

Meeting Package - April 2009 RETS Meeting

Information

There are four change proposals for the April meeting. Discussions on the changes will be discussed on the rets.org forum at <http://forums.rets.org> under the Discussion forum.

There will be limited time during the meeting to discuss the changes, so please review the materials and make your comments before the meeting.

This document consists of the four change proposals.

Paul Stusiak

RETS Change Proposal 74

Location Availability in Object Metadata

Information

Originating Workgroup: Standards Committee

Contact Information

Author: Libor Viktorin

Organization: Marketlinx Inc.

Phone: (865) 470-1500

Address: 1400 Centerpoint Blvd Ste 100,
Knoxville, TN, 37932

Email: lviktorin@marketlinx.com

Status: Accepted

Submitted Date: July 28, 2008

Voting Date: April 7th, 2009

RETS Version: 1.7.2

Synopsis

This proposal recommends adding a flag to the Object metadata, revealing whether the server supports the Location=1 request.

Rationale

Retrieving objects' URL by specifying Location=1 is an option in the GetObject request, that may not be honored by the server. Currently, if the server does not support this functionality, it SHOULD respond with an error 20414.

The problem is, a client have no means of detecting if this functionality is supported before it sends a real request, and even then (since the error 20414 is optional) may not know for sure. Servers which do not support the Location=1 option, will send the full object data, which makes for a lot of data that the client did not ask for.

This document proposes a simple method for the server to advertize its support for Location=1.

Proposal

Add a line to table 11-16 (Metadata Content: Resource Object) in chapter 11.4.1:

LocationSupport	BOOLEAN	When true, indicates that the server will honor the Location=1 parameter at least for some objects. When false, indicates that the server does not support the Location=1 functionality
------------------------	----------------	--

Modify the rets-metadata-content-1_7_2.dtd accordingly by

- adding **<!ELEMENT LocationSupport [#PCDATA]>** to the DTD
- adding **LocationSupport?**, to the Object element definition

Impact

This proposal updates the DTD by adding an optional element. Any previously existing metadata will be valid. Validation of new metadata using an old DTD would fail due to an unknown element, however this situation will be avoided by correct usage of the RETS Version.

Compatibility

This change modifies the DTD.

Document History

Date	Version	Author	Description
July 28, 2008	1.0	Libor Victorin	Initial Release
February 7, 2009	-	-	Resubmitted

RETS Change Proposal 75

Offset Availability in the Metadata

Information

Originating Workgroup: Standards Committee

Contact Information

Author: Libor Viktorin

Organization: Marketlinx Inc.

Phone: (865) 470-1500

Address: 1400 Centerpoint Blvd Ste 100,
Knoxville, TN, 37932

Email: lviktorin@marketlinx.com

Status: Accepted

Submitted Date: July 28, 2008

Voting Date: April 7th, 2009

RETS Version: 1.7.2

Synopsis

This proposal recommends adding a flag to the Resource metadata, revealing whether the server supports the Offset parameter in the Search transaction.

Rationale

The Offset parameter in the Search transaction is used by many clients to iterate thru large blocks of data. However, there are many servers that do not support this functionality. With such servers, the clients keep requesting with increasing offset number, but getting the same data all the times. The specification does not give any way to let the client know whether the Offset feature is supported or not.

This document proposes a simple addition to the metadata which will show the server's support for the Offset feature on a per-class level.

Proposal

Add a line to table 11-7 (Metadata Content: Resource Class) in chapter 11.3.1:

OffsetSupport	BOOLEAN	When true, indicates that the server will honor the Offset parameter when searching this class. When false, indicates that the server does not support the Offset functionality for this class.
---------------	---------	---

Modify the rets-metadata-content-1_7_2.dtd accordingly by

- adding **<!ELEMENT OffsetSupport [#PCDATA]>** to the DTD
- adding **OffsetSupport?**, to the Class element definition

Impact

This proposal updates the DTD by adding an optional element. Any previously existing metadata will be valid. Validation of new metadata using an old DTD would fail due to an unknown element, however this situation will be avoided by correct usage of the RETS Version.

Compatibility

This change modifies the DTD.

Document History

Date	Version	Author	Description
July 28, 2008	1.0	Libor Victorin	Initial Release
February 7, 2009	-	-	Resubmitted

RETS Change Proposal 76

GetPayloadList

Information

Originating Workgroup: Transport Workgroup

Contact Information

Author: Steve Clarke

Organization: Marketlinx Inc.

Phone: (505) 286-4800

Address: 1400 Centerpoint Blvd Ste 100,
Knoxville, TN, 37932

Email: sclarke@marketlinx.com

Status: Accepted

Submitted Date: February 26, 2009

Voting Date: April 7th, 2009

RETS Version: 1.7.2

Synopsis

The objective is to define a discovery mechanism for predefined XML payload documents that can optionally be accessed via RETS 1.x queries. This proposal defines a new transaction that advertises the available payloads on a RETS 1.x server and describes how these payloads would be accessed from a RETS 1.x query request.

Rationale

There has been a longstanding desire to be able to access standardized search results (payloads) via a RETS 1.x query request. Most urgently, the RESO community is looking for a way to provide a transport mechanism to support the Syndication XSD. This RCP satisfies that need.

Proposal

3.1 Login Response

Add new *GetPayloadList* transaction definition to login response.

Example:

```
<RETS ReplyCode="0" ReplyText="Success.">
  <RETS-RESPONSE>
    Info=MEMBERNAME;Character;MarketLinx Sys Config
    Info=USERID;Character;866424041
    Info=USERLEVEL;Int;90
    Info=USERCLASS;Character;!P
    Info=AGENTCODE;Character;sys_config
    Info=BROKERCODE;Character;GEAC
    Info=BROKERBRANCH;Character;GEAC01
    Info-METADATAID;Character;303_65_47_BRC
    Info=METADATAVERSION;Character;17.73.76591
    Info=METADATATIMESTAMP;DateTime;Tue, 12 Feb 2008 23:16:31 GMT
    Info=MINMETADATATIMESTAMP;DateTime;Tue, 12 Feb 2008 23:16:31 GMT
    Info=BOARD;Character;X
    Info=BROKERRECIPFLAG;Boolean;Y
    Info=MAINOFF;Character;GEAC
    Info=OFFICE;Character;GEAC01
    Info=SUL;Int;90
    Info=UC;Character;!P
    Info=USER;Character;sys_config
    ChangePassword=/ChangePassword.aspx/ChangePassword
    GetObject=/GetObject.aspx/GetObject
    Login=/Login.aspx/Login
    Logout=/Logout.aspx/Logout
    Search=/Search.aspx/Search
    GetMetadata=/GetMetadata.aspx/GetMetadata
    Update=/Update.aspx/Update
    PostObject=/PostObject.aspx/PostObject
    GetPayloadList=/GetPayloadList.aspx/GetPayloadList
  </RETS-RESPONSE>
</RETS>
```

3.2 *GetPayloadList* Transaction

The GetPayloadList transaction is used to retrieve a list of available payloads supported on the RETS Server. Payloads are defined as a subset of the “RESO schema” vocabulary.

Required Client Request Header Fields

There are no additional required client header fields.

Required Request Arguments

None.

Optional Request Arguments

ID

The ID argument in the GetMetadata transaction reflects the metadata hierarchy as shown in Figure 11.1. For any metadata element, the ID argument is a list of the names of the parent elements for the desired element, separated by colons. For example, to retrieve the payload list for a given named Resource, the argument is simply the ResourceID.

[/GetPayloadList?ID=Property](#)

To retrieve the payload list for a specific class within a resource:

[/GetPayloadList?ID=Property:RES](#)

Format

The format option does not apply to the *GetPayloadList* request because the response is always considered to be in COMPACT format. See below.

Required Server Response Header Fields

None.

Required Response Arguments

There are no required response arguments.

Optional Response Arguments

There are no optional response arguments.

GetPayloadList Response Body Format

Example COMPACT reply:

```
<RETSPayloadList Resource="Property" Class="2" Version="1.00.000" Date="2009-02-11T12:17:12-05:00" >
<COLUMNS>→PayloadName→Resource→Class→Description→URI→MetadataEntryID</COLUMNS>
<DATA>→RESO Syndication→Property→1→RESO Standard Syndication
Payload→http://rets.org/xsd/Syndication.xsd→1x123</DATA>
<DATA>→RESO Syndication→Property→2→RESO Standard Syndication
Payload→http://rets.org/xsd/Syndication.xsd→2x123</DATA>
<DATA>→RESO Syndication→Property→3→RESO Standard Syndication
Payload→http://rets.org/xsd/Syndication.xsd→3x123</DATA>
</RETSPayloadList>
```

Note: If there are multiple versions of a payload, the URI for the payload be the **Namespace URI** for the version of the schema that is supported. For example, one of the syndication versions is <http://rets.org/xsd/Syndication/2008-03>.

Reply Codes

RETS 1.7 requires all server responses to be well-formed XML, and additionally requires *GetPayloadList* responses to be valid XML. In addition, RETS requires that clients parse server responses as XML, not as simple text streams. The response formats shown here are normative with respect to content, but not normative with respect to form. That is, servers are free to produce response XML in any format that complies with the W3C XML 1.0 recommendation, so long as it is valid with respect to the appropriate DTD. XML escaping of content is implied, as is XML processing of white space and line endings. See the W3C *XML Recommendation 1.0, Third Edition*, for full information on XML.

Table X-X *GetPayloadList* Reply Codes (Sheet 1 of 2)

Reply Code Meaning

- 0 – Operation successful.
- 20500 Invalid Resource The request could not be understood due to an unknown resource.
- 20503 No Metadata Found No matching metadata of the type requested was found.
- 20511 Timeout The request timed out while executing.
- 20513 Miscellaneous error The server encountered an internal error.

3.3 Search Transaction

```
/Search?SearchType=Property&Class=2&Query=(ListPrice=300000-)  
&QueryType=DMQL2&Count=0&Payload=RESO Syndication
```

Note, a search request that specifies a **&Payload** **MUST not** include the **&Select** and **&Format** arguments. The payload itself defines the desired data elements and the format of the response.

Additional Reply Codes:

(ReplyCodes tbd because these should be maintained centrally by document manager)

- Payloads not supported at all.
- Invalid Payload for this Resource/Class
- Cannot combine &Select with &Payload.
- Cannot combine &Format with &Payload.

Impact

Compatible with 1.7.2 and above. This is a new transaction which is self-contained. A server can simply leave this out of their capabilities URL list in the Login response. If no Payloads are advertised, then none need to be supported in the search request/response.

Compatibility

Document History

Date	Version	Author	Description
December 1, 2008	1.0	Steve Clarke	Initial Release
February 26,2009	1.1	Steve Clarke	Revisions from review

RETS Change Proposal 77

Maximum Field Length

Information

Originating Workgroup: Standards Committee

Contact Information

Author: Matthew McGuire

Organization: Marketlinx Inc.

Phone: (865) 470-1500

Address: 1400 Centerpoint Blvd. Suite 100, Knoxville, TN 37932

Email: mmcguire@marketlinx.com

Status: Accepted

Submitted Date: February 26, 2009

Voting Date: April 7th, 2009

RETS Version: 1.7.2

Synopsis

This proposal recommends clarification of the meaning of the ***MaximumLength*** element in the *METADATA-TABLE* metadata. This proposal does not change the meaning of the ***MaximumLength*** element.

Rationale

The current specification defines the ***MaximumLength*** field as the ‘maximum possible un-encoded length of a value’ of the field. The definition refers to HTTP encoding specifically since RETS uses the HTTP transport for communication. Since this definition includes the term un-encoded it is possible that some developers interpret this with regard to a Lookup Value to Long Value encoding. The definition refers to Appendix D for examples in an attempt to clarify this. In Appendix D the examples show that the max length for string values applies to the Lookup Value not Lookup Long Value. Since this is not explicit developers have found this language to be ambiguous or confusing. This proposal attempts to clarify how the ***MaximumLength*** metadata value should be calculated based on the data type of the field.

Proposal

To solve this ambiguity define a formula of calculating the maximum character length of a

given field using existing metadata information. A formula is defined for each of the appropriate RETS Data Types in the appendix. This will provide clear guidelines for how each data type should use the **MaximumLength** metadata information.

It is the responsibility of the client to accurately calculate storage requirements locally, based on the metadata provided by the RETS server. However, server vendors should adhere to the formulas presented here so there is consistent functionality using the **MaximumLength** metadata information.

Example: The maximum storage length for a multi-valued lookup field is calculated with the following formula.

$$(\text{MaxSelect} * (\text{MaxValueLength} + 3)) - 1$$

The following variables are used in this calculation:

MaxSelect - The MaxSelect element (METADATA-TABLE). If MaxSelect is not provided, then we assume it is 1. This is a factor because if a field allows multiple selections, then the data storage will require space for the maximum number of instances of the data, potentially enclosed in quotes and separated by commas.

Example: "HW", "WWC", "CT"

MaxValueLength - The length of the longest Value entry (METADATA-LOOKUP_TYPE) that applies to the field being stored. The length of the Lookup Value is used in this formula assures that clients can handle any and all of the possible lookup values in the result data.

Note that both Lookup and LookupMulti fields can use the same calculation formula defined above.

Specification Changes

Clarify the **MaximumLength** definition and properly refer to the updated appendix D. Then add the maximum length calculation formulas to the Maximum Length appendix D.

Section 11.3.2 – Table

This section defines the Table Metadata used to define Field information. Table 11-12 defines the **MaximumLength** property of a field. This definition will change to the following:

The maximum possible character length of the value of the field after all Transport layer encoding. Transport layer encoding includes both HTTP and XML based encoding, but does not include RETS Lookup Value to Lookup Long Value encoding. See Appendix D for a definition and examples of how a RETS server should calculate the **MaximumLength** of a field based on the RETS data type.

Appendix D – Maximum Length

The language of this appendix will change to the following:

Opening:

This appendix defines formulas used to determine the maximum character length of field data based on the data type of the field. These formulas describe how the RETS server should calculate the **MaximumLength** for reliable use by RETS client applications.

Appendix D.1 – Maximum Length - Boolean

Any field with the Boolean data type should represent the **MaximumLength** as '1'. The definition of the Boolean data type requires a single character representation of the data.

Appendix D.2 – Maximum Length - Characters

Fields designated as the Character data type with an interpretation of Lookup or LookupMulti should calculate the **MaximumLength** according to the following:

$$(\text{MaxSelect} * (\text{MaxValueLength} + 3)) - 1$$

Multiply the Max Select of the field (or 1 if not provided) by the character length (plus 3 for delimiters and quotes) of the longest Lookup Value in the Lookup metadata defined by the Lookup Name of the field. Finally subtract 1 to represent one less delimiter.

Example: The Field 'Appliances' has 10 Lookup items and has a Max Select of 4 lookup items. The longest Lookup Value character length is 6 for the Value 'FRIDGE' used for the Long Value 'Refrigerator'. $(4 \times (6 + 3)) - 1 = 35$

Appendix D.3 – Maximum Length - Integers

Fields designated as Integer numbers include the Tiny, Small, Int, and Long data types. These numbers are limited in size by their respective binary representations. Therefore a Tiny number is 8 bits and has a maximum value of 256. This results in a Maximum Character Length of 3. This can also be called the maximum Decimal Precision of the number. The Maximum Length of the field if advertised should be equal to or greater than the maximum Decimal Precision of the number. For ease of use each are provided here:

Tiny – ± 4 – 2 characters

Small – ± 32786 – 6 characters

Int – ± 2147483648 – 11 characters

Long – ±9223372036854775808 – 20 characters

Appendix D.4 – Maximum Length - Decimal

Fields designated as Decimal only include numbers represented using a Decimal point. The Precision of the field determines the maximum number of decimal characters following the decimal point of the number. However, the maximum decimal precision of the data includes the decimal spaces before the decimal as well. For a decimal number the Maximum Character Length should match the maximum decimal precision of the value plus one to represent the decimal point itself. So that a signed 16 bit floating point number with a precision of 3 is a maximum value of ±32.786 which is 7 characters. The RETS standard does not define Decimal numbers by binary size so the server should advertise the Maximum Length as appropriate to the data exposed via the RETS server interface.

The Currency interpretation follows the same rules as a Decimal number with 2 decimal points of precision.

Impact

None

Compatibility

This change proposal strictly defines the Maximum Length property of the Metadata which may be interpreted differently by older applications. However, the previous language of the specification implies the same meaning so these changes should be backward compatible.

Document History

Date	Version	Author	Description
February 26, 2008	1.0	Matt McGuire	Initial Release
March 2, 2009	1.1	Matt McGuire	Revisions from review