)
</METADATA-LOOKUP_TYPE_BY_PARENT> ↵

```

*resource-id ::= 1*32ALPHANUM*

This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Lookup Type By Parent objects belong.

*parent-lookup-id ::= 1*32ALPHANUM*

This value MUST be a LookupName found in the Lookup Parent metadata for this Resource. It is the Lookup Parent to which the Lookup Type By Parent data applies.

*parent-lookup-value ::= 1*32ALPHANUM*

This value MUST be a valid Value found in the Lookup metadata for the parent-lookup-id specified. It is the parent value for which the Lookup Type By Parent data will be output.

*child-lookup-id ::= 1*32ALPHANUM*

This value MUST be a LookupName found in the Lookup metadata for this Resource. It is the Lookup to which the Lookup Type By Parent data applies.

*lookup-type-by-parent-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS*

This is the version number of this Lookup Type By Parent metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. Every time this Lookup Type By Parent metadata changes the version number should be increased.

lookup-type-by-parent-date ::= DATE

The latest change date of this Lookup Type By Parent metadata.

lookup-type-by-parent-field ::= <Field Name from Table 11-42>

lookup-type-by-parent-data ::= <valid value as defined in Table 11-42>

An example Table section follows:

GetMetadata request:

Type: METADATA-LOOKUP_TYPE_BY_PARENT
ID Format: Resource : ParentLookup : ParentLookup Value
ID Example: Property : STATES_OR_PROVINCES : MD

Compact reply:

```
<METADATA-LOOKUP_TYPE_BY_PARENT Resource="Property"
Parent="STATES_OR_PROVINCES=MD" Child="COUNTIES_OR_REGIONS"
Version="1.2.70" Date="Wed, 12 Jan 2005 17:38:19 GMT">
<COLUMNS>□LongValue□ShortValue□Value□</COLUMNS>
<DATA>□ALLEGANY-MD□ALLEGANY□10000004177□</DATA>
<DATA>□ANNE ARUNDEL-MD□ANNE ARUNDEL□10000004178□</DATA>
<DATA>□BALTIMORE CITY-MD□BALTIMORE CITY□10000004200□</DATA>
<DATA>□BALTIMORE-MD□BALTIMORE□10000004179□</DATA>
```

```

<DATA>□CALVERT-MD□CALVERT□10000004180□</DATA>
<DATA>□CAROLINE-MD□CAROLINE□10000004181□</DATA>
<DATA>□CARROLL-MD□CARROLL□10000004182□</DATA>
<DATA>□CECIL-MD□CECIL□10000004183□</DATA>
<DATA>□CHARLES-MD□CHARLES□10000004184□</DATA>
<DATA>□DORCHESTER-MD□DORCHESTER□10000004185□</DATA>
<DATA>□FREDERICK-MD□FREDERICK□10000004186□</DATA>
<DATA>□GARRETT-MD□GARRETT□10000004187□</DATA>
<DATA>□HARFORD-MD□HARFORD□10000004188□</DATA>
<DATA>□HOWARD-MD□HOWARD□10000004189□</DATA>
<DATA>□KENT-MD□KENT□10000004190□</DATA>
<DATA>□MONTGOMERY-MD□MONTGOMERY□10000004191□</DATA>
<DATA>□OTHER-MD□OTHER□10000006307□</DATA>
<DATA>□PRINCE GEORGES-MD□PRINCE GEORGES□10000004192□</DATA>
<DATA>□QUEEN ANNES-MD□QUEEN ANNES□10000004193□</DATA>
<DATA>□SAINT MARYS-MD□SAINT MARYS□10000004194□</DATA>
<DATA>□SOMERSET-MD□SOMERSET□10000004195□</DATA>
<DATA>□TALBOT-MD□TALBOT□10000004196□</DATA>
<DATA>□UNKNOWN-MD□UNKNOWN□10000142257□</DATA>
<DATA>□WASHINGTON-MD□WASHINGTON□10000004197□</DATA>
<DATA>□WICOMICO-MD□WICOMICO□10000004198□</DATA>
<DATA>□WORCESTER-MD□WORCESTER□10000004199□</DATA>
</METADATA-LOOKUP_TYPE_BY_PARENT>

```

Table 11-42 Metadata Content – Lookup Type By Parent

Metadata Field	Content Type	Description
LongValue	1*128ALPHANUM	The value of the field as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; expected uses include displays on reports and other presentation contexts.
ShortValue	1*32ALPHANUM	An abbreviated field value that is also localizable and human-readable. Use of this field is implementation-defined; expected uses include picklist values and other human interface elements.
Value	1*32ALPHANUM	The value to be sent to the server when performing a search. This field must be numeric for LookupBitmask and LookupBitstring types. For LookupBitmask fields, the $2^{(value-1)}$ is used to compute this component as part of the applicable choices. For LookupBitstring fields, this is the position within the field, 1-based, at which the values contains a “1”.

METADATA-COLUMN_GROUP_SET

This metadata defines a tree structure which should be used to render the data in any GUI system that is designed in order to satisfy the display requirements of an MLS.

The Column Group Set metadata starts with a <METADATA-COLUMN_GROUP_SET> tag with Resource, Class, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-43, followed by the <DATA> section, which contains the actual field information. The Column Group Set metadata has the following format:

```
<METADATA-COLUMN_GROUP_SET SP Resource="resource-id" SP
Class="class-id" SP Version="column-group-set-version" SP Date="column-
group-set-date"> ↵
<COLUMNS>→column-group-set-field *(→column-group-set-
field)→↵</COLUMNS> ↵
*(<DATA>→column-group-set-data *(→column-group-set-data)→↵</DATA> ↵)
</METADATA-COLUMN_GROUP_SET> ↵
```

*resource-id ::= 1*32ALPHANUM*

This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.

*class-id ::= 1*32ALPHANUM*

This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group Set applies.

*column-group-set-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS*

This is the version number of this Column Group Set metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. Every time this Column Group Set metadata changes the version number should be increased.

column-group-set-date ::= DATE

The latest change date of this Column Group Set metadata.

column-group-set-field ::= <Field Name from Table 11-43>

column-group-set-data ::= <valid value as defined in Table 11-43>

An example Table section follows:

GetMetadata request:

Type: METADATA-COLUMN_GROUP_SET
 ID Format: Resource : Class
 ID Example: Property : RES

Compact reply:

```
<METADATA-COLUMN_GROUP_SET Resource="Property" Class="RES"
Version="1.00.000"
Date= "Sat, 20 Mar 2002 12:03:38 GMT" >
<COLUMNS>→MetadataEntryId→ColumnGroupSetName→ColumnGroupSetParent→
Sequence→LongName→ShortName→Description→ColumnGroupName→Presentation
Style→PresentationColumns→URL→</COLUMNS>
<DATA>→10000123456→Residential→→→1→Residential Listing→Residential→The top
node of the Residential Listing Data Entry Hierarchy→→→→</DATA>
<DATA>→10000123457→WaterFront→Residential→1→WaterFront
Information→WaterFront→Details about water front for
property→WaterFront→Edit→1→→</DATA>
<DATA>→10000123457→Bedrooms→Residential→2→Bedroom
Information→Bedrooms→Details about bedrooms for
property→Bedrooms→Matrix→→→</DATA>
<DATA>→10000123457→AgentInfo→Residential→3→Agent
Information→Agent→Agent Website→
→→→www.mywebsite.com/agent?Agent=.AGENTCODE.?Listing=.ListingID.→</DATA>
</METADATA-COLUMN_GROUP_SET>
```

Table 11-43 Metadata Content – Column Group Set

Metadata Field	Content Type	Description
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged. In particular, it should be managed so as to allow the client to detect changes to the ColumnGroupName.
ColumnGroupSetName	1*32ALPHANUM	The name that uniquely identifies this Column Group Set within the Class.
ColumnGroupSetParent	1*32ALPHANUM	The ColumnGroupSetName of the Parent Column Group Set. If not specified, this Column Group Set is the top node in the tree.
Sequence	1*5DIGIT	The sequence that this Column Group Set is to be displayed in.
LongName	1*64ALPHANUM	The name of the Column Group Set as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this Column Group Set when it appears on a report.
ShortName	1*32ALPHANUM	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined; it is expected that clients will use this field in

		human-interface elements such as picklists.
Description	1*256ALPHANUM	A brief description of the purpose for this Column Group Set.
ColumnGroupName	1*32ALPHANUM	The name of the Column Group that is to be displayed in this Column Group Set. If not specified, this Column Group Set is to be treated as a node in the tree that displays no data. The ColumnGroupName must exist in the Column Group metadata for this Class.
PresentationStyle	1*32ALPHANUM	One of the following values: Edit – Basic Edit Block displayed in PresentationColumns number of columns. Matrix – Expected to be displayed using Normalization Grid. List – Show one record per row. Edit List – Show one record per row and allow the records to be added, edited and deleted. GIS Map Search – Special Case: Can only have 2 columns in Column Group. First column is Latitude and Second column is Longitude. These columns are expected to be filled in with results from GIS Map Search. URL – Indicates that this is to simply go to the specified URL, a ColumnGroup name MUST not be specified and a URL MUST be specified for this PresentationStyle.
PresentationColumns	1*5DIGIT	Indicates the number of columns to display an Edit screen with. Valid values are from 1 to 4.
URL	1*256ALPHANUM	Indicates a URL that is to be accessed using this entry instead of a standard Column Group. You may not specify a ColumnGroupName and a URL. The URL may be formed with place-holders surrounded by the ‘.’ character so that a substitution for any valid SystemName or Validation Expression Special Operand Tokens as specified in Table 11-20. Example: www.mywebsite.com/agent?Agent=.AGENTCODE.?Listing=.ListingID.
ForeignKeyID	1*32ALPHANUM	The identifier of the Foreign Key that is to be displayed in this ColumnGroupSet. If specified, the ForeignKeyID MUST exist in the METADATA-FOREIGNKEY metadata and the Parent MUST be the Property and Class of this ColumnGroupSet. When this is specified, it means that a multi-row block is expected to be displayed to the user within which he can Add, Edit or Delete records of the Child Resource and Class that is specified. Furthermore, the ChildSystemName field should always be filled from data found in the ParentSystemName field to provide for a proper Master/Detail relationship.

Notes: It is important to note that only one of ColumnGroupName, ForeignKeyId or URL may contain data. These three fields are mutually exclusive.

METADATA-COLUMN_GROUPS

This metadata defines grouping element which should be used to group columns together in any GUI system that is designed in order to satisfy the display requirements of an MLS.

The Column Group metadata starts with a <METADATA-COLUMN_GROUP> tag with Resource, Class, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-44, followed by the <DATA> section, which contains the actual field information. The Column Group metadata has the following format:

```
<METADATA-COLUMN_GROUP SP Resource="resource-id" SP
Class="class-id" SP Version="column-group-version" SP Date="column-group-
date"> ↵
<COLUMNS>→ column-group-field *(→column-group-field)→</COLUMNS> ↵
*(<DATA>→ column-group-data *(→ column-group-data)→</DATA> ↵)
</METADATA-COLUMN_GROUP> ↵
```

*resource-id ::= 1*32ALPHANUM*

This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.

*class-id ::= 1*32ALPHANUM*

This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group applies.

*column-group-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS*

This is the version number of this Column Group metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. Every time this Column Group metadata changes the version number should be increased.

column-group-date ::= DATE

The latest change date of this Column Group metadata.

column-group-field ::= <Field Name from Table 11-44>

column-group-data ::= <valid value as defined in Table 11-44>

An example Table section follows:

GetMetadata request:

Type: METADATA-COLUMN_GROUP
 ID Format: Resource : Class
 ID Example: Property : RES

Compact reply:

```
<METADATA-COLUMN_GROUP Resource="Property" Class="RES" Version="1.00.000"
Date= "Sat, 20 Mar 2002 12:03:38 GMT" >
<COLUMNS>→MetadataEntryId→ColumnGroupName→ControlSystemName→
LongName→ShortName→Description→Version→Date→</COLUMNS>
<DATA>→10001123456→WaterFront→WaterFrontFlag→Water Front
Information→Water Front→Listing data that contains water front information for the
Listing→1.00.000→Thu, 3 Feb 2005 20:35:15 GMT→</DATA>
</METADATA-COLUMN_GROUP>
```

Table 11-44 Metadata Content – Column Group

Metadata Field	Content Type	Description
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged. In particular, it should be managed so as to allow the client to detect changes to the ColumnGroupName.
ColumnGroupName	1*32ALPHANUM	The name that uniquely identifies this Column Group within the Class.
ControlSystemName	1*32ALPHANUM	The SystemName of the Table Metadata that identifies the data element that is used to control the display of this Column Group.
LongName	1*64ALPHANUM	The name of the Column Group as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this Column Group when it appears on a report.
ShortName	1*32ALPHANUM	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined; it is expected that clients will use this field in human-interface elements such as picklists.
Description	1*256ALPHANUM	A brief description of the purpose for this Column Group.
Version	1*2DIGIT . 1*2DIGIT .	The latest version of the Column Group metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. The version number is advisory only.

	1*5DIGIT	
Date	DATE	The date on which any of the Column Group metadata child elements were last changed. Clients MAY rely on this date for cache management.

METADATA-COLUMN_GROUP_CONTROL

This metadata defines the valid ranges of values that the specified SystemName may have that control the display of the Column Group. If the SystemName contains any of the values that fall within the ranges specified in this table, the Column Group may be displayed. If it does not, the Column Group should not be displayed. The data is returned as a list of high and low values that determine whether the Column Group should be displayed.

The Column Group Control metadata starts with a <METADATA-COLUMN_GROUP_CONTROL> tag with Resource, Class, ColumnGroup, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-45, followed by the <DATA> section, which contains the actual field information. The Column Group Control metadata has the following format:

```
<METADATA-COLUMN_GROUP_CONTROL SP Resource="resource-id" SP
Class="class-id" SP ColumnGroup="column-group-id" SP Version="column-
group-control-version" SP Date="column-group-control-date"> ↵
<COLUMNS>→ column-group-control-field *(→column-group-control-
field)→</COLUMNS> ↵
*(<DATA>→ column-group-control-data *(→ column-group-control-
data)→</DATA> ↵)
</METADATA-COLUMN_GROUP_CONTROL> ↵
```

*resource-id ::= 1*32ALPHANUM*

This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.

*class-id ::= 1*32ALPHANUM*

This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group Control applies.

*column-group-id ::= 1*32ALPHANUM*

This value MUST be a ColumnGroupName found in the ColumnGroup metadata for this Class. It is the ColumnGroup for which the control applies.

*column-group-control-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS*

This is the version number of this Column Group Control metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. Every time this Column Group Control metadata changes the version number should be increased.

column-group-control-date ::= DATE

The latest change date of this Column Group Control metadata.

column-group-control-field ::= <Field Name from Table 11-45>

column-group-control-data ::= <valid value as defined in Table 11-45>

An example Table section follows:

GetMetadata request:

Type: METADATA-COLUMN_GROUP_CONTROL
 ID Format: Resource : Class : ColumnGroup
 ID Example: Property : RES : WaterFront

Compact reply:

```
<METADATA-COLUMN_GROUP_CONTROL Resource="Property" Class="RES"
ColumnGroup="WaterFront" Version="1.00.000" Date="Sat, 20 Mar 2002 12:03:38
GMT" >
<COLUMNS>→MetadataEntryId→LowValue→HighValue→</COLUMNS>
<DATA>→10011123456→1→1→</DATA>
</METADATA-COLUMN_GROUP_CONTROL>
```

Table 11-45 Metadata Content – Column Group Control

Metadata Field	Content Type	Description
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged. In particular, it should be managed so as to allow the client to detect changes to an individual pair of High/Low values.
LowValue	1*64ALPHANUM	The minimum value that the ControlSystemName field of the ColumnGroup is allowed to have in order to display the ColumnGroup. It is expected that the actual data type returned is interpreted as per the data type of the ControlSystemName of the ColumnGroup.
HighValue	1*64ALPHANUM	The maximum value that the ControlSystemName field of the ColumnGroup is allowed to have in order to display the ColumnGroup. It is expected that the actual data type

		returned is interpreted as per the data type of the ControlSystemName of the ColumnGroup. If the restricting data is not a range, then HighValue may be left blank.
--	--	---

METADATA-COLUMN_GROUP_TABLE

This metadata defines the set of SystemNames that are to be displayed within a Column and the order in which they are to be displayed.

The Column Group Table metadata starts with a <METADATA-COLUMN_GROUP_TABLE> tag with Resource, Class, ColumnGroup, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-46, followed by the <DATA> section, which contains the actual field information. The Column Group Table metadata has the following format:

```

<METADATA-COLUMN_GROUP_TABLE SP Resource="resource-id" SP
Class="class-id" SP ColumnGroup="column-group-id" SP Version="column-
group-table-version" SP Date="column-group-table-date"> ↵
<COLUMNS>→ column-group-table-field *(→column-group-table-
field)→</COLUMNS> ↵
*(<DATA>→ column-group-table-data *(→ column-group-table-data)→</DATA>
↵)
</METADATA-COLUMN_GROUP_TABLE> ↵

```

*resource-id ::= 1*32ALPHANUM*

This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.

*class-id ::= 1*32ALPHANUM*

This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group applies.

*column-group-id ::= 1*32ALPHANUM*

This value MUST be a ColumnGroupName found in the ColumnGroup metadata for this Class. It is the ColumnGroup for which the table applies.

*column-group-table-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS*

This is the version number of this Column Group Table metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. Every time this Column Group Table metadata changes the version number should be increased.

column-group-table-date ::= DATE

The latest change date of this Column Group Table metadata.

column-group-table-field ::= <Field Name from Table 11-46>

column-group-table-data ::= <valid value as defined in Table 11-46>

An example Table section follows:

GetMetadata request:

Type: METADATA-COLUMN_GROUP_TABLE
 ID Format: Resource : Class : ColumnGroup
 ID Example: Property : RES : WaterFront

Compact reply:

```
<METADATA-COLUMN_GROUP_TABLE Resource="Property" Class="RES"
ColumnGroup="WaterFront" Version="1.00.000" Date= "Sat, 20 Mar 2002 12:03:38
GMT" >
<COLUMNS>→MetadataEntryId→SystemName→DisplayOrder→</COLUMNS>
<DATA>→10111123450→WaterFront→1→</DATA>
<DATA>→10111123451→WaterAccess→2→</DATA>
<DATA>→10111123452→WaterFrontage→3→</DATA>
<DATA>→10111123453→WaterView→4→</DATA>
</METADATA-COLUMN_GROUP_TABLE>
```

Table 11-46 Metadata Content – Column Group Table

Metadata Field	Content Type	Description
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged.
SystemName	1*32ALPHANUM	The SystemName of the field that is to be displayed in the ColumnGroup. This MUST be a valid SystemName for this Class. A SystemName MUST be unique within the ColumnGroup.
LongName	1*64ALPHANUM	The name of the Column Group Table (data field) as it is known to the user. This is a localizable, human-readable string. Use of this field is implementation-defined; it is expected that clients will use this value as a title for this Column Group when it appears on a report.
ShortName	1*32ALPHANUM	An abbreviated field name that is also localizable and human-readable. Use of this field is implementation-defined; it is expected that clients will use this field in

		human-interface elements such as picklists.
DisplayOrder	1*5DIGIT	The order within the ColumnGroup that this SystemName is to be displayed in. DisplayOrder values MAY contain gaps and may have the same value as other columns. If multiple columns have the same value, the client SHOULD display the columns in Alphabetical order.
DisplayLength	1*5DIGIT	The number of characters to allow when displaying data for this column.
DisplayHeight	1*5DIGIT	The number of rows to display the data in. A value greater than one in this column implies a multi-line data entry field of DisplayLength width. If users enter data into this field that is longer than will fit within this text box, it is expected that the field will scroll to allow further data entry.

METADATA-COLUMN_GROUP_NORMALIZATION

This metadata defines a grid that can be used by a client to display related fields in a manner more appropriate for data entry.

The Column Group Renormalization metadata starts with a <METADATA-COLUMN_GROUP_NORMALIZATION> tag with Resource, Class, ColumnGroup, Version, and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined in Table 11-47, followed by the <DATA> section, which contains the actual field information. The Column Group Normalization metadata has the following format:

```

<METADATA-COLUMN_GROUP_NORMALIZATION SP Resource="resource-
id" SP Class="class-id" SP ColumnGroup="column-group-id" SP
Version="column-group-normalization-version" SP Date="column-group-
normalization-date"> ↵
<COLUMNS>→ column-group-normalization-field *(→column-group-
normalization-field)→</COLUMNS> ↵
*(<DATA>→ column-group-normalization-data *(→ column-group-normalization-
data)→</DATA> ↵)
</METADATA-COLUMN_GROUP_NORMALIZATION> ↵

```

*resource-id ::= 1*32ALPHANUM*

This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Classes belong.

*class-id ::= 1*32ALPHANUM*

This value MUST be a ClassName found in the Class metadata for this Resource. It is the Class to which the Column Group Normalization applies.

*column-group-id ::= 1*32ALPHANUM*

This value MUST be a ColumnGroupName found in the ColumnGroup metadata for this Class. It is the ColumnGroup for which the grid applies.

*column-group-normalization-version ::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS*

This is the version number of this Column Group Normalization metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. Every time this Column Group Normalization metadata changes the version number should be increased.

column-group-normalization-date ::= DATE

The latest change date of this Column Group Normalization metadata.

column-group-normalization-field ::= <Field Name from Table 11-47>

column-group-normalization-data ::= <valid value as defined in Table 11-47>

An example Table section follows:

GetMetadata request:

Type: METADATA-COLUMN_GROUP_NORMALIZATION
ID Format: Resource : Class : ColumnGroup
ID Example: Property : RES : WaterFront

Compact reply:

```
<METADATA-COLUMN_GROUP_NORMALIZATION Resource="Property" Class="RES"
ColumnGroup="WaterFront" Version="1.00.000" Date= "Sat, 20 Mar 2002 12:03:38
GMT" >
<COLUMNS>→MetadataEntryId→TypeIdentifier→Sequence→ColumnName→SystemName→</COLUMNS>
<DATA>→10211123450→Bedroom→1→Length→Bedroom1Length→</DATA>
<DATA>→10211123451→Bedroom→1→Width→Bedroom1Width→</DATA>
<DATA>→10211123452→Bedroom→1→Area→Bedroom1Area→</DATA>
<DATA>→10211123453→Bedroom→2→Length→Bedroom2Length→</DATA>
<DATA>→10211123454→Bedroom→2→Width→Bedroom2Width→</DATA>
<DATA>→10211123455→Bedroom→2→Area→Bedroom2Area→</DATA>
<DATA>→10211123456→Bedroom→3→Length→Bedroom3Length→</DATA>
<DATA>→10211123457→Bedroom→3→Width→Bedroom3Width→</DATA>
<DATA>→10211123458→Bedroom→3→Area→Bedroom3Area→</DATA>
<DATA>→10211123459→LivingRoom→→Length→LivingRoomLength→</DATA>
<DATA>→10211123460→LivingRoom→→Width→LivingRoomWidth→</DATA>
<DATA>→10211123461→LivingRoom→→Area→LivingRoomArea→</DATA>
```

```

<DATA>→10211123462→DiningRoom→→Length→DiningRoomLength→</DATA>
<DATA>→10211123463→DiningRoom→→Width→DiningRoomWidth→</DATA>
<DATA>→10211123464→DiningRoom→→Area→DiningRoomArea→</DATA>
<DATA>→10211123465→Kitchen→→Length→KitchenLength→</DATA>
<DATA>→10211123466→Kitchen→→Width→KitchenWidth→</DATA>
<DATA>→10211123467→Kitchen→→Area→KitchenArea→</DATA>
</METADATA-COLUMN_GROUP_NORMALIZATION>

```

Example Screen Display based on Above Compact Reply:

Type	Sequence	Length	Width	Area
Bedroom	1	Bedroom1Length	Bedroom1Width	Bedroom1Area
Bedroom	2	Bedroom2Length	Bedroom2Width	Bedroom2Area
Bedroom	3	Bedroom3Length	Bedroom3Width	Bedroom3Area
LivingRoom		LivingRoomLength	LivingRoomWidth	LivingRoomArea
DiningRoom		DiningRoomLength	DiningRoomWidth	DiningRoomArea
Kitchen		KitchenLength	KitchenWidth	KitchenArea

The user's data entry area would be the greyed out section and the data that they enter would be assigned to the SystemName in that grid position.

Table 11-47 Metadata Content – Column Group Normalization

Metadata Field	Content Type	Description
MetadataEntryId	1*32ALPHANUM	A value that never changes as long as the semantic definition of this field remains unchanged.
TypeIdentifier	1*32ALPHANUM	Y Axis – Row Label – The Label that is to be displayed on the left side of the screen that identifies the Type of data that the user is entering.
Sequence	1*5DIGIT	Y Axis – Row Sequence – The Sequence number that is to be displayed on the left side of the screen after the TypeIdentifier. This itemizes the Type of data that the user is entering.
ColumnLabel	1*32ALPHANUM	X Axis – Column Label – This is the label that is to appear at the top of the screen for data within this column. It is expected that all data in this grid with the same ColumnLabel be displayed in the same column on the screen.
SystemName	1*32ALPHANUM	The SystemName of the field that is to be displayed in this position in the Grid for the ColumnGroup. This MUST be a valid SystemName for this Class and MUST be within the ColumnGroupTable of the ColumnGroup. Fields that appear within a ColumnGroup, but not within the Normalization for the ColumnGroup, are to be treated as separate data entry fields that are not part of the grid. The SystemName MUST be unique within the ColumnGroup.

4. Development Impact

This proposal does not change anything or remove anything from the existing specification, it merely adds additional metadata fields to existing metadata tables and some additional metadata tables.

As such, development impact depends on current RETS client and server implementations, but should not have any impact on future RETS client and server implementations. Each existing RETS server and client, prior to passing of this proposal, have the option to change accordingly to be RETS compliant. Future RETS server and client implementation after approval of RETS 1.7 MUST follow the RETS 1.8 standard.

5. Compatibility

Servers that do not implement any of the additional metadata tables will remain backwards compatible to RETS 1.7. Current servers that add the additional metadata fields will also remain backwards compatible to RETS 1.7 since the specification allows for additional fields to be returned by servers. Current servers that want to become RETS 1.8 compliant MUST implement all of the additional metadata tables and fields.

6. Proof/Need of Concept Examples

MRIS Keystone 2.0 – In Alpha Testing Now.

This application is fully functional, fully metadata driven and is using all of the above features to accomplish this functionality.