Paragon RETS
Developer's Start Guide

Abstract

This guide is intended as a starting point for Paragon Real Estate Transaction Standard (RETS) developers. It provides a basic overview of RETS and describes some useful techniques for beginning to work with Paragon RETS.

Bruce Wolven

Paragon RETS Best Practice Guide (Vendor) | v 2.2.1 | October 3, 2024

© 2024 Intercontinental Exchange, Inc.

Table of Contents

© 2024 Intercontinental Exchange, Inc.

Paragon RETS Developer's Start Guide

This document was last changed on October 8th, 2023October 2, 2024, by Joe Brueske

The Purpose of This Guide

This guide is intended as a starting point for Paragon Real Estate Transaction Standard (RETS) developers. It provides a basic overview of RETS and describes some useful techniques for beginning to work with Paragon RETS.

RESO has deprecated RETS in favor of the Data Dictionary Web API. RETS documentation is no longer available on their site. If you require the specifications, please reach out to us with any of the communication methods on our Vendor Support site. Information about RESO Web API can be found on their website at: https://www.reso.org/reso-web-api/.

What is RETS?

According to the RETS 1.8.0 Specification:

The Real Estate Transaction Standard (RETS) is a specification for a standard communication method between computer systems exchanging real estate information. It defines a standard interface for use by applications such as agent desktop software, IDX (Internet Data Exchange) systems, data aggregation systems, and many other systems that store, display or operate on real estate listing, sales and other data.

The Real Estate Transaction Standard is a common language spoken by systems that handle real estate information, such as multiple listing services. A common language enables computers like the one on your desk to receive information from many different Organizations (MLS, Association, Board, etc.) without being specially "trained" to understand the information from each.

Currently Paragon RETS supports RETS versions: RETS/1.5, RETS/1.7.2, and RETS/1.8. Although we would prefer if clients use the highest supported version if at all possible. Support for earlier versions may be removed in the future.

Paragon RETS Transaction Types

There are several transaction types RETS servers may support, some are optional and some required. The key transaction types client applications need to download property data using RETS are **Login, Logout**, **GetMetadata**, **Search**, and **GetObject**, all of which are provided by the Paragon server. There are some additional transactions: **Update** and **PostObject**, which may also be available depending on the application requirements and access allowed by the Organization. In this section we give brief descriptions of each of these transaction types. For examples of how to issue requests for each transaction type, see Getting Started.

Login

Before downloading property data, the client application must authenticate to prove the user has access to Paragon RETS, i.e., provide a valid user ID and password. This is done via the Login transaction. All RETS transactions **must be** done with HTTPS. Should you have issues with login, please verify the following:

- URL Format: mlsid-rets.paragonrels.com
- TLS Library v1.2
- Basic Authentication
- Case Sensitive Username and Password

Depending on your application requirements (i.e. if you need Update and PostObject support), you may also be required to provide a second level of authentication, client application authentication, which is handled by calculating and sending another header in your requests: RETS-UA-Authorization (Sections 3.4

© 2024 Intercontinental Exchange, Inc.

and 3.10 in the RETS specification). This requires special configuration on the RETS Server side and a special key will be provided for you to use in calculating the RETS-UA-Authorization header.

Logout

Once the RETS session is complete, a client application should **ALWAYS** use the **Logout** transaction to close the session on the server. This frees session resources on the server. Users with too many open sessions could be labeled as a network threat or bad behaving client and get disabled or blocked.

GetMetadata

The **GetMetadata** transaction is used to pull the RETS metadata from the RETS server.

RETS metadata is data describes data. It contains information about types, lengths, field names, and whether or not the field can be searched, etc.

Although all RETS providers carry property information, the specific information available will vary from one provider to another, and sometimes between one user and another depending on specific access requirement. Each provider may use different field names for the data that is returned. RETS metadata is used to determine what information is available from a specific site.

Search

The **Search** transaction is the primary transaction used to retrieve property and other information from a RETS server.

With a properly formulated Search transaction request, a client may specify a search type (e.g., Agent, Property, etc.), search criteria, data fields to be retrieved, the maximum number of results to return, etc. A full description of the request options for a Search transaction can be found in the RETS specifications.

NOTE: When requesting data in COMPACT or COMPACT-DECODED format, never assume a particular column order. **ALWAYS** use the COLUMNS tag to determine the actual order of the data in the result set and to ensure a particular column exists. Organization business rules and user account settings could restrict or remove columns from the resulting data. If you need a specific order or set of fields, use the Search transaction’s **Select** parameter to specify exactly what fields you need returned and the order in which they should be returned.

GetObject

The **GetObject** transaction provides a means of downloading objects associated with a given record. There are a few different ObjectTypes that can be retrieved, each of which require the following **Location** parameter.

- **Location = 1** - A link (URL) for the object is returned.

NOTE: Property image URL's should be pulled via the **Media** resource and not **GetObject**. Please contact Organizations for access. Only Documents, Office Logos, and Agent Photos require **GetObject**.

Update

The **Update** transaction provides a way to update property and other information through RETS. Access to this transaction requires Organization approval and usage may be limited by Organization business rules and user’s configuration and access rights.

PostObject

The **PostObject** transaction provides a way to upload photos and documents to be associated with property information. Access to this transaction requires MLS approval and usage may be limited by Organization business rules and user’s configuration and access rights.

© 2024 Intercontinental Exchange, Inc.

Getting Started

In this section, we show you how to get started by issuing RETS transaction requests directly from your favorite browser. This will allow you to experiment with the various transaction types and get a feel for how a RETS client interacts with a RETS server.

Sample Login Transaction

To issue a Login transaction request, type the following into your browser, substituting the MLSID provided to you by the Organization:

```
https://mlsid-rets.paragonrels.com/rets/fnisrets.aspx/mlsid/login?rets-version=rets/1.8
```

You will be prompted for a login name and password. Once you have successfully logged in, you should see an XML response similar to the one shown below (sample RETS/1.8 response):

```
<RETS ReplyCode="0" ReplyText="Login Request" >
        <RETS-RESPONSE>
        Info=USERID;Character;YourUserName
        Info=USERLEVEL;Int;9
        Info=USERCLASS;Character;1
        Info=AGENTCODE;Character;6
        Info=BROKERCODE;Character;1
        Info=BROKERBRANCH;Character;1
        Info=MEMBERNAME;Character;YourName
        Info=MetadataVersion;Character;16.3.25718
        Info=MetadataTimestamp;DateTime;2016-03-17T20:38:09.9Z
        Info=MinMetadataTimestamp;DateTime;2016-03-17T20:38:09.9Z
        Info=TimeoutSeconds;Int;5400
        Info=VendorName;Character;ICE
        Info=ServerProductName;Character;RETS-Paragon
        Info=ServerProductVersion;Character;1.0
        Info=OperatorName;Character;OrganizationName
        GetMetadata=/Rets/FNISRETS.aspx/RAYAC/getmetadata
        GetObject=/Rets/FNISRETS.aspx/RAYAC/getobject
        Login=https://localhost/Rets/FNISRETS.aspx/RAYAC/login
        Logout=/Rets/FNISRETS.aspx/RAYAC/logout
        Search=/Rets/FNISRETS.aspx/RAYAC/search
        Update=/Rets/FNISRETS.aspx/RAYAC/update
        PostObject=/Rets/FNISRETS.aspx/RAYAC/postobject
        </RETS-RESPONSE>
</RETS>
```

Notice that this response includes a capability list (list of transactions) that are available to your account. In the example above it shows transactions (GetMetadata, GetObject, Login, Logout, Search, Update, PostObject) are available on this RETS server for the user account.

Note: Do not assume any particular order for the items returned in the RETS-RESPONSE values. Also there could be spaces surrounding the "=" sign.

Sample GetMetadata Transactions

The following sample GetMetadata transaction will return a list of the Resources (search types) available from your RETS server.

```
https://mlsid-rets.paragonrels.com/rets/fnisrets.aspx/mlsid/getmetadata?Type=METADATA-
RESOURCE&Format=COMPACT&ID=0
```

If you are familiar with HTTP syntax, you will notice this transaction request passes a list of parameters and values as follows:

- Type=METADATA-RESOURCE

© 2024 Intercontinental Exchange, Inc.

- Format=COMPACT
- ID=0

The following describes these parameters in greater detail. The parameters marked with a '*' are required for all GetMetadata transactions.

***Type**: The type of metadata to be returned. A complete list of legal values for this parameter can be found in the RETS specification document, but the most useful values for most Paragon RETS users will be:

- METADATA-SYSTEM -- General system information
- METADATA-RESOURCE -- A list of the Resources (search types) available on the server
- METADATA-CLASS -- A list of the Classes (search subtypes) available on the server, organized by Resource
- METADATA-TABLE -- A list of the data fields available on the server, organized by Resource and Class
- METADATA-OBJECT -- A list of the object types (e.g., photographs) available on the server, organized by Resource
- METADATA-LOOKUP -- A list of the lookup tables (used for drop-down lists) available on the server, organized by Resource.
- METADATA-LOOKUP_TYPE -- A list of the codes and values in each lookup table, organized by Resource and lookup table name.

Format: The XML format in which the data should be returned. Currently Paragon RETS supports these formats:

- COMPACT - Data is arranged in a simple tabular form.
- STANDARD-XML – A more verbose format with a more strict XML structure.

***ID:** A parameter to restrict the metadata returned, e.g., to return the available data fields for Agent search types only. For details on how to use this parameter, consult the RETS Specification document. A value of '0' means no restriction; all the available metadata of the specified type is returned.

In Basics of Using Paragon Metadata below, we present some sample metadata and show you how to get started interpreting it. You are encouraged to experiment with variations on the query above to get a feel for what the different types of metadata look like.

Sample Search Transaction

The following sample Search transaction will return listing numbers and prices for residential properties whose price is greater than $300,000. (Note that this query assumes that "ResidentialProperty" is one of the data classes available for the "Property" search type; if this is not true for your Paragon RETS server, you can run this transaction by substituting a valid class. Available classes for each search type can be found in the metadata.)

```
https://mlsid-rets.paragonrels.com/rets/fnisrets.aspx/mlsid/search?SearchType=Property
&Class=ResidentialProperty&QueryType=DMQL2&Format=COMPACT&StandardNames=1&Select=ListingID,Lis
tPrice&Query=(ListPrice=300000%2B)&Count=1&Limit=10
```

(Note that in non-browser-based RETS requests, the above query parameter would be Query=(ListPrice=300000+). However, if you enter the query directly into a browser address, you need to use the ascii code "%2B" instead of "+".)

This transaction request passes the following list of parameters and values:

- SearchType=Property
- Class=ResidentialProperty
- QueryType=DMQL2
- Format=COMPACT
- StandardNames=1
- Select=ListingID,ListPrice

© 2024 Intercontinental Exchange, Inc.

- Query=(ListPrice=300000+)
- Count=1
- Limit=10

The following describes these parameters in greater detail. The parameters marked with a '*' are required for all Search transactions.

***SearchType:** The type of Organization data that the search is to return. There are nine standard search types (called "Resources") defined in the RETS specification; every RETS server is required to support at least one of these, and to report all supported resources in its metadata. The 'Property' resource, which encompasses the actual listing data, is provided by most if not all RETS servers.

***Class:** The subtype of the data to be returned. Every resource has at least one class associated with it, while some may have several; e.g., the Property resource may include separate classes for single-family homes, condominiums, rental units, etc. Classes are not specified in the RETS standard, and may vary widely from one RETS provider to another. They are, however, required to be listed in the metadata.

***QueryType:** The query language in which the **Query** parameter is expressed. This parameter exists as a placeholder in case multiple query languages become supported in later expansions of RETS; at present, however, its value is always 'DMQL2'.

Format: The XML format in which the data should be returned. These are the supported formats:

- COMPACT - Data is arranged in a simple tabular form. Fields with Interpretation set to Lookup or LookupMulti in the metadata are returned as stored.
- COMPACT-DECODED – Similar to COMPACT except fields with Interpretation set to Lookup or LookupMulti in the metadata are returned in expanded form substituting the LongValue contained in the lookups METADATA-LOOKUP_TYPE for the Value that is normally returned.
- STANDARD-XML – A more verbose format with a more strict XML structure. Currently this format is only supported in limited fashion by Paragon RETS.

StandardNames: This parameter may have a value of either 0 or 1. If a value of 1 is specified, then the RETS server will interpret the field selection list and query as being expressed in terms of *standard names*, which should be the same from one RETS server to the next. If the value is 0, or if the parameter is not specified, then the server assumes local, server-specific system names are being used instead. (For more details on standard vs. system names, see Basics of Using Paragon Metadata.)

Select: A comma-separated list of the fields whose values should be returned in the order you desire them returned in the search results. If this parameter is not specified, all available fields for the specified resource and class are returned.

***Query:** The actual search query for the transaction. A detailed specification of the DMQL2 query language is outside the scope of this document, but may be found in the RETS specifications.

Count: The count type for the search; this parameter may have a value of 0, 1 or 2.

- **Count = 0** or **Omitted** – Only the matching data is returned in the results.
- **Count = 1** - Total number of matching records is returned in a COUNT tag along with the results.
- **Count = 2** – Only the total number of matching records is returned in a COUNT tag. No result data returned.

Limit: The maximum number of results to be returned for the query. Please note, Paragon RETS has configurable, internal limits which are imposed on data returns, which could limit the amount of data returned, even if the Limit parameter is set higher.

Offset: The 1 based index into the search result set. This is used to overcome data record return limits.

© 2024 Intercontinental Exchange, Inc.

For Example:

If you specify an offset of 1001, a query with no specified limit would return the 1001st - 3500th matching records (assuming the default limit of 2500) or the 1001st through last, if there are not 3500 matching records. If the Limit parameter were set to 10, the search would return the 1001st through 1010th records. (For more information, see Question 3 in Frequently Asked Questions below.)

There are a number of other optional parameters to the Search transaction request; these are documented in detail in the RETS specification document.

The results of the sample query will look similar to the following:

```
<RETS ReplyCode="0" ReplyText="SUCCESS" >
<COUNT Records="20095"/>
<DELIMITER value="09" />
<COLUMNS>     ListingID ListPrice </COLUMNS>
<DATA> 803217  475000  </DATA>
<DATA> 810341  369900  </DATA>
<DATA> 901762  690000  </DATA>
<DATA> 902097  499000  </DATA>
<DATA> 904353  369900  </DATA>
<DATA> 907441  485000  </DATA>
<DATA> 910131  360000  </DATA>
<DATA> 910315  429900  </DATA>
<DATA> 910331  357900  </DATA>
<DATA> 910365  449900  </DATA>
<MAXROWS/>
</RETS>
```

Sample GetObject Transaction

The following sample GetObject transaction will download a photograph for a listing:

```
https://mlsid-
rets.paragonrels.com/rets/fnisrets.aspx/mlsid/getObject?Resource=Property&Type=Photo&Id=910131:0
```

This transaction request passes the following list of parameters and values:

- Resource=Property
- Type=Photo
- Id=910131:0

The following describes these parameters in greater detail. All three parameters are required for all GetObject transactions.

***Resource:** The Resource with which the object is associated, e.g., a property photo would be associated with the "Property" resource, while an agent's photo or office logo might be associated with the "Agent" resource. Available resources for your RETS server are listed in the metadata (query for METADATA-RESOURCE).

***Type:** The type of object to be retrieved. The allowable types for your RETS server can be found in the object metadata (Type=METADATA-OBJECT).

***Id**: The identifier for the record with which the object is associated. For property listings, this would normally be the listing Id but should always be whatever value of whatever field is listed as the KeyField in the resource metadata (Type=METADATA-RESOURCE). The ":0" at the end of the sample ID parameter is the ObjectID and in this case indicates that the "preferred" photograph should be returned, which in Paragon RETS is always the first one. Note: RETS photo indexes are 1 based with 0 indicating the primary or preferred object. Other options are:

- ":n" to return the nth photograph (e.g., ":2" would return the second)
- ":*" wildcard to return all photographs for the ID value

© 2024 Intercontinental Exchange, Inc.

If more than one photo is being requested the value is returned in multi-part mime format.

You can combine requests to request images from several properties at once. For example:

- 910131:1:3:4 (return photos 1,3 and 4 for specified property)
- 910131:*, 907441:1:5 (return all photos for property 910131 and 1st and 5th photo for 907441)
- 910131, 910131 (return preferred photo for both properties same as :0)

Basics of Using Paragon Metadata

In Getting Started above, we offered the following sample metadata query:

```
https://mlsid-rets.paragonrels.com/rets/fnisrets.aspx/mlsid/getmetadata?Type=METADATA-
RESOURCE&Format=COMPACT&ID=0
```

Suppose that you submitted this query and got the following XML results back:

```
<RETS ReplyCode="0" ReplyText="SUCCESS" >
<METADATA-RESOURCE Version="16.3.25718" Date="2016-03-17T20:38:09.9Z">
<COLUMNS>      ResourceID      StandardName  VisibleName     Description        KeyField ClassCount
        ClassVersion     ClassDate        ObjectVersion    ObjectDate       SearchHelpVersion
        SearchHelpDate   EditMaskVersion  EditMaskDate     LookupVersion    LookupDate
        UpdateHelpVersion         UpdateHelpDate  ValidationExpressionVersion
        ValidationExpressionDate  ValidationLookupVersion  ValidationLookupDate
        ValidationExternalVersion ValidationExternalDate   </COLUMNS>
<DATA> Property Property Property All Property Listings     L_ListingID     5       16.3.24004
        2016-03-16T16:04:17.8Z 16.3.3955      2016-03-02T17:55:43.1Z 0.0.0          15.7.23907
        2015-07-16T14:27:55.3Z 16.1.42909     2016-01-29T19:09:35.6Z 0.0.0          0.0.0
        0.0.0            0.0.0            </DATA>
<DATA> Agent    Agent    Agent    All Agents       U_AgentID       1       16.3.22426      2016-03-
15T13:46:39.8Z  14.5.39812       2014-05-27T15:32:03.4Z 0.0.0           15.7.23907      2015-07-
16T14:27:59.2Z  16.1.42910       2016-01-29T19:10:25.9Z 0.0.0           0.0.0           0.0.0
        0.0.0            </DATA>
<DATA> Office   Office   Office   Offices  O_OfficeID      1       16.1.42910      2016-01-
29T19:10:27.5Z  0.0.0            0.0.0            15.7.23908      2015-07-16T14:28:07.4Z 16.1.42910
        2016-01-29T19:10:27.3Z 0.0.0            0.0.0            0.0.0           0.0.0
        </DATA>
<DATA> Media    Media    Media    Media    MED_listing_media_id    5       16.3.25695      2016-03-
17T20:15:25.4Z  0.0.0            0.0.0            15.7.23909      2015-07-16T14:29:01.5Z 16.1.42822
        2016-01-29T17:42:20.6Z 0.0.0            0.0.0            0.0.0           0.0.0
        </DATA>
</RETS>
```

The <COLUMNS> entry tells you how to interpret the <DATA> lines that follow. For example, the first element of each <DATA> line is a Resource ID, the fourth is a description of the resource, and the sixth is the number of classes belonging to the resource. This particular output would indicate that the server supports three Resources (search types): "Property", "Agent", "Office", and “Media” resources.

© 2024 Intercontinental Exchange, Inc.

Now, consider the following (truncated) sample results from a METADATA-TABLE query:

```
RETS ReplyCode="0" ReplyText="SUCCESS" >
<METADATA-TABLE    Resource="Property"    Class="RF_1"    Version="16.3.22725"    Date="2016-03-
15T18:45:22.3Z">
<COLUMNS>    MetadataEntryID  SystemName    StandardName    LongName    DBName ShortName
        MaximumLength  DataType        PrecisionSearchable    Interpretation    Alignment
        UseSeparator   EditMaskID      LookupName     MaxSelect      Units     Index    Minimum
        Maximum        Default  Required SearchHelpID  Unique  ModTimeStamp  ForeignKeyName
        ForeignField   KeyQuery        KeySelect      InKeyIndex     FilterParentField
        DefaultSearchOrder       Case     </COLUMNS>
<DATA> 0166D2F74FDC2F40      L_ListingID     ListingIDSystemID        SystemID        SystemID
        10      Int             1               Right   0       int_10
                1                               1       0                                       1
                0               </DATA>
<DATA> 0166D2F74FDC3AF8      L_Area  ListingArea     School District     SchlDist Schl Dist5
        Small           1       Lookup  Right   0       int_5   Area
                6                       0       0                               0
        0               </DATA>
<DATA> 0166D2F74FDC42C8      L_AskingPrice   ListPrice List Price    LOrP    L/P     10      Int
                1       Currency        Right   0       int_10
                                        0       0                               0
        0               </DATA>
.
.
.
</METADATA-TABLE>
</RETS>
```

At first glance, this example doesn't appear to follow the interpretation guidelines previously provided. After all, there are 23 column names, but none of the data lines seems to have more than 10 entries!

The reason for this is some of the metadata values for any given field may be empty. Because the <DATA> line elements are tab-separated, empty values are difficult to detect visually, and column values will usually not line up under the column headers. It is up to each client RETS program to properly interpret the XML, recognizing tabs as separators and that two sequential tabs indicate a null value in the corresponding field.

Once properly interpreted, the metadata above gives information about three fields: "L_ListingID" (this is the keyfield for the resource), "L_Area" (school district), and "L_AskingPrice" (List Price). You will also be able to tell that L_AskingPrice is of type Currency, and L_Area contains encoded values with associated lookup table ("Area" n this case).

Note: "L_ListingID", "L_Area" and "L_AskingPrice " are the *System Names* for these fields, and can therefore be used in search transactions where the parameter: **StandardNames=0** or not set at all (see Sample Search Transaction, above). The *Standard Names* for these particular fields are populated in the metadata, which means these fields can be used in a search transaction where the parameter: **StandardNames=1**.

NOTE: Standard Name values are limited and many of the fields will not have this value populated. Many clients choose to use the System Name values to reference the fields.

© 2024 Intercontinental Exchange, Inc.

Frequently Asked Questions

In this section we answer some questions that are frequently asked by developers and users new to Paragon RETS.

It looks like there's something wrong with the dates. What's the problem?

Paragon RETS returns date and timestamp values in UTC (Universal Time Coordinated).

Why is my search only returning some of the listings when I know there are more?

Because of the size of the listing database, queries that are too general could return unmanageably large result sets. If no limits were placed on the number of records it could greatly impact to performance. To do this we have we have RETS User Profile configurable limits on the number of records that can be returned in a single transaction for any given resource and class. These limits should be sufficient for most users.

One way around this is to use the LIMIT=NONE parameter in the search request. Unfortunately, we only suspend the limits all of the fields being returned in the Select parameter are ones flagged in the METADATA-TABLE as InKeyIndex=1. Otherwise the normal limits apply.

The InKeyIndex flag is only set for a selection of critical fields which can vary by resource but usually contains the keyField and modification date fields.

Examples of InKeyIndex fields would be ones like the following:

- Listing ID
- Modification Timestamp
- Photo Count
- Photo Update Date
- User ID
- User Status
- Office ID
- Office Status

The actual field names can be found in your metadata. (See Basics of Using Paragon Metadata, above.)

I need to pull more results than my profile allows. Is there any way around this limit?

To download more than the limited result set value, you can use the **Offset** and **Limit** parameters described in Sample Search Transaction to incrementally step through the available records. Detailed instructions are as follows:

1. First, run a search on your desired query with the Count parameter set to 2, e.g., to return the actual number of records:

   ```
   <search url>?Class=ResidentialProperty&SearchType=PropertyQueryType=DMQL2&
   Format=COMPACT&StandardNames=1&Query=(<your query here>)&Limit=2500&Offset=1
   ```

© 2024 Intercontinental Exchange, Inc.

This will return a count of the records that match your query without trying to return the actual record set. If the count is less than your limit, you do not need to use Offset, but can simply run your search.

[NOTE: More advanced users may wish to skip this step, instead looking for the <MAXROWS> tag at the end of a search to determine whether all rows have been sent. More details can be found in the RETS Specifications.

2. Run the desired query as above, but eliminate the Count parameter, instead setting Limit to a valueless than or equal to your limit, for example Limit=2500 and Offset=1.

   Note: Using Offset=1 on your first search request can be important because on some servers, it notifies the server that you are using offset functionality and ensures the records are ordered by keyField to which minimize the possibility duplicate records in the combined result set.

   ```
   <search url>?Class=ResidentialProperty&SearchType=PropertyQueryType=DMQL2&
   Format=COMPACT&StandardNames=1&Query=(<your query here>)&Limit=2500&Offset=1
   ```

3. Run the query again with an offset of previous Limit+1 (2501 in the example above) for the next block of records, then (2*Limit)+1 for the next, and so on, until all matching records have been retrieved:

   ```
   <search url>?Class=ResidentialProperty&SearchType=PropertyQueryType=DMQL2&
   Format=COMPACT&StandardNames=1&Query=(<your query here>)&Limit=2500&Offset=2501
   <search url>?Class=ResidentialProperty&SearchType=PropertyQueryType=DMQL2&
   Format=COMPACT&StandardNames=1&Query=(<your query here>)&Limit=2500&Offset=5001
   ```

Is there another way to pull larger result sets without using the Offset parameter?

The Offset parameter is the simplest and most efficient way to get around the download limit. When offset is used, Paragon RETS server returns the records in key field order which will normally not result in duplicates.

However, if you prefer, you can implement your own strategy yourself as follows: First, perform a query that selects only the listing ID and modification timestamp fields. As noted in FAQ #2 above, there is no restriction on the number of records that may be downloaded for a query of this type. After downloading the listing IDs, issue a second set of queries using these IDs, separated by commas, to retrieve the full listing data.

Detailed Instructions:

1. Issue a RETS query to the RETS server using "ModificationTimestamp >= [Date]" in the argument andspecifying only two columns: LN (MLS Number) and ModificationTimestamp. In this case, [Date] is a point in time at which you last queried the RETS server for updates or a point in time from which you want to start over. The RETS server will return the value of these two columns for all records matching the search criteria.

   Example query string:

   ```
       <search url>?Class=ResidentialProperty
       &SearchType=Property
       &QueryType=DMQL2
       &Format=COMPACT-DECODED
       &StandardNames=1
       &Select=ListingID,ModificationTimestamp
       &Query=(ModificationTimestamp=2016-01-01T00:00:00+)
   ```

2. After the query has returned, use the ModificationTimestamp for the listing to compare against what you have in your local data store to determine what has changed. Build a list of records to that need to be downloaded to update your local store by ListingID values (see next).

© 2024 Intercontinental Exchange, Inc.

3. Issue a RETS Search to the RETS server using a batch (500 or more) comma delimited ListingID values at a time, requesting whatever fields you need returned. Repeat this process until all the records are have been returned. (Note: You can use higher numbers up to your configured record limit, but in most cases 500 or 1000 should be sufficient for optimal performance).

   Example query string:

   ```
   <search url>?Class=ResidentialProperty
   &SearchType=Property
   &QueryType=DMQL2
   &Format=COMPACT-DECODED
   &StandardNames=1
   &Query=(ListingID=123,124,125,...)
   ```

How do I download the images for a listing?

See Sample GetObject Transaction above.

If your account is allowed access you can get Photo URLs using the Media resource. Note: The media sequence is ZERO based. So the first image would be sequence=0.

Why do I see an error "Access to object not allowed [ABC1234:2]" for some of these photos?

The List Agent has marked the photo as private, which is notated in the X-Accessibility response header. The Organization you're requesting from has restricted access to Private photos. Please contact the Organization to request access.

You will not get this error when pulling photo URL's through the Media resource. Instead, the Organization will have privacy flag in a field called MED_Permissions. Vendors should honor this flag.

Is there a way to get thumbnails instead of full-resolution images?

For a thumbnail version of a photo, use the GetObject transaction, but substitute "Type=Thumbnail" for "Type=Photo".

See Sample GetObject Transaction above.

Is there a way to get Agent and Office images?

For agent images, use the GetObject transaction, but change parameters to "Resource=Agent" or "Resource=ActiveAgent" and "Type=Photo" or "Type=OfficeLogo".

Currently we only support one image of each type per agent. Also agent images are only available through GetObject and are NOT available through the Media resource.

See Sample GetObject Transaction above.

© 2024 Intercontinental Exchange, Inc.

Additional Resources

Additional Paragon RETS specific information is available here:

- Paragon Vendor Support Site: https://vendorsupport.paragonrels.com

© 2024 Intercontinental Exchange, Inc.