Data Dictionary Testing Specification

Data Dictionary Version: 1.7.0+
Web API Core 1.0.2+
June 2020

RESO End User License Agreement (EULA)

This End User License Agreement (the “EULA”) is entered into by and between the Real Estate Standards Organization (“RESO”) and the person or entity (“End User”) that is downloading or otherwise obtaining the product associated with this EULA (“RESO Product”). This EULA governs End Users use of the RESO Product and End User agrees to the terms of this EULA by downloading or otherwise obtaining or using the RESO Product.

Purpose

The primary goal of Data Dictionary certification is interoperability. While the Web API Server specification ensures that servers can talk to each other in a uniform manner, if they are using different fields to represent the same data, it causes additional effort where mapping is concerned. This means products that need to interoperate between systems will be slow to market and complex. The point of the RESO Data Dictionary is to give data consumers and producers a common language to exchange data with.

When standards are approved for the RESO Data Dictionary, those changes are stored in a format that allows their corresponding testing rules to be generated automatically, which ensures consistency with Data Dictionary and Web API specifications. This also allows for new versions of the testing tool to be created almost immediately when Data Dictionary standards are passed.

Robust statistics are created through the use of the RESO Data Dictionary application, which are then ingested into a real-time analytics framework that lets users see marketwide statistics about resources, fields, and enumerations. This information can be used to inform decisions about standardization and data mapping between RESO certified servers.

Background

The RESO Data Dictionary testing tool ensures compliance with RESO Data Dictionary definitions of resources, fields, and enumerations. Nonstandard or "local" data elements are also allowed, provided Data Dictionary resources are used whenever present on a given server and when metadata for any additional items are in a supported and valid transport format.

Resources are top-level containers in the RESO ecosystem. Some examples are *Property*, *Member*, *Office*, *Media*, and *OpenHouse*.

Fields exist within a given resource and have name and type definitions that must be adhered to in order to be considered compliant. In the case of *Property*, examples of fields are *ListPrice*,

ModificationTimestamp, etc. Fields don't exist on their own in the metadata. They will always be contained within a top-level resource definition that MUST match RESO Standard Resource definitions when they exist.

Lookups (or enumerations) provide a set of well-known values for a given field. For example, *StandardStatus* has predefined *Active*, *Pending*, and *Sold* lookup values, to name a few.

Some lookups are open, meaning they can be extended with locally enumerated items, while others are closed, meaning they MUST only contain standard items.

Testing Framework

Data Dictionary Certification is provided by the RESO Commander.

The RESO Commander is an open source, cross-platform Java library created by RESO that uses established community libraries, such as the Apache Olingo OData Client, XML parsers, and JSON Schema Validators, to provide a testing API.

Acceptance tests define the requirements applicants are expected to meet in order to achieve certification. Data Dictionary acceptance tests are written in a high-level language (DSL) called Gherkin. This is part of a Behavior Driven Development (BDD) platform called Cucumber, which allows for the expression of testing workflows using a natural language that is intended to be accessible to business analysts and QA testers in addition to programmers.

BDD acceptance tests are automatically generated from the adopted Data Dictionary spreadsheet for each given version of the specification, and can target any version of the Data Dictionary from 1.0 onwards. The benefit of this strategy is that when a new Data Dictionary version is ratified, the tests may be generated and testing can begin right away, significantly reducing tool development time and adoption of the standard.

A command-line interface (CLI) has been provided for local testing. This provides the environment to be used for certification and self-assessment, as well as that needed to run the automated testing tools in a continuous integration and deployment (CI/CD) pipeline, on platforms such as GitHub CI, Jenkins, Travis, or CircleCI, to help prevent regressions in a RESO-certified codebase.

A graphical user interface (GUI) is also available through popular and free Integrated Development Environment (IDE) plugins for IntelliJ and Eclipse. IDEs provide an enhanced testing experience, with better informational messages and the ability to easily run and debug each test step, when needed. The availability of plugins saves significant time in testing, development, and certification. The level of community support is one of the reasons open source tools were chosen as a testing platform.

Testing Methodology

RESO Data Dictionary certification is based on adherence to:

- Resource, Field, and Lookup definitions outlined in each approved RESO Data Dictionary spreadsheet (1.7 at the time of publication).
- Transport requirements regarding authentication and OData conformance. This includes conformance with Data Dictionary to Web API data mappings outlined later in this document, as well as in RCP-031 (RESO login required).

There are several phases of Data Dictionary testing.

Configuring the Test Client

The starting point is for applicants to create a configuration file in RESOScript (XML) format which contains credentials and a server's RESO Web API endpoint. A sample RESOScript file and instructions for how to use it will be provided with the initial release of the testing tool.

Metadata Request Using RESO Standard Authentication

When testing begins, an HTTP request is made to an applicant's given service location with either OAuth2 Bearer Tokens or Client Credentials. Both of these authentication strategies allow for data consumption to be machine automated so that additional interaction from a user isn't necessary during the authentication process. As such, the RESO Data Dictionary Commander can be used for automated testing.

The metadata request is expected to function according to the OData specification in terms of request and response headers and response formats. RESO specifically uses an XML version of OData metadata, which contains an Entity Data Model (EDM) and model definitions, and is often referred to as EDMX.

OData Metadata Validation

Syntax Checking

Metadata returned from a RESO Web API server are checked for XML validity as well as validated against Entity Data Model (EDM) and EDMX definitions published by OASIS, the creators of the OData specification. If metadata are invalid for any reason, Data Dictionary testing will halt.

Semantic Checking

After metadata syntax has been validated, declared data models are checked for correctness. For example, if a given server declares support for the RESO *Property* resource, then the RESO Commander will look for an OData EntityType definition for *Property*. If the underlying data model is not found, metadata validation will fail with a diagnostic message to help users understand why a given error occurred. Once the model is found, its field and enumeration definitions will be checked for correctness as well.

Another aspect of semantic checking is ensuring that all models have keys so they can be indexed, meaning that a data request can be made to the server by key. This is a basic requirement for fetching data from a server.

RESO Certification

Several requirements must be met during Data Dictionary testing to ensure conformance with RESO Certification rules.

Conformance with the RESO Standard Data Model

In this step, tests that have been generated from a given adopted RESO Data Dictionary version are run to locate and verify resources, fields, and enumerations contained within a server's metadata. This phase of testing is designed to test that items declared in the metadata using RESO Standard Field Names are consistent with the Data Dictionary definitions for those items.

Resources

Standard Resources MUST be expressed using RESO Standard Resource Names. For instance, *Property* would be used rather than *Properties* otherwise they will not be counted. These will be verified during the certification process.

For each RESO Standard Resource found, its standard fields and lookups will be verified. Normative resource names for any Standard Resource can be found in the RESO DDWiki.

Fields

Fields have both naming and data type mapping requirements. Implementers are allowed to commingle their own fields and data types alongside RESO standard fields, but standard fields MUST match their Data Dictionary type definition mappings.

Standard Field Names

RESO Standard Fields MUST be named in accordance with the Data Dictionary definitions of those fields when present on a given server instance..

For example, if a server presents a *Property* resource and list price field data are present, they MUST be conveyed as *ListPrice*. Local fields SHOULD use the same naming conventions, when practical. There may be reasons to use nonstandard field names, such as for backwards compatibility, but they MUST pass OData validation.

Variations such as *Price* or any Data Dictionary synonym of the ListPrice field such as *AskingPrice* will fail. Various techniques are used to find potential matches with Data Dictionary definitions of resources, fields, and enumerations that don't conform to the RESO Definitions of these items. *See Additional Compliance Checking for more information*.

Additional requirements for Standard Fields are outlined in section on Data Type Mappings.

Standard Display Names

Note: RESO Standard Display Names will not be tested at the current time. They had previously been tested but were not meant to have been a standard way to convey information about *which field or lookup* is intended at the transport level. Standard Field Names and Lookup Values MUST be used for this purpose instead.

There is a proposal in progress in the RESO Data Dictionary and Transport workgroups to add further testing requirements for Display Names, which will most likely have its own Endorsement.

There is a MAY specification (RESO login required) for both `StandardName` and `MlsName` annotations that supports special characters, since OData fields and enumerations don't allow them. *Vendors may still use existing display name annotations as long as they pass the metadata validation process.*

Lookups

Standard lookup values are expected to be used to provide interoperability for RESO Standard Lookups (enumerations) during transport.

One reason lookup values have been provided is that special characters, including spaces, are not allowed by the OData specification.

One example is the *StandardStatus* field. Historically, lookups were matched against the "*Active Under Contract" display name* attached to the underlying lookup value to meet OData requirements about field and lookup names.

However, as of Data Dictionary 1.7, Display Names are not allowed for this purpose and *ActiveUnderContract* MUST be used as the underlying LookupValue instead.

Standard LookupValues are provided in the Data Dictionary 1.7 Spreadsheet.

Data Type Mappings

The following mappings apply to the RESO Data Dictionary and Web API specifications.

Data Dictionary data types shown in the following table are contained in the *SimpleDataType* column of the adopted Data Dictionary 1.7 spreadsheet, for instance those for the Property Resource.

Data Dictionary (1.6+)	Web API Core (1.0.2+)
Boolean	Edm.Bool
Collection	Related Resource Expansion, e.g. PropertyRooms or Units expanded into the Property resource. Requires $expand Endorsement.
Date	Edm.Date
Number	Edm.Decimal **OR** Edm.Double for decimal values; Edm.Int64 **OR** Edm.Int32 **OR** Edm.Int16 for integers.
String	Edm.String
String List, Single	**EITHER** Edm.EnumType **OR** Edm.String
Sting List, Multi	**EITHER** Collection(Edm.EnumType) **OR** Edm.EnumType with IsFlags="true" **OR** Collection(Edm.String)
Timestamp	Edm.DateTimeOffset

Each data type mapping has a corresponding Cucumber BDD acceptance test template that enforces the rules of a given type.

Acceptance Test Templates

Boolean

Boolean values are mapped to the Edm.Bool data type and MUST contain a literal value of "true" or "false" when returned in a payload for a given Boolean field, which is enforced by the RESO Commander. Boolean fields MAY be null as any OData field is nullable. Null values are interpreted as "false."

Sample Test

```
Scenario: AdditionalParcelsYN
  When "AdditionalParcelsYN" exists in the "Property" metadata
  Then "AdditionalParcelsYN" MUST be "Boolean" data type
```

Collection

***Note**: Collections are not yet supported in the Data Dictionary or Web API Specifications for types other than Edm.EnumType. Collections in the Data Dictionary mean expanded resources, which will not be tested until the $expand Endorsement has been created.*

Collection data types are used in the Data Dictionary to indicate *possible expansions*, which use the OData Edm.Collection type to present a collection of instances of a given type, such as Media items related to a Property record.

RESO will not be certifying this data type for related Data Dictionary Resources and until the Expand Endorsement has been created. Vendors MAY use OData $expand functionality on their servers as long as server metadata pass OData validation.

Standard Relationships have been provided in the adopted Data Dictionary spreadsheet to and reference metadata to guide vendors in the meantime. It's also worth noting that If a property definition for a Collection is nullable, it means that collection members are nullable. If there are no items in a given collection, the field would return an empty collection, but the field itself may not be null (by the OData specification).
Edit Distance Matching

Date

Date data types use the OData Edm.Date data type. Dates are expected to be in the format "yyyy-mm-dd" and should not include time zone offsets. For dates with time zone support, see Timestamp.

Sample Test

```
Scenario: AvailabilityDate
  When "AvailabilityDate" exists in the "Property" metadata
  Then "AvailabilityDate" MUST be "Date" data type
```

Number

Numbers may either be Integers or Decimals.

Integers

Numbers without Scale and Precision are treated as Integers in the Data Dictionary.

Integers are expected to be expressed using the OData Edm.Int data type and MUST NOT contain length, precision, or scale attributes.

Sample Test

```
Scenario: BathroomsFull
  When "BathroomsFull" exists in the "Property" metadata
  Then "BathroomsFull" MUST be "Integer" data type
  And the following synonyms for "BathroomsFull" MUST NOT exist in the metadata
    | FullBaths |
```

***Note*:** *Synonyms testing is shown in the last line of the above example and is discussed further in a subsequent section.*

Decimals

Decimals are expected to be Edm.Decimal or Edm.Double according to the Data Dictionary Type Mappings. They MAY contain Precision and Scale attributes, as described by the entity data model type definition, which also MAY be omitted.

If the vendor declares Precision and Scale attributes, they SHOULD match those defined by the Data Dictionary but this is not an absolute requirement. Suggested values are provided in the Data Dictionary specification but they are not mandatory at this time. This is reflected in the BDD acceptance tests.

Sample Test

```
Scenario: BuildingAreaTotal
  When "BuildingAreaTotal" exists in the "Property" metadata
  Then "BuildingAreaTotal" MUST be "Decimal" data type
  And "BuildingAreaTotal" precision SHOULD be less than or equal to the
```

```
      RESO Suggested Max Length of 14
    And "BuildingAreaTotal" scale SHOULD be less than or equal to the
      RESO Suggested Max Scale of 2
```

Note: *The Data Dictionary contains references to Length and Precision which have been found to be inaccurate with respect to standard definitions of decimal numbers. It uses Length and Precision to mean Precision and Scale, respectively. These items have been corrected in the code generation for decimal acceptance tests.*

String

String values use the OData Edm.String data type. These strings represent a sequence of UTF-8 characters. String data types MAY specify a length attribute that specifies the length of a string a given server supports. The length property is not required by OData and may be omitted.

RESO provides recommended best practices for these lengths, and applicants will be informed when their length definitions don't match the RESO definitions, but will not fail certification in these cases.

Sample Test

```
  Scenario: AboveGradeFinishedAreaSource
    When "AboveGradeFinishedAreaSource" exists in the "Property" metadata
    Then "AboveGradeFinishedAreaSource" MUST be "Single Enumeration" data type
    And "AboveGradeFinishedAreaSource" MAY contain any of the following standard
lookups
      | lookupValue | lookupDisplayName |
      | Appraiser | Appraiser |
      | Assessor | Assessor |
      | Builder | Builder |
      | Estimated | Estimated |
      | Other | Other |
      | Owner | Owner |
      | Plans | Plans |
      | PublicRecords | Public Records |
      | SeeRemarks | See Remarks |
    But "AboveGradeFinishedAreaSource" MUST NOT contain any similar lookups
```

Note: *Similar Name matching is used in cases where existing enumerations exist to ensure compliance with RESO Standards for LookupValues. This is described further in Similar Name Matching.*

String List, Single

The current RESO Web API specification uses the OData Edm.EnumType for single enumerations. As such, Data Dictionary items use this data type as well.

These items are similar to fields in that they MUST follow OData field naming conventions. Enumerations not following these conventions will fail certification in the metadata validation step.

LookupValues have been provided for standard lookups in the approved Data Dictionary spreadsheet. When present, these are shown in the example values for a the given test:

Sample Test

```
  Scenario: AboveGradeFinishedAreaSource
    When "AboveGradeFinishedAreaSource" exists in the "Property" metadata
    Then "AboveGradeFinishedAreaSource" MUST be "Single Enumeration" data type
    And "AboveGradeFinishedAreaSource" MAY contain any of the following standard
lookups
      | lookupValue | lookupDisplayName |
      | Appraiser | Appraiser |
      | Assessor | Assessor |
      | Builder | Builder |
      | Estimated | Estimated |
      | Other | Other |
      | Owner | Owner |
      | Plans | Plans |
      | PublicRecords | Public Records |
      | SeeRemarks | See Remarks |
    But "AboveGradeFinishedAreaSource" MUST NOT contain any similar lookups
```

String List, Multi

As of Web API 1.0.2 Core, there are two formats allowed for String List, Multi.

Edm.EnumType with IsFlags="true"

The Web API Server Core 1.0.2 specification outlines the use of the OData `Edm.EnumType` data type with the `IsFlags="true"` attribute set to signify that a given field supports multivalued enumerations. Applicants using this format will still be able to be certified.

Collection(Edm.EnumType)

As there are limitations to the IsFlags approach in cases where multi-select items contain more than 64 distinct values, support for Collections(Edm.EnumType) was added to the Data Dictionary Type Mappings and backported to the Web API 1.0.2 Core specification to be used instead.

The following sample test covers both representations:

Sample BDD Test

```
 Scenario: CommonWalls
    When "CommonWalls" exists in the "Property" metadata
    Then "CommonWalls" MUST be "Multiple Enumeration" data type
    And "CommonWalls" MAY contain any of the following standard lookups
      | lookupValue | lookupDisplayName |
      | EndUnit | End Unit |
      | NoCommonWalls | No Common Walls |
      | NoOneAbove | No One Above |
      | NoOneBelow | No One Below |
      | OneCommonWall | 1 Common Wall |
      | TwoOrMoreCommonWalls | 2+ Common Walls |
    But "CommonWalls" MUST NOT contain any similar lookups
```

Timestamp

Timestamps are expected to use the OData `edm:DateTimeOffset` data type. This represents an ISO 8601 compliant date that includes support for both fractional seconds and time zones. The `edm:DateTimeOffset` doesn't have any additional length, precision, or scale attributes. Data conveyed using this format is expected to match the date timestamp data type in the W3C specification.

Sample BDD Test

```
 Scenario: ModificationTimestamp
    When "ModificationTimestamp" exists in the "Property" metadata
    Then "ModificationTimestamp" MUST be "Timestamp" data type
    And the following synonyms for "ModificationTimestamp" MUST NOT exist in the
metadata
      | ModificationDateTime |
      | DateTimeModified |
      | ModDate |
      | DateMod |
      | UpdateDate |
      | UpdateTimestamp |
```

Closed Enumerations

Tests for closed or "locked" enumerations look different than those for items that are open or open with enumerations.

The most notable example is *StandardStatus* in the *Property* resource, as shown in the next sample test:

Sample BDD Test

```
Scenario: StandardStatus
  When "StandardStatus" exists in the "Property" metadata
  Then "StandardStatus" MUST be "Single Enumeration" data type
  And "StandardStatus" MUST contain only standard enumerations
  And "StandardStatus" MUST contain at least one standard enumeration
```

As shown in the test, the rules for closed enumerations are that:

- They MUST contain only standard enumerations.
- They MUST contain at least one standard enumeration.

Closed enumerations are ***not*** open to extension and tests will fail if non-standard items are found.

Lookups Resource and Location-Based Fields

After further discussion in the Transport and Certification subgroups, certain lookups will allowed to be Edm.String for String List, Single, or Collection(Edm.String) for String List, Multi, pending further discussion of Lookups, as outlined in RCP-032. Note that RCP-032 started off as being specifically for location-based lookups, such as City and CountyOrParish, and has potentially been expanded to include any Lookup.

Testing requirements will be added pending approval of a Lookups Resource by the Certification Subgroup and Transport Workgroup. This has also been noted in the section on Data Type Mappings. Depending on the progress in the groups, the allowance for string-based lookups may precede testing rules for a general Lookups resource.

Additional References

The current version of the generated BDD acceptance tests from which the Sample BDD Tests above were taken from may be found here. Note that this link will be updated once the Data Dictionary 1.7 testing tool codebase has been merged into the main branch.

Additional Compliance Checking

In addition to finding exact matches for Standard Resources, Fields, and Lookups, algorithmic techniques and heuristics will be used to determine potential matches with the RESO Data Dictionary standard.

In contrast with the other testing methodologies outlined in this document, the techniques used for additional compliance checking exist to enforce the stated policy that data being presented MUST match the RESO Data Dictionary format when they exist on the server. These methods will continue to be refined over time.

The methods will be published along with the Data Dictionary testing tool for transparency, community review, and to allow self-assessment by applicants prior to RESO Certification.

Informational messages will be generated in cases where potential matches with an existing Data Dictionary definition is found.

Some of the techniques used are described in the following sections.

Synonym Matching

The metadata for a given server is checked for synonyms at the resource, field, and enumeration level.

Synonyms MUST NOT be used at the field or lookup level. If a synonym is found in the list of field names returned by a server, certification will fail.

Examples of Synonym Checking are shown in the sample Timestamp testing rules.

***Note**: This feature is complete and will be included in the MVP.*

Similar Name Matching

Edit distance matching has been incorporated into the RESO Commander in order to find potential variations of Data Dictionary Resource, Field, and Enumeration Standard Names. Specifically, the Levenshtein Distance method is used.

The edit distance computes the number of edits required to turn one string into another. For example, the edit distance between *BayWindow* and *BayWindows* is 1.

A configuration value has been provided that allows the "fuzziness" threshold to be set to a fraction of the length of each term, currently greater than 25% of the word length. This means that terms of length 5-8 characters will allow up to 1 edit distance variation, and 9-12 will allow 2

variations, etc. The threshold has been chosen to provide a low error rate, while still providing meaningful fuzzy matching results.

Edit distance matches within the given threshold will trigger an error in the Data Dictionary Commander. Unresolved matches will not be granted exceptions and will prevent certification from proceeding.

Due to the probabilistic nature of "fuzzy matching," some false negatives may be generated when local terminology too closely resembles RESO Standard items.

Applicants are expected to provide a list of corrections in a configuration file they will submit at the time of certification to address any cases that arise. These corrections will be added to the testing tool in order to ensure that once a particular case has been addressed, it won't be flagged again in other cases.

***Note**: This feature is complete and will be included in the MVP.*

Substring Matching

Using substring matching, each Data Dictionary item of a given class will be compared to all the allowed terms for the given RESO Standard Resource, Field, or Lookup using the *Longest Common Substring* method.

Substring matching on word fragments will be performed in both directions using metadata returned from the server on one side and RESO Data Dictionary definitions on the other. If an applicant provided *Bay* in the *WindowFeatures* enumeration, it would substring match against *BayWindows* and be flagged as a potential match. Like *Similar Name Matching*, a configuration file will be used to ignore any false negatives.

***Note**: This feature will not be included in the MVP.*

Data-Driven Matching

RESO certification testing has progressively become more prescriptive and proscriptive over time. Exceptions for nonadherence to Data Dictionary rules were previously granted when an applicant filed a Supplemental Additional Information (SAI) request with RESO.

While SAIs are no longer part of the RESO Certification process, information that was previously recorded will be used to create a corpus of matching terms to help guide vendors toward the Standard Names for Resources, Fields, and Enumerations.

***Note**: This feature **will not** be included in the MVP. It will be released as part of a later enhancement.*

Reporting

Data Collection

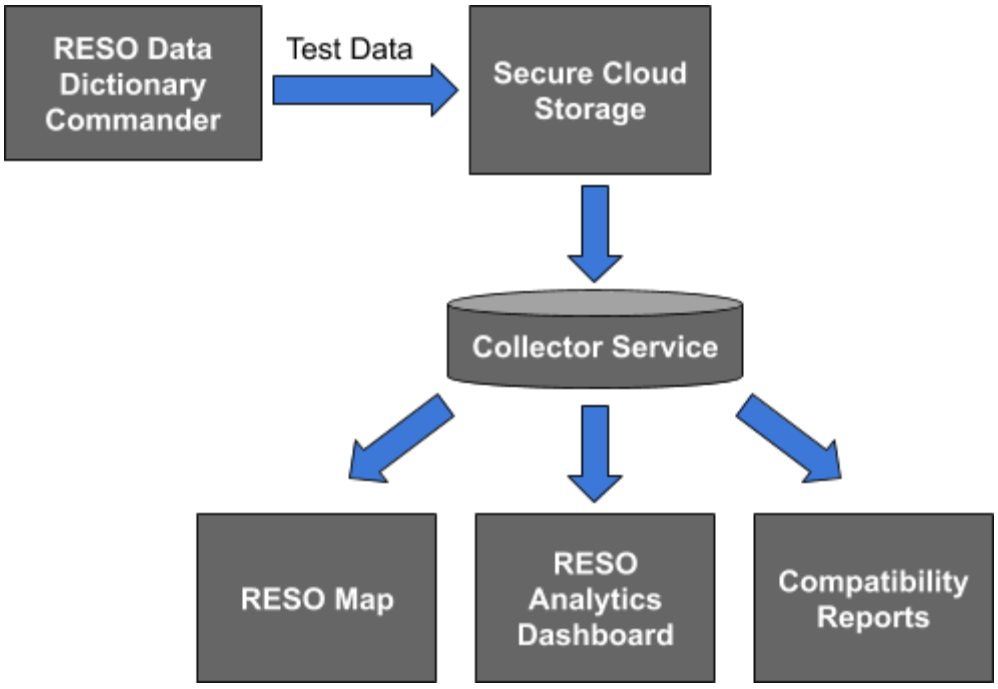
Metadata for a given server instance will be consumed by the RESO Commander in the OData XML CSDL metadata format but is not stored locally. Data analysis is done in memory and discarded upon termination of the application so applicants' source code is not retained.

A report will be generated when a certification application is processed that will contain statistics about what was found on a server when the testing tool was run. The report will be used to help the RESO Certification Department and the applicant evaluate results. The report will be emailed to the applicant and kept on file at RESO as proof of certification.

The RESO Commander will also produce summary test statistics in the JSON format with the results of each test step and include relevant data such as Resources, Fields, and Lookups found during testing. These reports will be uploaded into a RESO data collection service for the purpose of analytics.

Data Collection Pipeline

Test data will be collected for analytics purposes. This information will be stored on a cloud drive in order to catalog results.



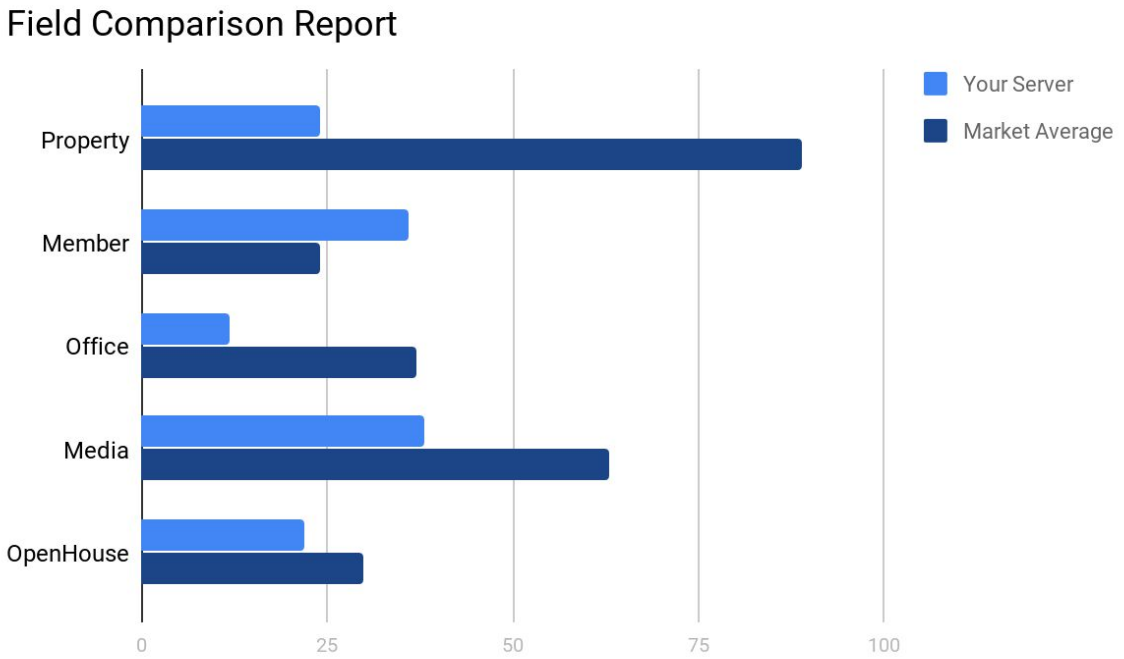
Once test results are stored, they are sent to a collector service for analysis. The collector will be implemented in ElasticSearch.

While the Collector Service and ancillary reports will be delivered after the MVP testing tool, test data will be available from an API so that analytics may be shown on the RESO Certification Map during the initial release of the Data Dictionary testing tool.

RESO Certification Map

Certification results will be published to the RESO Certification Map, which shows information about certified applicants in a geographical manner.

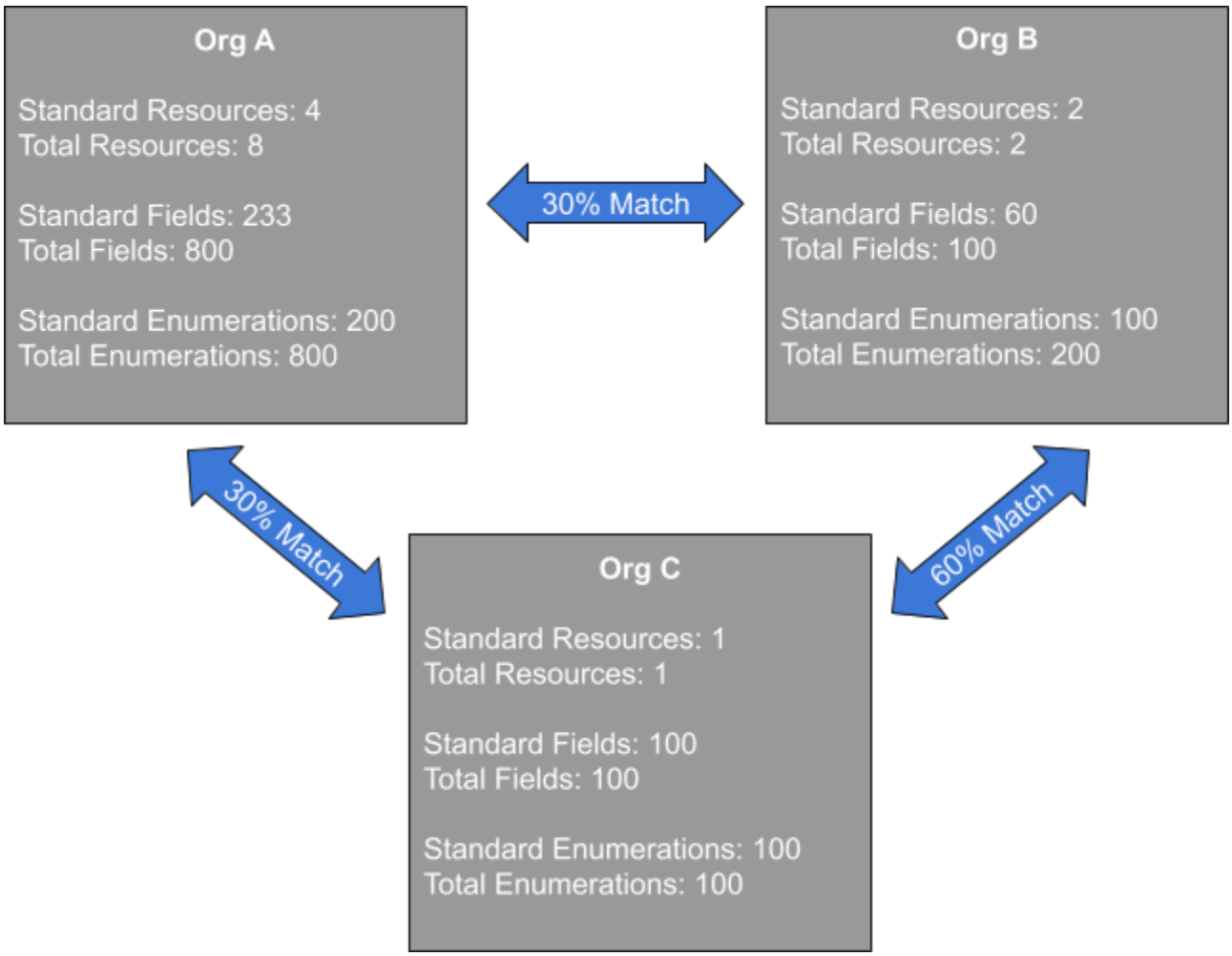
These information includes, but is not limited to (1) a report showing the RESO Standard Resources, Fields and Lookups in relation to the total number available on a per-resource basis; and, once enough aggregate data have been collected, (2) a field comparison report showing how an applicant scored relative to the market average, as shown in the following diagram:



RESO Data Compatibility Report

A comparison tool will be created to show alignment between resources, fields, and lookups between two or more RESO certified organizations. This will be useful for planning conversions and data shares, among other things.

While the reporting format has yet to be decided, conceptually the tool will find the intersection and difference between sets of resources, fields, and lookups between organizations. The information needed to produce these reports will be produced upon the initial release of the Data Dictionary testing tool, and a web-based UI will be created at a later time.



RESO Analytics Dashboard

An analytics dashboard will be populated with testing data, and will be driven by Kibana, a popular real-time analytics tool. This dashboard will be available to RESO staff and workgroup chairs for planning purposes and to provide information regarding adoption of RESO standards.

Display of Information on RESO Website

RESO may use anonymous aggregates collected during the certification process for display on its public websites. These items consist of Resource, Field, and Enumeration tallies but will not be displayed for a given area so as not to reveal the source, unless permission is specifically granted. Aggregate summary reports will be available at the Resource, Field, and Enumeration level.

For example:

- For each discovered resource, how many implementations have that resource?
- For each discovered field, how many implementations have that field?
- For each discovered enumeration, how many implementations have that enumeration?

Data Retention Policies

Applicants and certification recipients have the right to be forgotten.

At the time of writing, the Data Dictionary testing tool does not store any information during automated testing aside from generating a local log during runtime and producing JSON-based test results used for reporting.

RESO will be retrieving and saving server metadata in XML (EDMX) format at the time of Data Dictionary Certification for further analysis and to show what was retrieved from the server at the time of testing in case future questions arise. Metadata will be stored securely in the cloud and not available publicly. Information about resources, fields, and lookups found in the metadata during certification will be created as a derivative report.

Certification Workflow

The Certification workflow has been optimized around self-assessment prior to certification.

Self Assessment

It's expected that applicants will ensure they pass all RESO Data Dictionary tests and have reviewed results to their satisfaction prior to applying for certification.

Guides exist to help them with the evaluation process
TODO: Create guide.

Any questions regarding automated testing tools and revised certification procedures should be directed to Josh. For any other questions, or to start the certification process please contact RESO Certification.

Application

Those seeking RESO Certification will apply with the Membership Department prior to having their application reviewed by the Certification Department. Once an application has been processed, RESO will confirm the outcome of the automated testing tools using a RESOScript provided by the vendor, as described in the next section.

Certification Review

A RESOScript file is required for review. This file should contain credentials and the service location of the Web API Server instance hosting the Data Dictionary metadata to be tested.

Certification Issuance

RESO will verify that a given configuration produces no errors, unhandled warnings, or exceptions, at which point the applicant will be issued a certification report. Once the results have been confirmed, the applicant will be issued a confirmation of their certification.

Once certification has been issued, the applicant's status will be updated on the RESO Certification website and a field report will be presented as well as a count of the number of RESO Standard Resources, Fields, and Lookups found during the certification process as well as a count of local items.

Feature Requests

Feature requests can be requested as issues on the RESO Commander's GitHub project or by contacting Josh.

Support

To apply for certification, or for help with an existing application, please contact RESO Certification.

For questions about revised certification procedures or for help or questions about RESO's automated testing tools, please contact Josh.

Contributors

Thanks to the following contributors for their help with this project:

Sergio Del Rio
Eric Finlay
Dylan Gmyrek
Rob Larson
Paul Stusiak

If you would like to contribute, please contact Josh or RESO Certification. This could mean anything from QA or beta testing to technical writing to doing code reviews or writing code.

Also, thanks to FBS for donating prototypes of their BDD acceptance tests to be used as models, and to the Austin Board of REALTORS® for donating RESO Web API reference servers.