RETS Web API v1.0.1

Chapters

Additional Sections

List of Tables & Figures

Figures

Samples

Appendix 2 - Query Examples

Section 1 - Proposal

1.1 Purpose

The RESO transport workgroup has been tasked with recommending a new industry-wide standard for real-time access to real estate data {directly from Web and Mobile applications}. The goal of this new standard is to provide a more open approach to data access using widely-adopted technology standards in use across industries, including the real estate industry. Specifically, the approach focuses on the use of the REST (REpresentational State Transfer) architectural approach documented by Roy Thomas Fielding and adopted by tens of thousands of developers worldwide.

The goal driving the move toward a RESTful standard for the real estate industry is to encourage and promote access to real estate information directly from Web, mobile, social and other HTTP-based applications. Using a RESTful transport will enable web applications to directly interact with RESO enabled data services. (*note: more information on RESTful can be found here*)

This workgroup sought to find an approach that does not deviate from either the solid foundations already employed from past RESO accomplishments or the existing technology standards that set out to solve similar problems for other industries.

The goals of this group were to:

1. Honor existing data service capabilities from RETS 1[1].x
2. Adopt existing standard technologies in use across industries
3. Leverage existing production-ready software toolkits

As such the group proposes the use of an existing standard that was designed specifically for data transport. The standard, Open Data Protocol or "OData" (http://www.odata.org/) serves as a set of fundamental building blocks for what the group is proposing.

The group chose OData for the following reasons:

- Well-established and robustly documented existing standard.
- Significant community adoption including "Open Government Data Initiative."
- Well-defined functionality supports most significant RESO use cases.
- Existing open source technology implementations to support community adoption.

- Extensibility to handle specific use cases as needed.

As a standards body, we will follow the OData standard and will extend, where needed, to fulfill our industry's needs. We will not, however, deviate from the RESTful principles, standard capabilities or query syntax that is inherent to the OData standard.

OData Overview[2]

The Open Data Protocol (OData) is an application-level protocol for interacting with data via RESTful web services. The protocol supports the description of data models and the editing and querying of data according to those models. It provides facilities for:

- Metadata: a machine-readable description of the data model exposed by a particular data provider.
- Data: sets of data entities and the relationships between them.
- Querying: requesting that the service perform a set of filtering and other transformations to its data, then return the results.
- Editing: creating, updating, and deleting data.
- Operations: invoking custom logic.
- Vocabularies: attaching custom semantics.

The OData Protocol provides a uniform way to describe both the data and the data model. This improves semantic interoperability between systems and allows an ecosystem to emerge.

Towards that end, the OData Protocol follows these design principles:

- Prefer mechanisms that work on a variety of data stores. In particular, do not assume a relational data model.
- Extensibility is important. Services should be able to support extended functionality without breaking clients unaware of those extensions.
- Follow REST principles unless there is a good and specific reason not to.
- OData should build incrementally. A very basic, compatible service should be easy to build, with additional work necessary only to support additional capabilities.
- Keep it simple. Address the common cases and provide extensibility where necessary.

Further details pertaining to OData may be found at the below link:

OData V3 - http://www.odata.org/documentation/odata-version-3-0/

---

[1] RETS 1x is a legacy protocol produced by RESO and still in use today

[2] Source: © Copyright OASIS Open 2013.

1.2 Scope

The initial scope of this standard is to support read only searching of data resources that have been defined by the Data Dictionary Workgroup and other RESO data providers.

Explicitly in scope in this initial release will be:

1. Metadata Representation
2. Read Access / Standard Search
3. Geospatial Search
4. Hypermedia Representation

Explicitly out of scope in this initial release will be:

1. Create, Update, Delete resource content
2. A Data Replication Framework
3. Requesting Binary Media Resources
4. Updating Binary Media Resources
5. Saved Searches and Resources

Explicitly out of scope for the transport specification will be:

1. Authentication and Authorization
    a. Please See the "RETS Web API Security" document.
2. The underlying Data Dictionary and Resource definitions
    a. Please see the latest "Data Dictionary" files for details.

1.3 Approach

The RESO OData Transport standardizes access to Real Estate data over the Internet using a Representational State Transfer (REST) style interface. Compatible RESO OData Transport client and server applications MUST be implemented according to the OData V3 standard specification. All further references to OData in this document refer to the OData V3 standard. Compatible server and client applications MUST send or receive data in JSON or ATOM/XML format. In keeping with OData both the client and server applications will use the standard HTTP methods GET and POST to perform the operations outlined by this document.

A compatible server takes action based on the HTTP method called by a compatible client. The following HTTP methods must be honored as follows.

- GET - gets the requested item or collection data in JSON or ATOM/XML format.
- POST - used in conjunction with X-HTTP-Method-Override header.

For POST Usage: While this is a non-standard approach, HTTP request header is the "de facto" standard for instructing a server to override the method requested with the value supplied in the header (if supported).The approach is being taken to fully leverage the existing capabilities within OData for our industry's needs.

Where possible, we will leverage existing syntax that may be augmented. Where this is not possible, new extensions will be created and may be proposed back to the OData standards group for inclusion in future releases.

In all cases, where an extension is made, a reference implementation will also be created and shared with the community.

The initial focus will be on HTTP GET for search.

Required output will be:

1. ATOM (XML)
2. JSON

The response format is defined by use of Content Negotiation (*http://www.w3.org/Protocols/rfc2616/rfc2616-sec12.html*) and the Accept Header may be used to define the desired data output. If Accept: */* is used the default response format is ATOM - XML.

Section 2 - Specification

This specification outlines the current, minimum set of functionality required by RESO as a subset of the OData V3 specification.

2.1 Terminology

Table 1 - Terminology

Term	Definition
REST	Representational State Transfer. For more information see: http://en.wikipedia.org/wiki/Representational_state_transfer
Resource	In a RESTful API a resource is an object with a type, associated data, relationships to other resources, and a set of methods that may operate on it.
RESO Data Dictionary	A uniform set of field names and data type conventions that set a baseline across the real estate industry for how real estate data will be defined. See http://www.reso.org/data-dictionary.
Standard Resource	A data source or collection of data that is represented using the definitions found in the RESO Data Dictionary.
Custom Resource	A data source or collection of data that is represented using the something other than the RESO Data Dictionary. This may also be localized data such as language localization.
Metadata	Descriptive information about a data set, object or resource that helps a recipient understand how the data is formatted.
Payload	For purposes of the RESO community the term "payload" is synonymous with the OData term "resource." A resource refers to the object(s) you wish to retrieve in response from the server.

Schema	A way of logically defining, grouping, organizing and structuring information so it may be understood by different systems.
MUST	This word or the adjective "required" means that the item is an absolute requirement of the specification. A feature that the specification states MUST be implemented is required in an implementation in order to be considered compliant. If the data is available in the system AND the data is presented for search then it MUST be implemented in the manner described in the specification.
SHOULD	This word or the adjective "recommended" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course. A feature that the specification states SHOULD be implemented is treated for compliance purposes as a feature that may be implemented.
MAY	This word or the adjective "optional" means that this item is truly optional. A feature that the specification states MAY be implemented need not be implemented in order to be considered compliant. However, if it is implemented, the feature MUST be implemented in accordance with the specification.
Out of Scope	This statement means that the specific topic has not been addressed in the current specification but may be addressed in future versions.
N/A	This term means “not applicable” to the scope of this standard and will not be addressed by this standard specification.

2.2 HTTP Protocol

A compatible server implementation MUST use either HTTP or HTTPS as the protocol declared by the server URL. The version MUST be HTTP 1.0 or above.

"OData protocol, which is based on the AtomPub [RFC5023] specification, which, in turn, relies on HTTP [RFC2616]. Either HTTP 1.1 or HTTP 1.0 may be used with the OData protocol. The OData protocol uses HTTP headers that are defined in the HTTP specification, but are not referenced in the AtomPub specification.

Information contained in this section is pulled from section 1.4 of the OData Specification for versions 1, 2 and 3.

2.2.1 Version Header

```
RESO-OData-Version: [Version]

[Version] = MAJOR.MINOR
```

The version header is used by the server to communicate the currently supported version of the specification.

- If Client Requests No version: Server MUST return the current supported version
- If Client Requests the Current version: Server MUST return the current version
- If Client Requests an Older version that the server still supports: Server MUST return requested version
- If Client Requests an Older version than the server supports: Server MUST return HTTP 400 Bad Request
- If Client Requests a Newer version than the server supports: Server MUST return HTTP 400 Bad Request

Please see 2.5 Response Message Bodies for details on expected responses.

2.2.2 X-HTTP-Method-Override Header

Servers SHOULD accept X-HTTP-Method-Override so that clients may use POST in place of GET when the request string is too long for the client or server implementation.

2.3 URL Formatting

The RESO OData Transport defines a few standardized URL formatting requirements for ease of use and application interoperability. These

requirements are designed to permit standards-compliant applications and servers to interoperate in a pluggable manner requiring minimal configuration.

2.3.1 Hostname

The hostname of the URL is arbitrary and no naming convention in required. For the purposes of this standard the following example protocol and hostname will be used for clarity.

```
http://odata.reso.org
```

2.3.2 URI Stem

The RESO OData Transport recommends the following URI stem naming convention to simplify client application interoperability.

1. Service = http://odata.reso.org/RESO/OData/
2. Resource = http://odata.reso.org/RESO/OData/DataSystems
3. Resource Entity By ID = http://odata.reso.org/RESO/OData/DataSystems('RESO_MLS')

The /RESO section denotes that a RESO standardized interface is provided.

The /OData section denotes a RESO OData Transport compliant interface is provided.

2.3.3 Data Systems Endpoint

There will be a top level URI to expose the “Data Systems” endpoint. The Data Systems end point will allow a user to inspect the Data Systems available on the service including the following details:

1. Specification Version - The version of the Transport Specification supported.
2. Data System Endpoint - The URI identifying the location of the service for that data system.
3. Available Resources - The list of available Standard or Custom Resources available in the data system.
4. Localizations of Resources - The list of available “localized” or “custom” resources that may not conform to the RESO Data Dictionary.

```
http://odata.reso.org/RESO/OData/DataSystems
```

This methodology permits a server to expose multiple systems as deemed appropriate. This may be used to describe a catalog of Data Systems content which a client may use.

Pleas see response references in Appendix 2 - DataSystem XML Schema.

2.3.4 Resource Endpoint

The [Resource] section denotes the standardized or custom resource provided. The RESO Data Dictionary defines all standardized resources. The server defines all custom resources. All resources are defined using XML Schemas.

```
http://odata.reso.org/RESO/OData/[Resource]
```

2.3.5 Metadata Endpoint

The metadata endpoint provides metadata for the Resources associated with a specific service endpoint URI.

```
http://odata.reso.org/RESO/OData/$metadata
```

The metadata returned by a compatible server must adhere to OData Metadata requirements. This metadata endpoint will return the metadata for all Resources provided by the service endpoint.

Note: The .svc extension is omitted. Though this is common for .Net OData implementations it is not for other platforms so we do not require .svc here.

2.4 Search

Section 10 of the OData Version 3.0 specification provides full details about OData service requests and query support.

2.4.1 Search by Unique ID

2.4.2 Query Support

2.4.3 Basic Search Filters ($filter)

2.4.4 Logical operators

2.4.5 Equality operators

2.4.6 Data Type - Operator Compatibility Matrix

2.4.7 String functions

2.4.8 Enumeration functions

2.4.9 Geospatial Search

2.4.1 Search by Unique ID

Accessing a single item in a provided resource must adhere to the OData standard taking the following form:

```
http://odata.reso.org/RESO/OData/[Resource]('[ID]')
```

The [ID] must contain content conforming to the resource key as described by the resource metadata. The [ID] section is the unique ID of the requested item.

Note: You may request multiple resources using the **$filter** parameter to perform a search.

2.4.2 Query Support

You can use OData queries to filter the items you get back. See Built-in Filter Operations for further details.

A client may retrieve a list of objects that match supplied search criteria. This is done using OData query parameters. The RESO OData Transport explicitly supports the following parameters.

- $select – MUST support
  Selects desired resource elements to be returned.
- $filter – MUST support
  Filters returned items according to filter criteria.
- $top – MUST support
  Designates the maximum number of matching items returned.
- $skip – MUST support
  Designates the number of matching items to omit before returning any items. When using $skip, it is expected that the first query sent to the server starts with a $skip=0 in order to allow servers wishing to implement consistent pagination an indication that they should prepare to receive multiple requests with differing $skip values and matching $filter.
- $orderby – MAY support
  Designates the field used to order items returned - A server may return an HTTP 400 Bad Request error for any request that is deemed too resource intensive.

NOTE: Field names are case sensitive when used in the $select, $filter, and $orderby parameters. Therefore you MUST respect case sensitivity defined in the resource metadata.

2.4.3 Basic Search Filters ($filter)

Although OData V3 provides a vast array of search functionality, this version of the specification only requires a compatible server to support the following data types.

1. Text string
   a. Remarks
   b. Features
   c. Area Names

2. Numeric
    a. Beds
    b. Baths
    c. Price
3. DateTime
    a. List date
    b. Sale date
    c. Open House Start
4. Boolean
    a. Waterfront
    b. Pets Allowed
5. Geospatial Point
    a. Location of a property
6. Geospatial Multi-Point
    a. Locations of a number of properties
7. Geospatial Polygon
    a. A neighborhood boundary
8. Geospatial Multi-Polygon
    a. Multiple areas drawn on a map
9. Geospatial Line
    a. A line drawn on a map
10. Enumeration (Lookup Values)
    a. Property types
    b. Features

2.4.4 Logical operators

Search for content using logically combined criteria:

And – logically ‘ands’ the criteria operand

Or – logically ‘ors’ the criteria operand

Not – logically negates the criteria operand

2.4.5 Equality operators

Search for any content that matching a provided value

Eq – Equal

Ne – Not Equal

Gt – Greater Than

Ge – Greater than or equal to

Lt – Less than

Le – Less than or equal to

2.4.6 Data Type - Operator Compatibility Matrix

Table 2 - Data Type / Operator Compatibility Matrix

Data Type	Eq	Nq	Gt	Ge	Lt	Le
Text String	Y	Y	N	N	N	N
Numeric	Y	Y	Y	Y	Y	Y
DateTime	Y	Y	Y	Y	Y	Y
Boolean	Y	Y	N	N	N	N
Geospatial Point	N	N	N	N	N	N
Geospatial Multi-Point	N	N	N	N	N	N

Geospatial Polygon	N	N	N	N	N	N
Geospatial Multi-Polygon	N	N	N	N	N	N
Geospatial Line	N	N	N	N	N	N
Enumeration	Y	Y	N	N	N	N

2.4.7 String functions

Search for content within a string using the following OData functions:

e.g. Specific text within agent remarks

- substringof

e.g. search for records that do NOT contain a specific string

- not substringof

e.g. An agent name

- endswith OR startswith

2.4.8 Enumeration functions

Search for one or more members of an enumerated value

e.g. Where one or more of the desired values are present

- Use any() to select Property records with a Room of RoomType equal to ‘Kitchen’

$filter=Property/any(Rooms: Room/Type eq 'Kitchen')

e.g. Where all of the desired values are present

- Use all() to select Property records with all Units that have BedsTotal greater than or equal to 2

$filter=Property/any(Units: Unit/BedsTotal ge 2)

(See ‘OData Section 8 Enumeration Type Constructs’)

2.4.9 Geospatial Search

Geographic search MUST be supported using the following OData V3 Geospatial Functions.

- geo.distance - Search for resources nearby
- geo.intersects - Search for resources within an area (intersection of point and area)

The following geospatial data types are supported natively based on OGC specifications:

- Point – A longitude, latitude definition of a point on earth - MUST
- Polygon – A combination of points to create an area - MUST
- Multi-polygon - A combination of points to create multiple areas - MUST
- -Multi-point - A collection of points that are not connected - SHOULD
- Line String – A collection of points and the “linear” interpretation between those points - SHOULD

Please refer to the following document for a full description of the geospatial primitives and functions supported in OData V3: http://www.odata.org/2011/10/geospatial-properties/

2.5 Response Message Bodies

2.5.1 Standard Data Resources

2.5.2 HTTP Response Codes

2.5.3 Error Message Bodies

2.5.1 Standard Data Resources

2.6.1 Standard Data Resources

The RESO OData Transport only defines one data object for use with the transport standard, the DataSystem object. The DataSystem object describes the content provided by the compliant server implementation and only serves as a discovery mechanism which a compliant client implementation may use to identify available resources. The standard DataSystem XML Schema is located in Appendix A of this document. An available resource may include but is not limited to standard data objects as defined by the RESO Data Dictionary Standard maintained outside of this document.

At least ONE of the following resources MUST be supported by any compliant server:

1. Property - a “Property” resource based on the 1.1 data dictionary and MAY support other versions of the dictionary.
2. Member - a “Member” resource based on the 1.1 data dictionary and MAY support other versions of the dictionary.
3. Office - an “Office” resource based on the 1.1 data dictionary and MAY support other versions of the dictionary.
4. Media - a “Media” resource based on the 1.1 data dictionary and MAY support other versions of the dictionary.

Servers MAY support more than one version of a response and may also define additional resources to support specific use cases. For example, a server could provide a “Mobile” resource that returns a condensed list of fields to reduce the size of a response. Servers may also support “custom” or “localized” resources that may not follow the RESO Data Dictionary Standard.

2.5.2 HTTP Response Codes

A compatible server implementation MUST return a valid HTTP status code for each request indicating the status of the request when ATOM-XML is requested. If the response was not successful the server MAY include an error message in the body of the HTTP response. There is a defined response body for JSON but there is no explicit requirement in the OData standard.

See Section 2.5.3 for response details.

`Table 3 - HTTP Response Codes`

Code	Short Description	Detail
200	OK	Returned by GET method when retrieving a record or records. If no records are found an empty result set is returned.
400	Bad Request	Returned by GET method calls when the data fails validation and more detail on the error may be found in the body of the response.
403	Forbidden	Returned when the selected Authentication mechanism is not successful.
404	Not Found	Returned when a GET cannot find a resource or collection.
415	Unsupported Media	Returned when a media format requested is not supported by the system.
500	Internal Server Error	Returned when an unexpected error is encountered and more detail may be provided in the response body.
501	Not Implemented	Returned when the requested method is not available.

2.5.3 Error Message Bodies

When the client makes a request which cannot be satisfied or produces an error condition, a compliant server MUST follow the OData error handling guidelines specified by the ATOM format. Full details of this mechanism may be found in the ATOM format specification at the following URL.

http://www.odata.org/documentation/odata-v3-documentation/atom-format/#15_Errors_as_XML

The following example includes a client request and a compliant server error response for reference.

Example Client Request:

```
http://odata.reso.org/RESO/OData/Members.svc/Members?$orderby=MemberID&$top=5&$skip=5
```

Example Server Response:

```
Sample 1 - Error Message Body from Server
```

```
<error xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <code>NOSKIP</code>
  <message xml:lang="en-US">Bad Request - Resource does not support $skip parameter</message>
</error>
```

Section 3 - Security

Authentication and authorization is not covered in this document. It is expected that implementations will follow the standard recommendation from the RESO RETS Web API Security v1.0.

Following are the recommended standards, their uses, and relevant tool kits for implementation:

- **MUST** = Must support this functionality
- **SHOULD** = Should support this functionality based on proposed approach
- **MAY** = May support this functionality but no proposed approach. Roadmap item

- **HTTP Digest Authentication SHOULD** be supported, as the easiest standard to implement which addresses the first and most prevalent use case for RETS, and which can be made to serve some other use cases as well.
- **oAuth 2 SHOULD** be supported as needed to support additional use cases, especially where three-legged authorization is required.
- **SAML 2.0 Bearer Assertion Grant Type Profile for OAuth 2.0** In environments where SAML is already in use, SAML **MAY** be used as an oAuth Profile.
- **SAML** In environments where SAML is already in use, SAML **MAY** be used.

2.1 HTTP Digest and Basic Authentication

- 2.1.1 References
- 2.1.2 Toolkits
- 2.1.3 Implementation Recommendations

2.2 OAuth 2.0

- 2.2.1 Reference
- 2.2.2 Client Toolkits
- 2.2.3 Server Toolkits
- 2.2.4 Implementation Recommendations
  - 2.2.4.1 Client Password Credentials
  - 2.2.4.2 Access Token Expirations / Refresh Tokens
  - 2.2.4.3 Format and Construction of Tokens
  - 2.2.4.4 Redirect_uri Enforcement

2.3 SAML 2.0 Bearer Assertion Grant Type Profile for OAuth 2.0

2.4 SAML

Section 4 - Authors

Author	Company
Scott Petronis	Onboard Informatics

Matthew McGuire	CoreLogic
Sergio Del Rio	Templates for Business, Inc.
Fred Larsen	UtahRealEstate.com
James McDaniel	UtahRealEstate.com
Robert Gottesman	RESO

RESO Transport Workgroup

Section 5 - References

Table 4 - Document References

Description	Link
REST	http://en.wikipedia.org/wiki/Representational_state_transfer
Open Data Protocol or “OData”	http://www.odata.org/
OData V3	http://www.odata.org/documentation/odata-v3-documentation/
Geospatial Support in OData V3	http://www.odata.org/2011/10/geospatial-properties/
Filter Operations OData V3	http://www.odata.org/documentation/odata-v3-documentation/odata-core/#10231_The_filter_System_Query_Option
Enumeration Type Constructs OData V3	http://www.odata.org/documentation/odata-v3-documentation/common-schema-definition-language-csdl/#8_Enumeration_Type_Constructs
URI Conventions OData V3	http://www.odata.org/documentation/odata-v3-documentation/url-conventions/
RFC2616 HTTP Protocol 1.1	http://www.ietf.org/rfc/rfc2616.txt

Section 6 - List of Tables & Figures

Section 6 - List of Tables & Figures

Tables

Samples

Figures

Section 7 - Revision List

N/A

Section 8 - Appendices

Appendix 1 - Use Cases

MUST = Must support this functionality**.**
SHOULD = Should support this functionality based on proposed approach.
MAY = May support this functionality but no proposed approach. Roadmap item.
N/A = Not available in first release and no proposed approach. May be a roadmap item.

Table 5 - Use Cases

UC ID#	Category	Use Cases / Functionality	Example	Required
1	001 - Listing Search	Listing search by geography (name)	Akron, Ohio	MUST
2	001 - Listing Search	Listing search by point + radius	(long, lat) 20 unit = miles	MUST
3	001 - Listing Search	Listing search by boundary	polygon, multi-polygon	MUST
4	001 - Listing Search	Listing search by address + radius	123 Main St. Akron Ohio 20 unit=miles	MUST
5	001 - Listing Search	Listing search by specific address	123 Main St. Akron Ohio	MUST
6	001 - Listing Search	Listing search by specific street	Main St. Akron Ohio	MUST
7	001 - Listing Search	Listing search by map bounds	Upper left, lower right, what falls within	MUST
8	003 - Other Search	Retrieve listing details by ID	Give me the details of this specific listing	MUST
9	005 - Group Search	Get count of listings by geography (name)	How many listings are there in Chicago, IL (ZIP Code, County, Neighborhood, etc.)	MUST
10	010 - Authentication	Authentication	Give me access to the API	N/A

11	010 - Authentication	Authorization	What data do I have access to?	N/A
12	010 - Authentication	Authorization	What capabilities do I have access to?	N/A
13	010 - Authentication	Authorization	Give me access to this specific data.	N/A
14	020 - Media	Get specific media for this specific listing (agent) or record	Give me the main photo, thumbnail, etc. (urls)	MUST
15	020 - Media	Get all media for this group of listings (agent) or record	Give me the main photo urls for all these listings	MUST
16	030 - Metadata	What data dictionaries does the server support	Can I request 1.0, 1.1, 1.2, etc., names? What other resources are supported (non-RESO)?	MUST
17	030 - Metadata	MLS Rules	What are the specific rules for this MLS?	N/A
18	030 - Metadata	MLS Rules, retrieve specific info	Give me the disclaimer, logo, copyright, etc.	N/A
19	030 - Metadata	Record rules	What can I do with this specific record?	N/A
20	040 - Agent / Office	Agent or office search by geography (name)	Akron, Ohio (See lines 1-9)	MUST
21	040 - Agent / Office	Agent or office search by point + radius	(long, lat) 20 unit = miles (See lines 1-9)	SHOULD
22	040 - Agent / Office	Agent or office search by boundary	polygon, multi-polygon (See lines 1-9)	SHOULD
23	040 - Agent / Office	Agent or office search by address + radius	123 Main St. Akron Ohio 20 unit=miles (See lines 1-9)	SHOULD
24	040 - Agent / Office	Agent or office search by specific address	123 Main St. Akron Ohio (See lines 1-9)	SHOULD
25	040 - Agent / Office	Agent or office search by geography (name)	Akron, Ohio (See lines 1-9)	MUST
26	040 - Agent / Office	Agent or office search by map bounds	Upper left, lower right, what falls within (See lines 1-9)	SHOULD
27	040 - Agent / Office	Retrieve agent or office details by ID	Give me the details of this specific agent or office (See lines 1-9)	MUST
28	040 - Agent / Office	Get count of agents/offices by geography (name)	How many agents/offices are there in Chicago, IL (See lines 1-9)	MAY
29	050 - Open House	Open house search by date range and geography (name)	Akron, Ohio (See lines 1-9)	MUST

30	050 - Open House	Open house search by date range and point + radius	(long, lat) 20 unit = miles (See lines 1-9)	SHOULD
31	050 - Open House	Open house search by date range and boundary	polygon, multi-polygon (See lines 1-9)	SHOULD
32	050 - Open House	Open house search by date range and address + radius	123 Main St. Akron Ohio 20 unit=miles (See lines 1-9)	SHOULD
33	050 - Open House	Open house search by date range and specific address	123 Main St. Akron Ohio (See lines 1-9)	SHOULD
34	050 - Open House	Open house search by date range and specific street	Main St. Akron Ohio (See lines 1-9)	MAY
35	050 - Open House	Open house search by date range and map bounds	Upper left, lower right, what falls within (See lines 1-9)	SHOULD
36	050 - Open House	Retrieve open house details by date range and ID	Give me the details of this specific open house (See lines 1-9)	MUST
37	050 - Open House	Get count of open houses by date range and geography (name)	How many open houses are there in Chicago, IL (See lines 1-9)	N/A
38	060 - Statistics	Search by Statistics	Count of listings with price reductions between 5-10% in the last 30 days in specified zip codes	Out of Scope
39	080 - System	Manage Pagination	Identify the number of records per page and the specific page they would like to get back.	MUST
40	080 - System	Retrieve system capabilities, metadata	Query system to determine what types of resources, search capabilities, record limits and other constraints there may be.	MUST
41	001 - Listing Search	Simple result sortation	Allow the user to select the desired sort based on a single field (e.g. <field> ascending or descending)	MUST
42	001 - Listing Search	Advanced sort	Allow the user to apply multiple sort rules on multiple fields (e.g. sort by this, then by this)	MUST
43	001 - Listing Search	Preferential sort	Bring "my" listings to the top then sort the rest by the simple or advanced sort criteria	Out of scope
44	001 - Listing Search	Search by multiple boundaries	Bring back listings that are within any of the provided boundaries.	MUST

45	001 - Listing Search	Search in boundary intersection	Bring back listings that are within the boundary created by the intersection of two or more boundaries	MUST
46	001 - Listing Search	Saved search	"Push" content to the user based on pre-selected "search" criteria	Out of scope
47	001 - Listing Search	Alerts	Alert a "subscriber" when a listing becomes available in a specific area.	Out of scope
48	001 - Listing Search	Exclude listings with specific attributes	I don't want short sales or beach front	MUST
49	070 --Resource	Request Just IDs (Keys)	I only want to bring back the IDs of the records matching my request.	MUST
50	070 --Resource	Request Defined Resource	I want to bring back a specific resource (e.g., Full IDX, Mobile, VOW, Syndication, etc.) for the records matching my request.	MUST
51	070 - Resource	Request Specific Fields	I want to bring back only the specific fields I indicate for the records matching my request.	MUST
52	011 - Edit	Modify specific listing attributes	As an agent I want to modify specific listing attributes from my mobile device.	Out of scope
53	020 - Media	Retrieve additional documents pertaining to a listing or other record.	As an agent I want to retrieve documents such as disclosures, HOA minutes and other related documents pertaining to a listing. (This is another type of media)	MUST
54	001 - Listing Search	Keep my local database synchronized (replication) against a remote data store via an API	As an application developer I want to be able to request updates to my local data from one or more MLSs	Out of scope
55	001 - Listing Search	Aggregate data from multiple sources for local storage	As an application developer I want to be able to request updates to my local data from one or more MLSs and keep track of source details	Out of scope

Appendix 2 - Query Examples

This appendix provides a set of example queries using OData V3 and the specific RESO resources discussed in this document. This is intended to highlight various common use cases, not to describe all the possible queries that may be executed.

1 - How do I get system data?

2 - Get the DataSystem Service URI

3 - Request the list of Data Systems

4 - Select a single data system

5 - How do I look at the metadata for a specific service?

6 - How do I retrieve data using this metadata?

7 - Retrieve metadata for a resource

8 - Get a single Property

9 - Change the response to JSON

10 - Select specific field values [52]

11 - Filter by field value

12 - Filter by multiple field values

13 - Get the first five Members

14 - Get the second five Members

15 - Get the top ten Residential properties within 1 mile of a specific point ordered by distance [2]

16 - Get all the properties with a price range of $250k to $500k within a specific area drawn on map (polygon) [3]

17 - Get all the properties with a price range of $250k to $500k within the map on the screen (polygon) [3]

18 - Get all properties with price range of $250k to $500k within a complex drawn area on map (multi-polygon) [3]

19 - Get all the Residential properties within a half mile of a specific road (linestring) [2]

20 - Request only IDs [50]

21 - Get all the properties with a listing price less than $300K

22 - Get all the properties with a listing price greater than $300K

23 - Get all the properties with a listing price of $300K

24 - Query using boolean to find all properties that are short sales

25 - Combine multiple criteria in a search

26 - Get records back in a certain order

27 - Get a count of records

28 - Get all members whose first name starts with ‘Joh’

29 - Get all members whose last name ends with ‘ith’

30 - Get all members whose last name contains the string ‘ohns’

31 - Get all members whose first name is ‘James’ or ‘Adam’ and who are active

32 - Get all properties that were listed in the year 2013

33 - Get all properties that were listed in May of 2013

34 - Get records by enumeration (lookup) values

35 - Get records by values in a contained collection

1 - How do I get system data?

One of the first interactions anyone will have with the server will be to retrieve the system level data in order to understand what is supported on the server. This section describes that interaction.

2 - Get the DataSystem Service URI

```
http://odata.reso.org/RESO/OData/DataSystem.svc
```

3 - Request the list of Data Systems

```
http://odata.reso.org/RESO/OData/DataSystem.svc/DataSystem
```

Each Data System provided by the service will be listed as <entry> items. The client may select a single DataSystem using the ID of the desired DataSystem.

4 - Select a single data system

```
http://odata.reso.org/RESO/OData/DataSystem.svc/DataSystem('RESO_MLS')
```

The client then selects a Data System and underlying Resource to use. The resulting XML for this is verbose so only the relevant parts of the response are shown here.

```
Sample 5 - Select a single data system
```

```
<entry>
<id>http://odata.reso.org/DataSystem.svc/DataSystem('RESO_MLS')</id>
<category term="RESO.OData.Transport.DataSystem"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
<link rel="edit" title="DataSystem" href="DataSystem('RESO_MLS')" />
<title />
<updated>2013-11-22T19:13:50Z</updated>
<author>
<name />
</author>
<content type="application/xml">
 <m:properties>
 <d:Name>RESO_MLS</d:Name>
 <d:ServiceURI>http://odata.reso.org/DataSystem.svc/</d:ServiceURI>
 <d:DateTimeStamp m:type="Edm.DateTime">2013-11-22T14:13:50.3810781-05:00</d:DateTimeStamp>
 <d:TransportVersion>0.9</d:TransportVersion>
 <d:Resources m:type="Collection(RESO.OData.Transport.Resource)">
 <d:element>
  <d:Name>Property</d:Name>
  <d:ServiceURI>http://odata.reso.org/Properties.svc</d:ServiceURI>
  <d:Description>RESO Standard Property Resource</d:Description>
  <d:DateTimeStamp m:type="Edm.DateTime">2013-11-22T14:13:50.3810781-05:00</d:DateTimeStamp>
  <d:TimeZoneOffset m:type="Edm.Int32">-5</d:TimeZoneOffset>
```

The same content is available in JSON format as well and the above example will look like the following one in JSON format.

Sample 6 - Select a single data system w/JSON

```
{
"odata.metadata": "http://odata.reso.org/DataSystem.svc/$metadata#DataSystem/@Element",
"Name": "RESO_MLS",
"ServiceURI": "http://odata.reso.org/DataSystem.svc/",
"DateTimeStamp": "2013-11-22T17:41:34.8131432-05:00",
"TransportVersion": "0.9",
"Resources": [{
"Name": "Property",
"ServiceURI": "http://odata.reso.org/Properties.svc",
"Description": "RESO Standard Property Resource",
"DateTimeStamp": "2013-11-22T17:41:34.8131432-05:00",
"TimeZoneOffset": -5,
...etc...
```

5 - How do I look at the metadata for a specific service?

The client will select a resource within the system using the Name property of the Resource. In the following example the client has selected the Property Resource and referred to the ServiceURI to find the service that provides Property data.

Sample 7 - How do I look at the metadata for a specific service? (URI endpoint)

```
<d:Name>Property</d:Name>
        <d:UID>1</d:UID>
        <d:ServiceURI>http://odata.reso.org/Properties.svc</d:ServiceURI>
        <d:Description>RESO Standard Property Resource</d:Description>
        <d:DateTimeStamp
                m:type="Edm.DateTime">2013-11-14T16:02:14.828328-05:00</d:DateTimeStamp>
        <d:TimeZoneOffset m:type="Edm.Int32">-5</d:TimeZoneOffset>
```

Using the ServiceURI value (which must be a valid URI) the client can request metadata for the desired Resource.

6 - How do I retrieve data using this metadata?

From this point a client may begin interacting with the server to search for a retrieve the desired data. The following section highlights various common types of searches that may be conducted against any fully functional RESO OData Transport Server.

Now that the client has the Property service URI and metadata, it may request Property records from that service by key or search criteria.

This first example covers retrieving a record by key.

7 - Retrieve metadata for a resource

```
http://odata.reso.org/RESO/OData/Properties.svc/$metadata
```

This will return the metadata for the content provided. In this case the example points to a Data Dictionary 1.1 implementation. The example metadata is truncated for brevity.

Sample 8 - Retrieve metadata for a resource

```
<edmx:Edmx Version="1.0" xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx">
 <edmx:DataServices m:DataServiceVersion="3.0" m:MaxDataServiceVersion="3.0"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <Schema Namespace="CoreLogic.DataService.RESO"
   xmlns="http://schemas.microsoft.com/ado/2009/11/edm">
  <EntityType Name="Property">
  <Key>
   <PropertyRef Name="ID" />
  </Key>
    <Property Name="ID" Type="Edm.String" Nullable="false" />
    <Property Name="AboveGradeFinishedArea" Type="Edm.Single" />
    <Property Name="AboveGradeFinishedAreaSpecified" Type="Edm.Boolean" Nullable="false" />
    <Property Name="AboveGradeFinishedAreaSource" Type="Edm.String" />
    <Property Name="AboveGradeFinishedAreaUnits" Type="Edm.String" />
    <Property Name="AccessibilityFeatures" Type="Collection(Edm.String)" Nullable="false" />
    <Property Name="AdditionalParcelsDescription" Type="Edm.String" />
    <Property Name="AdditionalParcelsYN" Type="Edm.String" />
    <Property Name="ApprovalStatus" Type="Edm.String" />
```

8 - Get a single Property

`http://odata.reso.org/RESO/OData/Properties.svc/Properties('ListingId3')`

By default this will return an ATOM XML format for the response. Here is a truncated example response for the request above.

Sample 9 - Get Single Property return ATOM XML

```
<?xml version="1.0" encoding="utf-8"?>
<entry xml:base="http://odata.reso.org/Properties.svc/"
xmlns="http://www.w3.org/2005/Atom"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:georss="http://www.georss.org/georss" xmlns:gml="http://www.opengis.net/gml">
  <id>http://odata.reso.org/Properties.svc/Properties('ListingId3')</id>
  <category term="CoreLogic.DataService.RESO.Property"
   scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
<link rel="edit" title="Property" href="Properties('ListingId3')" />
<title />
 <updated>2013-11-14T21:27:24Z</updated>
<author>
    <name />
</author>
<content type="application/xml">
  <m:properties>
    <d:ID>ListingId3</d:ID>
    <d:AboveGradeFinishedArea m:type="Edm.Single">3</d:AboveGradeFinishedArea>
    <d:AboveGradeFinishedAreaSource>AboveGradeFinishedAreaSource3
    </d:AboveGradeFinishedAreaSource>
    <d:AboveGradeFinishedAreaUnits>AboveGradeFinishedAreaUnits3
    </d:AboveGradeFinishedAreaUnits>
...etc...
```

The client may change this to JSON as well.

9 - Change the response to JSON

`http://odata.reso.org/RESO/OData/Properties.svc/Properties('ListingId3')?$format=json`

This will return the following example result again truncated for brevity.

`Sample 10 - Change the response to JSON`

```
{
    "odata.metadata": "http://odata.reso.org/Properties.svc/$metadata#Properties/@Element",
    "ID": "ListingId3",
    "AboveGradeFinishedArea": 3,
    "AboveGradeFinishedAreaSpecified": false,
    "AboveGradeFinishedAreaSource": "AboveGradeFinishedAreaSource3",
    "AboveGradeFinishedAreaUnits": "AboveGradeFinishedAreaUnits3",
    "AccessibilityFeatures":
        ["AccessibilityFeatures1",
        "AccessibilityFeatures2",
        "AccessibilityFeatures3"],
    "AdditionalParcelsDescription": "AdditionalParcelsDescription3",
    "AdditionalParcelsYN": "AdditionalParcelsYN3",
    "ApprovalStatus": "ApprovalStatus3",
    "ArchitecturalStyle": "ArchitecturalStyle3",
...etc...
```


These additional examples show how to use various select, filter and function options provided by the RESO OData Transport Server.

10 - Select specific field values [52]

`http://odata.reso.org/RESO/OData/Members.svc/Members?$select=MemberLastName,MemberFirstName,MemberID`

Note: *All names in the $select option are case-sensitive to match the names of elements provided by the resource.*

11 - Filter by field value

`http://odata.reso.org/RESO/OData/Members.svc/Members?$filter=(MemberLastName eq 'Smith')`

Note: All names in the $filter option are case-sensitive to match the names of elements provided by the resource.

12 - Filter by multiple field values

`http://odata.reso.org/RESO/OData/Members.svc/Members?$filter=(MemberFirstName eq 'Joe' and MemberLastName eq 'Smith').`

Note: *Query strings MUST be URL encoded where appropriate by a compliant client.*

13 - Get the first five Members

`http://odata.reso.org/RESO/OData/Members.svc/Members?$orderby=MemberID&$top=5`

14 - Get the second five Members

`http://odata.reso.org/RESO/OData/Members.svc/Members?$orderby=MemberID&$top=5&$skip=5`

Note: *The implementation of $top and $orderby is defined by the server and may restrict what values may be used in either option. A compliant client SHOULD use the $orderby query to sustain consistency between requests, however a compliant server is not required to guarantee consistent results between requests.*

15 - Get the top ten Residential properties within 1 mile of a specific point ordered by

distance [2]

```
http://odata.reso.org/RESO/OData/Properties.svc/Properties?$filter=/PropertyType/Name eq "Residential" and geo.distance(Location,POINT(-127.89 45.23)) lt 1&$orderby=geo.distance(Location, POINT(-127.89 45.23))&$top=10
```

16 - Get all the properties with a price range of $250k to $500k within a specific area drawn on map (polygon) [3]

```
http://odata.reso.org/RESO/OData/Properties.svc/Properties?$filter=ListPrice gt 250000 and ListPrice lt 500000 and geo.intersects(Location,POLYGON((-127.01 45.50,-127.00 45.49,-127.01 45.49,-127.00 45.50)))
```

17 - Get all the properties with a price range of $250k to $500k within the map on the screen (polygon) [3]

```
http://odata.reso.org/RESO/OData/Properties.svc/Properties?$filter=ListPrice gt 250000 and ListPrice lt 500000 and geo.intersects(Location,POLYGON((-127.02 45.08,-127.02 45.38,-127.32 45.38,-127.32 45.08,-127.02 45.08)))
```

18 - Get all properties with price range of $250k to $500k within a complex drawn area on map (multi-polygon) [3]

```
http://odata.reso.org/RESO/OData/Properties.svc/Properties?$filter=ListPrice gt 250000 and ListPrice lt 500000 and geo.intersects(Location,MULTIPOLYGON(((-127.02 45.08,-127.023 45.38,-127.32 45.38,-127.32 45.08,-127.02 45.08)),((-127.12 45.18,-127.12 45.28,-127.22 45.28,-127.22 45.18,-127.12 45.28))))
```

19 - Get all the Residential properties within a half mile of a specific road (linestring) [2]

```
http://odata.reso.org/RESO/OData/Properties.svc/Properties?$filter=PropertyType/Name eq "Residential" and geo.distance(Location, LINESTRING (-118.62 34.22, -118.61 34.22, -118.61 34.21, -118.62 34.2, -118.62 34.22))lt 0.5
```

20 - Request only IDs [50]

```
http://odata.reso.org/RESO/OData/Properties.svc/Properties?$filter=ID
```

21 - Get all the properties with a listing price less than $300K

```
http://odata.reso.org/RESO/OData/Properties.svc/Properties?$filter=ListPrice lt 300000
```

22 - Get all the properties with a listing price greater than $300K

```
http://odata.reso.org/RESO/OData/Properties.svc/Properties?$filter=ListPrice gt 300000
```

23 - Get all the properties with a listing price of $300K

```
http://odata.reso.org/RESO/OData/Properties.svc/Properties?$filter=ListPrice eq 300000
```

24 - Query using boolean to find all properties that are short sales

```
http://odata.reso.org/RESO/OData/Properties?$filter=ShortSale eq true
```

25 - Combine multiple criteria in a search

```
http://odata.reso.org/RESO/OData/Properties?$filter=ListPrice gt 250000 and ListPrice lt 500000
```

26 - Get records back in a certain order

```
http://odata.reso.org/RESO/OData/Properties?$filter=ListPrice lt 300000&$orderby=ListPrice desc
```

27 - Get a count of records

```
http://odata.reso.org/RESO/OData/Properties?$filter=ListPrice lt 300000&$inlinecount=allpages
```

28 - Get all members whose first name starts with ‘Joh’

```
http://odata.reso.org/RESO/OData/Members.svc/Members?$filter=startswith(MemberFirstName, 'Joh')
```

29 - Get all members whose last name ends with ‘ith’

```
http://odata.reso.org/RESO/OData/Members.svc/Members?$filter=endswith(MemberLastName, 'ith')
```

30 - Get all members whose last name contains the string ‘ohns’

```
http://odata.reso.org/RESO/OData/Members.svc/Members?$filter=indexof(MemberLastName, 'ohns')
```

31 - Get all members whose first name is ‘James’ or ‘Adam’ and who are active

```
http://odata.reso.org/RESO/OData/Members.svc/Members?$filter=(MemberStatus eq 1 and (MemberFirstName eq
'James' or MemberFirstName eq 'Adam'))
```

32 - Get all properties that were listed in the year 2013

```
http://odata.reso.org/RESO/OData/Properies.svc/Properties?$filter=year(ListDate) eq 2013
```

33 - Get all properties that were listed in May of 2013

```
http://odata.reso.org/RESO/OData/Properies.svc/Properties?$filter=year(ListDate) eq 2013 and
month(ListDate) eq 5
```

34 - Get records by enumeration (lookup) values

An enumeration or lookup is a data field that contains one or more string values. The following examples detail how to search for records containing specific string values.

The lookup is AccessibilityFeatures and I want to search for records that have HandicapAccess as one of the items:

```
http://odata.reso.org/RESO/OData/Properies.svc/Properties?$filter=AccessibilityFeatures/any(a: a eq
'HandicapAccess')
```

The ‘a: a’ represents the content (or predicate) that is being tested in the statement where the letter ‘a’ follows the ‘:’ character. Since the AccessibilityFeatures property is a string collection this means ‘a’ is a string representing one of the values. Please note that the any() operation is declared following the AccessibilityFeatures/ because the property represents a collection of strings.

35 - Get records by values in a contained collection

A given record may contain a property that itself contains a collection of items. This is useful for situations such as Rooms and Units. The following example uses Rooms and the details apply to any collections contained within a given resource.

The Rooms property is a collection of Room elements and these are complex types which contain underlying properties. In this case the any function again uses the predicate to designate what is being inspected by the statement. Therefore if the predicate is a complex type the properties of that type become accessible to the statement. For example criteria for a Room with the Room/Type equal to ‘Living’ will look like the following:

```
http://odata.reso.org/RESO/OData/Properies.svc/Properties?$filter=Rooms/any(Room: Room/Type eq 'Living')
```

Here the ‘a: ’ from above is replaced with the more useful ‘Room: ‘ predicate. Now the next statement “Room/Type eq ‘Living’” makes more sense

as it declares criteria looking for Room properties where Type = ‘Living’. If the record has ANY Room properties with Type = ‘Living’ then the record will be returned. Please note that the any() operation is declared following the Rooms/ property so that it operates over the collection of Room properties contained by Rooms.

The all() operation uses the exact same syntax with the additional requirement that ALL items in the collection must match the criteria or it will return false.

Appendix 3 - DataSystem XML Schema

Figure 1 - DataSystem XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.reso.org/RESO/DataSystem"
    xmlns:tns="http://www.reso.org/RESO/DataSystem"
    elementFormDefault="unqualified">

<complexType name="DataSystem">
        <sequence>
            <element name="Name" type="tns:Name"></element>
         <element name="ServiceURI" type="anyURI"></element>
            <element name="DateTimeStamp" type="tns:DateTimeStamp"></element>
        <element name="TransportVersion" type="tns:TransportVersion"></element>
            <element name="Resources" type="tns:Resources"></element>
        </sequence>
    </complexType>
    <complexType name="Resources">
        <sequence>
            <element name="Resource" type="tns:Resource"
            minOccurs="1" maxOccurs="unbounded"></element>
        </sequence>
    </complexType>
    <complexType name="Resource">
        <sequence>
            <element name="Name" type="tns:Name"></element>
            <element name="ServiceURI" type="anyURI"></element>
            <element name="Description" type="tns:Description"></element>
            <element name="DateTimeStamp" type="tns:DateTimeStamp"></element>
            <element name="TimeZoneOffset" type="tns:TimeZoneOffset"></element>
            <element name="Localizations" type="tns:Localizations"></element>
        </sequence>
    </complexType>
    <complexType name="Localizations">
        <sequence>
            <element name="Localization" type="tns:Localization"
```

```
            minOccurs="0" maxOccurs="unbounded"></element>
        </sequence>
    </complexType>
    <complexType name="Localization">
        <sequence>
            <element name="Name" type="tns:Name"></element>
         <element name="ServiceURI" type="anyURI"></element>
            <element name="Description" type="tns:Description"></element>
            <element name="DateTimeStamp" type="tns:DateTimeStamp"></element>
        </sequence>
    </complexType>
    <simpleType name="Name">
        <restriction base="string"></restriction>
    </simpleType>
    <simpleType name="ServiceURI">
        <restriction base="anyURI"></restriction>
    </simpleType>
    <simpleType name="Description">
        <restriction base="string"></restriction>
    </simpleType>
    <simpleType name="DateTimeStamp">
        <restriction base="dateTime"></restriction>
    </simpleType>
    <simpleType name="TimeZoneOffset">
        <restriction base="int"></restriction>
    </simpleType>
    <simpleType name="TransportVersion">
        <restriction base="string"></restriction>
    </simpleType>
</schema>
```

The following is a sample OData XML EDMX instance of the schema for reference.

Figure 2 - OData XML EDMX instance of the schema

```
<edmx:Edmx xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx" Version="1.0">
  <edmx:DataServices xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
m:DataServiceVersion="3.0" m:MaxDataServiceVersion="3.0">
   <Schema xmlns="http://schemas.microsoft.com/ado/2009/11/edm" Namespace="RESO.OData.Transport">
    <EntityType Name="DataSystem">
   <Key>
<PropertyRef Name="ID"/>
</Key>
<Property Name="Name" Type="Edm.String"/>
<Property Name="ServiceURI" Type="Edm.String"/>
<Property Name="DateTimeStamp" Type="Edm.DateTime" Nullable="false"/>
<Property Name="TransportVersion" Type="Edm.String"/>
<Property Name="DataDictionaryVersion" Type="Edm.String"/>
<Property Name="Resources" Type="Collection(RESO.OData.Transport.Resource)" Nullable="false"/>
<Property Name="ID" Type="Edm.String" Nullable="false"/>
</EntityType>
   <ComplexType Name="Resource">
<Property Name="Name" Type="Edm.String"/>
<Property Name="ServiceURI" Type="Edm.String"/>
<Property Name="Description" Type="Edm.String"/>
<Property Name="DateTimeStamp" Type="Edm.DateTime" Nullable="false"/>
<Property Name="TimeZoneOffset" Type="Edm.Int32" Nullable="false"/>
<Property Name="Localizations" Type="Collection(RESO.OData.Transport.Localization)" Nullable="false"/>
</ComplexType>
   <ComplexType Name="Localization">
<Property Name="Name" Type="Edm.String"/>
<Property Name="ServiceURI" Type="Edm.String"/>
<Property Name="Description" Type="Edm.String"/>
<Property Name="DateTimeStamp" Type="Edm.DateTime" Nullable="false"/>
</ComplexType>
<EntityContainer Name="DataSystemDataProvider" m:IsDefaultEntityContainer="true">
<EntitySet Name="DataSystem" EntityType="RESO.OData.Transport.DataSystem"/>
</EntityContainer>
</Schema>
</edmx:DataServices>
</edmx:Edmx>
```

The following is a sample OData XML (ATOM) encapsulated instance of the schema for reference.

Figure 3 - OData XML (ATOM) encapsulated instance of the schema for reference

```
   <feed xmlns="http://www.w3.org/2005/Atom" xmlns:d="http://schemas.microsoft.com/ado/2007/08/datase
   rvices"xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:georss="http:
   //www.georss.org/georss" xmlns:gml="http://www.opengis.net/gml"xml:base="http://localhost:2099/Dat
   aSystem.svc/">

     <id>http://localhost:2099/DataSystem.svc/DataSystem</id>

     <title type="text">DataSystem</title>

     <updated>2014-04-11T15:24:00Z</updated>

     <link rel="self" title="DataSystem" href="DataSystem"/>

      <entry>

        <id>
         http://localhost:2099/DataSystem.svc/DataSystem('RESO_MLS')
       </id>

       <category term="RESO.OData.Transport.DataSystem" scheme="http://schemas.microsoft.com/ado/2007/
       08/dataservices/scheme"/>

       <link rel="edit" title="DataSystem" href="DataSystem('RESO_MLS')"/>

       <title/>

       <updated>2014-04-11T15:24:00Z</updated>
```

```
 <author>
  <name/>
</author>
 <content type="application/xml">
   <m:properties>
    <d:Name>RESO_MLS</d:Name>
    <d:ServiceURI>http://odata.reso.org/DataSystem.svc/</d:ServiceURI>
    <d:DateTimeStamp m:type="Edm.DateTime">2014-04-11T11:24:00.6508563-04:00</d:DateTimeStamp>
    <d:TransportVersion>0.9</d:TransportVersion>
    <d:DataDictionaryVersion>1.3</d:DataDictionaryVersion>
     <d:Resources m:type="Collection(RESO.OData.Transport.Resource)">
       <d:element>
        <d:Name>Property</d:Name>
        <d:ServiceURI>http://odata.reso.org/Properties.svc</d:ServiceURI>
        <d:Description>RESO Standard Property Resource</d:Description>
        <d:DateTimeStamp m:type="Edm.DateTime">2014-04-11T11:24:00.6508563-04:00</d:DateTimeStamp
        >
        <d:TimeZoneOffset m:type="Edm.Int32">-5</d:TimeZoneOffset>
         <d:Localizations m:type="Collection(RESO.OData.Transport.Localization)">
           <d:element>
            <d:Name>Single Family</d:Name>
            <d:ServiceURI>http://odata.reso.org/SingleFamily.svc</d:ServiceURI>
            <d:Description>Localized Single Family Residential Resource</d:Description>
            <d:DateTimeStamp m:type="Edm.DateTime">2014-04-11T11:24:00.6508563-04:00</d:DateTimeS
            tamp>
          </d:element>
           <d:element>
            <d:Name>Multi Family</d:Name>
            <d:ServiceURI>http://odata.reso.org/MultiFamily.svc</d:ServiceURI>
            <d:Description>Localized Multi Family Residential Resource</d:Description>
            <d:DateTimeStamp m:type="Edm.DateTime">2014-04-11T11:24:00.6508563-04:00</d:DateTimeS
            tamp>
           </d:element>
         </d:Localizations>
       </d:element>
```

```
            <d:element>
             <d:Name>Office</d:Name>
             <d:ServiceURI>http://odata.reso.org/Office.svc</d:ServiceURI>
             <d:Description>RESO Standard Office Resource</d:Description>
             <d:DateTimeStamp m:type="Edm.DateTime">2014-04-11T11:24:00.6508563-04:00</d:DateTimeStamp
             >
             <d:TimeZoneOffset m:type="Edm.Int32">-5</d:TimeZoneOffset>
             <d:Localizations m:type="Collection(RESO.OData.Transport.Localization)"/>
            </d:element>
            <d:element>
             <d:Name>Member</d:Name>
             <d:ServiceURI>http://odata.reso.org/Member.svc</d:ServiceURI>
             <d:Description>RESO Standard Member Resource</d:Description>
             <d:DateTimeStamp m:type="Edm.DateTime">2014-04-11T11:24:00.6508563-04:00</d:DateTimeStamp
             >
             <d:TimeZoneOffset m:type="Edm.Int32">-5</d:TimeZoneOffset>
             <d:Localizations m:type="Collection(RESO.OData.Transport.Localization)"/>
            </d:element>
          </d:Resources>
          <d:ID>RESO_MLS</d:ID>
        </m:properties>
      </content>
    </entry>
  </feed>
```

The following is a sample OData JSON encapsulated instance of the schema for reference.

Figure 4 - OData JSON encapsulated instance of the schema for reference

```
}
"odata.metadata": "http://localhost:2099/DataSystem.svc/$metadata#DataSystem",
"value": [{
   "Name": "RESO_MLS",
   "ServiceURI": "http://odata.reso.org/DataSystem.svc/",
   "DateTimeStamp": "2014-04-11T12:02:48.509401-04:00",
   "TransportVersion": "0.9",
   "DataDictionaryVersion": "1.3",
   "Resources": [{
      "Name": "Property",
      "ServiceURI": "http://odata.reso.org/Properties.svc",
      "Description": "RESO Standard Property Resource",
      "DateTimeStamp": "2014-04-11T12:02:48.509401-04:00",
      "TimeZoneOffset": -5,
      "Localizations": [{
            "Name": "Single Family",
            "ServiceURI": "http://odata.reso.org/SingleFamily.svc",
            "Description": "Localized Single Family Residential Resource",
            "DateTimeStamp": "2014-04-11T12:02:48.509401-04:00"
      },
      {
            "Name": "Multi Family",
            "ServiceURI": "http://odata.reso.org/MultiFamily.svc",
            "Description": "Localized Multi Family Residential Resource",
            "DateTimeStamp": "2014-04-11T12:02:48.509401-04:00"
      }]
   },
   {
      "Name": "Offfice",
      "ServiceURI": "http://odata.reso.org/Office.svc",
      "Description": "RESO Standard Office Resource",
      "DateTimeStamp": "2014-04-11T12:02:48.509401-04:00",
      "TimeZoneOffset": -5,
      "Localizations": []
   },
   {
      "Name": "Member",
      "ServiceURI": "http://odata.reso.org/Member.svc",
      "Description": "RESO Standard Member Resource",
      "DateTimeStamp": "2014-04-11T12:02:48.509401-04:00",
      "TimeZoneOffset": -5,
      "Localizations": []
   }],
   "ID": "RESO_MLS"
}]

}
```


The following is a sample OData JSON encapsulated instance of the schema for reference.

Figure 4 - OData JSON encapsulated instance of the schema for reference