# REBR: A Standard Notation for Real Estate Business Rules

## Reference Manual

V1.0

8-4-2017

**Revision History**

| Date | Version | Details |
|---|---|---|
| 6/14/17 | 1.0 | Version 1.0 DRAFT is published for community review. |
| 6/17/17 | 1.0 | Fixed typos. Added clarification about Require_Value, Constrain_... |
| 8/17/17 | 1.0R | This is an updated copy which includes an additional command ALLOW_VALUE to represent optional fields. |
| | | |
| | | |

## Trademarks

*Semantics of Business Vocabulary and Rules*™ (SBVR™) is a trademark of the standards body Object Management Group (OMG). SBVR™ is an OMG approved standard for documentation of business rules. See http://www.omg.org/spec/SBVR/

RuleSpeak ™ and its companion, DecisionSpeak ™ are trademarks of Business Rules Solutions, LLC. Both are implementations of OMG standard SBVR™. Both are made freely available by Business Rules Solutions, LLC. See BRsolutions.com and Rulespeak.com.

## Related Documents

- ***REBR: A Standard Notation for Real Estate Business Rules - Formal Syntax Specification in EBNF***

## EXECUTIVE SUMMARY

The need for a standard notation for documenting and communicating Real Estate Business Rules was recognized by RESO several years ago. However, a viable standard has been elusive until now.

This document proposes a standard notation – the Real Estate Business Rule (REBR) notation. Starting with the "Structured English" notation provided by RuleSpeak™, REBR was formulated as an equivalent notation. REBR uses a predictable, parseable syntax, for clearly and unambiguously specifying several real estate business rules. **Just six of these REBRs, including fewer than a total of twenty syntactic variants, are sufficient to fully specify almost all of the business rules governing input of listing data for most MLSs**.

An example is provided below.

| | |
|---|---|
| ***Rule in "raw" form*** | MLS Staff and Super Users may ignore the requirement to complete required fields - changing a listing to closed status but not filling in the ClosingDate with only a warning. Agents and Brokers may not ignore this requirement. |
| ***Rule in Structured English, using common Enforcement Levels*** | A Closed Listing **must have** ClosingDate.<br>*Enforcement Level:*<br>Warning for MLS Staff and Super User;<br>Strict for agents and brokers. |
| ***Rule in proposed REBR notation*** | **REQUIRE_VALUE**<br>FIELD ClosingDate<br>**IF** ListingStatus = "Closed" **ENDIF**<br>**;;**<br>**ENFORCEMENT** WARNING **IF** UserRole INLIST ("MLS STAFF", "SUPERUSER") **ENDIF;;**<br>**ENFORCEMENT** STRICT **IF** UserRole INLIST ("AGENT", "BROKER") **ENDIF ;;**<br>**_end_Rule** |

The notation is independent of how the rule is implemented – be it in the user interface, a stored procedure in the database, application software, or one of many unwritten assumptions implicit in the vendor platform.

Our current focus is on developing a notation for MLS business rules governing input and maintenance of Listing data. Rules governing the display of Listing data are not yet addressed.

All of the rules for a given real estate organization can be packaged into a **REBR_RULEBOOK** construct which can then be transmitted electronically.

**Formal syntax specifications for REBR can be found in the document "*REBR: A Standard Notation for Real Estate Business Rules - Formal Syntax Specification in EBNF".***

**This document is a Reference Manual, which supplements the formal specification document.**

## CONTENTS

# 1. ABOUT THIS DOCUMENT

**This document is a draft standard for the Real Estate Business Rules (REBR) notation.**

Formal EBNF syntax specifications for REBR can be found in the document "*REBR: A Standard Notation for Real Estate Business Rules - Formal Syntax Specification in EBNF".*

# 2. INTRODUCTION

The need for a standard notation for documenting and communicating Real Estate Business Rules was recognized by RESO several years ago, but a substantive effort was not initiated by the R&D work group until late 2015, when there was an increase in industry interest in this area. MLSs with well-documented business rules can move to a new MLS system, add additional front-ends, or integrate other software that requires use of business rules, all without manual gathering of business rules and inaccurate results. Creating a standard for real estate business rules will result in smoother conversions, more software choice, and enhanced competition and innovation.

Efforts to standardize business rules can be elusive for two key reasons:

1. The *statement* of business rules can easily be bound by the system in which they are implemented, or the local requirements and context of the implementation.
2. The *communication* of business rules can likewise easily be subject to the exigencies of a particular project or of application silos and legacies.

As a result, business rules analysis can fall into a common trap: resorting to a particular Business Rules Engine (BRE) technology for an easy answer. The result is a form where rules are written in "computerese" and often buried in the specific software implementation. The rules are then neither accessible to, nor understood by the business stakeholders. Thus, the very people who make and are responsible for the rules, often do not have insight into what has been implemented.

In the real estate context, where the rules are generally vendor-documented, the documentation format has come from the vendors' BRE technologies. The rules have been output in forms that are both system and programming language specific, and defined by the context of the particular vendor product in use at the time. The rules are also often incompletely documented, reflecting platform-specific functional assumptions and processes. Lastly, rules change, often constantly, and the negative effect of rules being inaccessible to the business people compounds over time.

This time around, the starting point for defining the **Real Estate Business Rule (REBR) notation** was RuleSpeak™[1]. Field tested since the 1990s, RuleSpeak ™ is a well-documented business rule notation. Furthermore, RuleSpeak ™

[1] RuleSpeak ™ and DecisionSpeak ™, from Business Rules Solutions, LLC, are implementations of the Object Management Group (OMG) approved standard called *Semantics of Business Vocabulary and Rules*™ (SBVR™). See http://www.omg.org/spec/SBVR/ and http://www.rulespeak.com/en/

includes guidelines, syntax and patterns to express business rules effectively in "Structured English". The "Structured English" notation is perfect for clearly and unambiguously expressing business rules, even apparently complex ones, in non-technical language using business vocabulary.

However, there is a big learning curve to correctly use Structured English. To alleviate this issue, starting with the "Structured English" notation provided by RuleSpeak™, REBR was formulated as an equivalent notation. REBR uses a predictable syntax for clearly and unambiguously specifying several real estate business rules. **Just six of these REBRs, with fewer than a total of twenty syntactic variants, are sufficient to fully specify most of the business rules governing input of listing data for most MLSs**.

The notation is independent of how the rule is implemented – be it in the user interface, in application software, a stored procedure in the database, or one of many unwritten assumptions implicit in the vendor platform.

The current focus is on developing syntax for MLS rules governing input and maintenance of listing data, and secondarily how listings will automatically change over time in a database. Rules governing the display of listing data are not addressed in this document at this time.

# 3. REBR NOTATION - A HUMAN CAN UNDERSTAND IT AND A COMPUTER CAN PARSE IT

## REBR Design Objectives

What criteria must a proposed standard notation for real estate business rules satisfy?

We adopted the following:

- A **business stakeholder** can read and ***understand*** each rule ***unambiguously***. [Put on your business hat!]
- A **computer** can read and ***parse*** each rule ***predictably***. [Put on your IT hat!]
- All rules for a specific MLS or can be ***packaged*** in a ***"Rulebook"*** which can be electronically transmitted.

Thus, the proposed notation must provide the ability to state real estate business rules

- in terms such that another human will understand the rule clearly and unambiguously.
- using a concise, predictable syntax that can be correctly parsed (to extract each component of the rule) programmatically. This is necessary so that two parties can exchange the rules electronically.
- as a collection of rules for an MLS, i.e., as a "Rulebook" for that MLS.

The REBR notation proposed here meets the above objectives.

## Listing Input and Maintenance Rules

The current focus is on developing syntax for MLS rules governing input and maintenance of listing data. Rules governing the display of listing data are not addressed in REBR at this time.

## Misconception: A Business Rule is no more than an IF…THEN…ELSE statement.

When we first started researching notations for business rules, we started with the same assumption as everyone else: A business rule is nothing more than an "IF…THEN…ELSE…" statement. That notion was quickly dispelled as we better understood the Structured English approach pioneered by Ron Ross of Business Rule Solutions. There are several issues with the IF…THEN...ELSE approach of stating business rules. We mention a few here.

1. **Not understood by business & non-technical persons:** It perpetuates the computer programmer mindset, which focuses on implementability at the expense of clear and unambiguous understanding by non-technical business people.
2. **Obscured by details:** The intent of the rule is in the details of the IF…THEN conditions, especially when they are complex.
3. **Artificially imposed IF/THEN:** Not all business rules are amenable to the IF…THEN format, which is then artificially applied. For example, where is the IF/ENDIF in the rule: "An Agent must have a name."?

Hence, you will not see a single REBR rule structured in an IF…THEN…ELSE format. Instead, the general format of each REBR rule is:

- Starts with a rule name which indicates the rule purpose, e.g., "REQUIRE_VALUE"
- Followed by what the rule applies to, e.g., "FIELD *fieldname*"
- Followed by applicable conditionals, i.e., the IF…ENDIF block
- Finally, additional rule details are added, including Min/Max counts, Lookup list names, Definitions etc.

We leave the IF…THEN…ELSEs to the implementation, where they are needed.

## REBR is NOT Automated Rule Implementation!

It is important to get one thing out of the way first. **The notation does not provide a syntax which can be directly ingested by the recipient's business rule engine (BRE) and implemented automatically without some human intervention** – though many of the simpler rules happen to have a syntax that *can* be directly ingested if the implementation is so designed. Automated Business Rules Engine (BRE) implementation is not feasible with current technology because there are too many differences among BREs – each has different automation capabilities, and each implements a different syntax, security model and so forth. Furthermore, REBR provides the flexibility to specify certain elements in natural language. These elements will usually require a human to interpret the meaning of the rule.

## A silly example to illustrate the point.

Consider the following, rather silly business rule.

**In plain language:** If the moon is not made of cheese and pigs cannot fly, then APN is required.

**In Structured English:** APN must have a value if all of the following are true: (1) the moon is not made of cheese, (2) pigs cannot fly.

***Analysis*:** A value must be entered for APN if the conditions are satisfied. Hence, the REBR rule is as follows[2].

**Rule In REBR Notation:**

**REQUIRE_VALUE**
**FIELD APN**
**IF (The moon is not made of cheese) AND (Pigs cannot fly) ENDIF ;;**
**ENFORCEMENT STRICT**
**ROLE** (All user roles)
**MSG "**APN must have a value If (the moon is not made of cheese) and (pigs cannot fly)" **ENDMSG**
**;;**
**_end_Rule**

Perhaps the programmer in you is thinking: But how can my system implement the conditions in this rule? The answer is: it varies. If your implementation presents a human being with pop-ups to answer the questions – "Is the moon made of cheese?" and "Can pigs fly?" you should have no problem getting the correct result. On the other hand, if you want to automate the rule execution, perhaps IBM Watson's artificial intelligence may be needed. Or, perhaps in anticipation of having to answer these questions programmatically, you may have implemented each question and its answer in a lookup table.

In any case, the REBR notation successfully met the criteria set forth above as you can see from the following:

- Question 1: Can a human understand the rule unambiguously? Answer: Yes
- Question 2: Can a computer parse the above rules predictably? Answer: Yes. An EBNF formulation of the rule provides parsing details.
- Question 3: Can all the rules for an MLS be packaged and transmitted in a RuleBook construct? Answer: Yes.

The point is that the REBR notation adequately handles even the above rather complex, unstructured conditions, in part because it did not get tainted by how you might choose to implement the business rule in your own software.

Another important point to note is that **the condition described in the IF/ENDIF clause does NOT need be stated in terms of computer programming variables – English works just fine!** This will be particularly true for new rules which the MLS has not yet implemented. In fact, the recommended approach is to avoid using implementation variable names, and refer to fields by their business names (e.g., "List Price" instead of "Price-Lst").

[2] Of course, your analysis may correctly interpret the rule to be simply "APN is ALWAYS Required", and you can state the Rule as: **REQUIRE_VALUE FIELD APN UNCONDITIONAL;;_end_Rule**.

## Can REBR handle Rule-Chains?

In our experience, so called Rule-Chains result from mixing business rule(s) with the sequence in which a specific implementation happens to execute them. Upon analysis, we find that the underlying business rule(s) can be stated in REBR without difficulty. An example is provided below.

## EXAMPLE: A "Rule-Chain" in Raw, Structured English, and REBR Notation.

**An MLS has the following "raw" rules.**
*Rule 1.* An image is required in all listing Statuses (image required at this stage).
*Rule 2.* If "under construction" is selected, an image is not required (overriding Rule 1).
*Rule 3.* If property is sold, an image is required (an image required now, overriding Rule 2).

**Analysis**
Using guidelines from RuleSpeak, the three rules above translate into a single rule as follows, with no need for complicated and difficult to test rule sequencing.

***Rule in Structured English:***
A listing must have at least one image if the real property is not under construction.

***1st TRY - Same Rule in REBR notation:*** **Not recommended.**
**Uses Implementation specific variables to specify the rule.**

**REQUIRE_VALUE**
**FIELDGROUP** ImageFieldGroup **//Implementation specific field group is used instead of Object.**
**MINCOUNT 1**
**IF** ConstructionStatus **NE** "Under Construction" **ENDIF** **//Implementation specific field name is used**
**DEFINITION** ImageFieldGroup **MATCHING** (Picture%) **ENDDEFINITION**
**// Note: For this MLS, Image fields are named Image1 through Image20**
**;;**
**_end_Rule**

***2nd Try – BETTER! Same Rule in REBR notation:*** **Recommended.**
**Rule statement is independent of implementation.**

**REQUIRE_VALUE**
**OBJECT**
**IF** (Property is NOT Under Construction) **ENDIF**
**PropertyImage MINCOUNT 1**
;;
**DEFINITION** PropertyImage is an image file containing a current photograph of the listed property. **ENDEFINITION**
;;
**_end_Rule**

**Note**
An alternate interpretation of the rule can be stated in Structured English as follows.

- A listing must have at least one image if any of the following conditions is true
  - the real property is not under construction
  - the real property is sold

This interpretation can easily be stated in REBR by modifying the IF /ENDIF conditional block in either of the two formulations above.

## 4. NOTATION

In describing the REBR syntax, the following notation is used. In this Reference Manual, the EBNF notation is not explicitly used.

**IMPORTANT:**

In REBR syntax

- Color coding is used to aid comprehension, but not required.
- Rule statements are case insensitive. Upper or lower case is used for better readability but not required.
- Indentation and line breaks are used for better readability, but not required.

Thus, an entire syntactically correct rule statement can be written on a single line with no indents or line breaks, in plain black and white, using any choice of upper/lower case, but it will still be syntactically correct.

| | |
|---|---|
| **RULE_NAME** | Rule name, e.g., ALLOW_EDIT. Case insensitive. |
| **KEYWORD or keyword** | A keyword or a constant. Case insensitive. For example: **DEFINITION, ENDDEFINITION, _end_Rule,** etc**.** |
| ***aParameter*** | Replace by the value of the specified parameter. See Appendix for definitions. |
| ***aRebrClause*** | Replace by a REBR Clause, such as **ENFORCEMENT** clause or **TRIGGER** clause |
| **$[ ]$** | Syntax enclosed between $[ and ]$ is required. The brackets and $ signs are omitted. |
| **[ ]** | Syntax enclosed between square brackets is optional. The brackets are omitted. |
| **...** | Preceding syntax element can be repeated. The ellipsis is omitted. |
| a **\|** b | Exactly one of **a** or **b** can be specified. The vertical bar is omitted. |
| default | Suggested default value is underlined. |
| **//text** | Explanatory comments. Not part of the Rule. |
| **;;** | Double semicolons are used to mark the end of specific segments in a rule statement. This removes ambiguity in parsing, as well as improves readability. |
| **( )** | Parentheses may be used whenever necessary to avoid ambiguity. |

### Historical Values

Sometimes, there is a need to differentiate between the existing value versus the user entered value. We also came across a rule that referenced a historical value – i.e., the one that existed just prior to the present value. To differentiate between current and historical values, the following keywords are used where necessary.

| Keyword | Description |
|---|---|
| **NEW** ***aFieldName*** | Defined as the value of ***aFieldName*** which, if persisted, will replace CURRENT ***aFieldName***.<br>[Think of this as the value entered by the user on the screen]. |
| **CURRENT** ***aFieldName*** | Defined as the currently persisted value of ***aFieldName*** |
| **CURRENT** ***aSystemFieldName*** | Defined as the System value of ***aSystemFieldName*** at the time the system is queried for that field. |
| **PRIOR** ***aFieldName*** | Defined as the historical value of ***aFieldName*** immediately prior to the CURRENT value. |

An example is provided below using the CONSTRAIN_VALUE rule.

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Status of an Active Listing of Residential Property Type **may only change** to one of the following: "Active", "Cancelled", "Extended", "Under Agreement", "Temporarily Withdrawn".<br><br>Analysis: The above rule is applicable to residential listings whose **current** status is "Active". The rule limits the choice of **new** status value for the listing. | **CONSTRAIN_VALUE**<br>FIELD Status<br>IF<br>**CURRENT** Status = "Active" AND ListingPropertyType = "Residential"<br>ENDIF<br>INLIST("Active","Cancelled","Extended","Under Agreement", "Temporarily Withdrawn")<br>;;<br>**_end_Rule** |

## 5. LIST OF RULES

### Core Rules for Input of Listing Data

1. **ALLOW_EDIT**
   - **Syntax 1: Single Field**
   - **Syntax 2: Group of Fields**
   - **Syntax 3: Group of Listings**
2. **REQUIRE_VALUE**
   - **Syntax 1: Single field**
   - **Syntax 2: Group of fields**
   - **Syntax 3: Objects**

3. **REQUIRE_EMPTY**
   - **Syntax 1: Single field**
   - **Syntax 2: Group of fields**
4. **ALLOW_VALUE**
   - **Syntax 1: Single field**
   - **Syntax 2: Group of fields**
   - **Syntax 3: Objects**
5. **DERIVE_VALUE**
   - **Syntax 1: Single field (computation)**
   - **Syntax 2: Single field (system value)**
   - **Syntax 3: Single field (autopop)**
   - **Syntax 4: Group of fields (autopop)**
6. **CONSTRAIN _DATE_TIME**
   - **Syntax 1: Specific date**
   - **Syntax 2: Date Range**
7. **CONSTRAIN _VALUE**
   - **Syntax 1: Single Value**
   - **Syntax 2: Value Range**
   - **Syntax 3: Filter**
   - **Syntax 4: Prohibited values**
   - **Syntax 5: Single-Lookup List**
   - **Syntax 6: Multi-Lookup List**

## Other Rules

8. **CONSTRAIN _MEDIA**
   - **Ensures media can be accepted – file types and size, pixels for images**
9. **ALLOW_DELETE_LISTING**
   - **Manual listing deletion**
10. **WATERMARK_PHOTO**
    - **Image file details; content and location of the watermark**
11. **AUTO_UPDATE**
    - **Time triggered Listing status change**

## Proposed Rules not yet finalized.

The following proposed rules need further analysis before they can be finalized.

1. **COPY_LISTING**
   - **Clone field values and media from existing listing into new listing**
2. **AUTO_UPDATE MEDIA**
   - **For example, delete all non-primary photos one year after listing is off-market**

3. **ALLOW_DUPLICATE_LISTING**
   - **Address duplicate listing cross-listing**
4. **Rule which incorporates RETS Query specifications**

**Work done so far on these rules is included in the Appendix.**

## 6. THE EXPRESSIVE POWER OF REBR – AN EXAMPLE RULEBOOK

Before delving into the nitty gritty details of REBR syntax, a small REBR example is provided below to demonstrate the expressive power of REBR.

Shown below is a skeletal **REBR Rulebook**. The Rulebook is a complete collection of all business and process rules for a given MLS or Vendor. There are two types of REBR rules.

- **REBR Business Rules DO NOT alter data**. Hence, a group of REBR Business Rules can be executed in any order, and/or any number of times, to achieve the **same outcome.**
- **REBR Process Rules DO alter data**. Hence, execution sequence as well as the number of times the rule is executed will result in **different outcomes**.

The Rulebook example shown below focuses on a single listing type (aka Scope) of "SFR-Sale" with just 4 business rules which may be difficult to express clearly in other notations.

```
RULEBOOK "Business Rules for MyMLS"
// In a future REBR version, consider use of OUID and more structure in RULEBOOK name.
//=======Scope = SFR-Sale  ===========
SCOPE  (
Name ="SFR-Sale";
TransactionType=Sale;
PropertyType=Residential;
PropertySubType=SingleFamily;
DEFAULT ENFORCEMENT
        STRICT
        ROLE (All User Roles)
        MSG (Same as Rule Description) ENDMSG
        ;;
DEFAULT TRIGGER OnSave
        ;;
)
_begin_Scope
//=======Business Rules for Scope SFR-Sale ===========
//Rule #1 Description: List Price is Required.
REQUIRE_VALUE
        FIELD  List Price
        ;;
_end_Rule
//Rule #2 Description: SFR with List Price above $3M must have at least: 2 BR, 2BA, 1 Kitchen
//Enforcement: Coordinator can override. For all other roles, default applies.
REQUIRE_VALUE
```

```
        OBJECT
        IF  ListPrice GREATERTHAN $3M  ENDIF
        Bedroom  MINCOUNT 2,
        Bathroom  MINCOUNT 2,
        Kitchen MINCOUNT 1
        ;;
RULELEVEL ENFORCEMENT  AUGMENTS SCOPELEVEL ;;
ENFORCEMENT OVERRIDE
        ROLE (Coordinator)
        ;;
_end_Rule
//Rule #3 Description: List Price must be between $1000 and $3 Million
//Enforcement: Warning for all Roles. Replaces default.
CONSTRAIN_VALUE
        FIELD List Price
        LOWRANGE  1000
        HIGHRANGE 3 Million
        ;;
RULELEVEL ENFORCEMENT  REPLACES SCOPELEVEL ;;
ENFORCEMENT WARNING
        ROLE (All user roles)
        MSG "WARNING"  + (Same as rule description)  ENDMSG
        ;;
_end_Rule

//Rule #4: List Price cannot be reduced or increased more than 20%
CONSTRAIN_VALUE
        FIELD List Price
        LOWRANGE  (CURRENT List Price minus 20%)
        HIGHRANGE (CURRENT List Price plus 20%)
        ;;
_end_Rule
_end_Scope
_end_RuleBook
```

## 7. SCOPE, ENFORCEMENT, SECURITY, TRIGGER, HISTORICAL VALUES

Structured English best practices advocate separation of the Rule from both Rule Enforcement and Rule Triggering. For clarity and the ability to have a configurable implementation, a third item has been separated from the individual rules: Rule Scope.

Each of these is discussed below.

### Rule Scope

## DESCRIPTION

Every rule will operate within a **scope**.

**Tip: Think of Rule Scope as the Listing Type.** Some examples are: SFR, Mobile-Manufactured, Commercial-Sale, Commercial-Lease, etc.

**A Scope statement applies to all REBR Rules between the "_begin_Scope" and the "_end_Scope" tokens.**

## SYNTAX

```
SCOPE (
[ Name = aScopeName ; ]
[TransactionType = ( $[ aTransactionType ]$   [, aTransactionType]... ); ]
//Optional list of comma separated
//Transaction Types. Ended by semi-colon.
$[
PropertyType=aPropertyType,
PropertySubType=aPropertySubtype;
]$...
//1 or more Property Type+SubType pairs
[ DEFAULT  ENFORCEMENT
        $[STRICT | WARNING | OVERRRIDE]$
        ROLE   INLIST  (aUserRole, ...)
        [MSG  aMessage  ENDMSG]
        ; ;
]...
[DEFAULT  TRIGGER
        $[
        aUserAction |
        aSpecifiedTime |
        aEvent          |
        aIfEndif
        ]$
        ; ;
]...
)
_begin_Scope
//-NOTE: All rules in this scope go here ---
//-NOTE: All rules in this scope go here ---
//-NOTE: All rules in this scope go here ---
```

## _end_Scope

### EXAMPLE

**SCOPE (**
**Name = SFR-Sale;**
Transactionype=Sale;
PropertyType=INLIST (Residential), PropertySubType=SingleFamily;
DEFAULT ENFORCEMENT STRICT
ROLE INLIST (All user roles except Coordinator)
MSG (same as Rule Description) ENDMSG
;;
DEFAULT ENFORCEMENT OVERRIDE
ROLE INLIST (Coordinator)
;;
DEFAULT TRIGGER OnSave
**)**
**_begin_Scope**
**// One or more REBR rules for SFR-Sale Listings are stated here.**
**//….**
**_end_SCOPE**

The above Scope statement implies that the associated rules are applicable to

- Scope named "SFR-Sale" which is defined to include listings with
  - Transaction Type= Sale
  - Property Type and Subtype combo of Residential, Single Family dwelling.
- Default Enforcements for different roles are specified for all rules under this scope.
- Default Trigger is specified for all rules under this scope.

## Rule Level vs. Scope Level Enforcement, Trigger

### DESCRIPTION

Rule level Enforcement and/or Trigger may either replace or augment the default Enforcement and/or Trigger specified at the Scope level.

- **Replace** means that the scope level default for Enforcement or Trigger is ignored.
- **Augment** means the following
  - ***For Enforcement***: The rule-level enforcement for specified User Roles replaces the corresponding scope level default. For other user roles, i.e., those for which no rule-level enforcement is specified, the scope level default enforcement is applied.
  - ***For Trigger*****:** The rule-level Trigger is treated as an additional trigger for that rule, in addition to scope level trigger(s).

The choice between replace and augment is indicated within the Rule by using the following clause.

### SYNTAX

**RULELEVEL**
**$[ ENFORCEMENT | TRIGGER ]$**
**$[ REPLACES | AUGMENTS ]$**
**SCOPELEVEL**
**;;**

**NOTATION: For convenience, the term a*RULE-LEVEL-DIRECTIVE* will be used to represent the above syntax.**

### EXAMPLE

```
RULELEVEL   ENFORCEMENT
        REPLACES   SCOPELEVEL
        ;;
```

## Enforcement Levels and Security

### DESCRIPTION

Each rule will have one or more context-dependent Enforcement Levels. Default enforcement levels can be specified at the Scope level in REBR (or in the implementation, at any level, including globally). Enforcement Levels can be used to create specific types of rule overrides, such as 'strictly enforced', 'override with prior authorization', 'override with warning', or 'guideline'.

Currently, for simplicity, three levels of Enforcement are proposed in REBR.

- **Strict** means the rule is strictly enforced. The user may be shown a message, but will not be allowed to proceed further.
- **Warning** means the user may be shown a message but will be allowed to proceed further.
- **Override** means the user will bypass the rule with no message.

### SYNTAX

## ENFORCEMENT

**$[STRICT | WARNING | OVERRRIDE]$**
//**Strict** means the rule is strictly enforced,
//**Warning** means the user may be shown a message but will be allowed to proceed
//**Override** means the user will bypass the rule with no warning.

**ROLE INLIST (*aListOfUserRoles*)**
**[MSG *aMessage* ENDMSG]**
**;;**

**Where**
***aListOfUserRoles =***
***aUserRole* [, *aUserRole*]...**
**is a comma separated list of user roles**

**NOTATION: For convenience, the term a*ENFORCEMENT* will be used to represent the above syntax.**

## EXAMPLE

### Example 1:

**ENFORCEMENT WARNING**
ROLE INLIST (Agent, Coordinator)
MSG "Warning: List Price cannot be less than $1000."
ENDMSG
**;;**

The enforcement clause may be repeated, allowing a rule to specify different enforcement levels based on Role. For example: A rule may specify STRICT enforcement for Salespersons and Brokers, and WARNING for MLS staff.

### Example 2:

| | |
|---|---|
| **Rule and Enforcement instructions in plain English:** | MLS Staff and Super Users may ignore the requirement to complete required fields - changing a listing to closed status but not filling in the ClosingDate with only a warning. Salespersons and Brokers may not ignore this requirement. |
| **Rule in Structured English, using common Enforcement Levels** | **Rule**: A Closed Listing must have ClosingDate.<br>**Enforcement Level:** Override with a warning for MLS Staff and SuperUser.<br>**Enforcement Level:** Strict for Salespersons and Brokers. |
| **Rule in REBR notation, with Enforcement Levels** | **REQUIRE_VALUE FIELD** Closing Date<br>IF ListingStatus = "Closed" ENDIF<br>**;;**<br>**RULELEVEL ENFORCEMENT** |

| | |
|---|---|
| | **REPLACES SCOPELEVEL**<br>**;;**<br>**ENFORCEMENT** WARNING<br>ROLE INLIST (“MLS STAFF”, “SUPERUSER”)<br>MSG “You did not enter a Closing Date for a Closed Listing. Continue?” ENDMSG ;;<br>**ENFORCEMENT** STRICT<br>ROLE INLIST (“SALESPERSON”, “BROKER”)<br>MSG (Same as Rule Description) ENDMSG ;;<br>**_end_Rule** |

## Trigger

### DESCRIPTION

Each rule is executed by one or more Triggers, which may be specified at the Scope level or at the Rule level. Triggers may refer back to activities, events in a business process model. Or they may reference specific dates/times, or “point-in-time” conditions enclosed in an IF/ENDIF block. Without having a common, defined business process model, triggers can only be described somewhat generically in REBR. For Listing input, the trigger will be OnSave or OnInput or something similar. For Listing maintenance, such as Auto_Update, the trigger will be an event, a point-in-time condition, or clock based.

### SYNTAX

**TRIGGER**
**$[**
***aUserAction*** **|**
***aSpecifiedTime*** **|**
***aEvent*** **|**
***aIfEndif***
**]$**
**;; //Two semicolons**
**//*aUserAction* = a user initiated action,**
**// such as OnSave or OnInput.**
**//*aSpecifiedTime* = a specified date and/or**
**// time, such as "2am every Wednesday"**
**//aEvent such as: arrival of a specific document.**
**//*aIfEndif*= a specified point-in-time logical condition**
**//Of course, the IF/ENDIF must enclose a condition that is only true at certain specific**
**//points in time. If the condition is true over a continuous time period, the rule may be triggered**
**//an infinite (or very large) number of times for the same listing.**

```
//For example: "TRIGGER IF Listing is Active ENDIF" will
//cause multiple executions of the triggered rule(s) – unless, of course, one of the triggered
//rules successfully changes the status to something other than Active.
// Multiple triggers may be specified.
```

**NOTATION: For convenience, the term a*TRIGGER* will be used to represent the above syntax.**

## EXAMPLE

```
REQUIRE_VALUE FIELD Closing Date … ;;
        ENFORCEMENT  … ;;
        RULELEVEL TRIGGER
                AUGMENTS SCOPELEVEL
                ;;
        TRIGGER (OnSearch);;
        //Where OnSearch is a defined event or activity in the business process model.
_end_Rule
```

## A Complete Rule Example

Putting together all the above into a single example, we may have a rule that looks as follows:

```
SCOPE (
        Name = SFR-Sale;
        Transactionype=Sale;
        PropertyType=INLIST (Residential),  PropertySubType=SingleFamily;
        DEFAULT ENFORCEMENT STRICT
                ROLE INLIST (All except Coordinator)
                MSG (same as Rule Description) ENDMSG
                ;;
        DEFAULT ENFORCEMENT OVERRIDE
                ROLE INLIST (Coordinator)
                ;;
        DEFAULT TRIGGER OnSave ;;
)
_begin_Scope
//Rule description: A Closed Listing must have ClosingDate
REQUIRE_VALUE
        FIELD Closing Date
        IF ListingStatus = "Closed" ENDIF
        ;;
RULELEVEL ENFORCEMENT
```

```
        REPLACES SCOPELEVEL ;;
RULELEVEL TRIGGER
        AUGMENTS SCOPELEVEL ;;
ENFORCEMENT WARNING
        ROLE INLIST ("MLS STAFF", "SUPERUSER")
        MSG "WARNING: You did not enter a Closing Date for a Closed Listing. Continue?" ENDMSG
        ;;
ENFORCEMENT STRICT
        ROLE INLIST ("SALESPERSON", "BROKER")
        MSG "A Closed Listing must have ClosingDate." ENDMSG
        ;;
TRIGGER (OnSearch);;
_end_Rule
// One or more REBR rules for SFR-Sale Listings are stated here.
//….
_end_SCOPE
```

### Interpretation

The above rule is interpreted as follows.

- The rule is applicable to all SFR-Sale listings, which is defined by Transaction Type = Sale, Property Type = Residential and Property Subtype = SingleFamily.
- Default Enforcements are Strict (for all roles except coordinator), and Override (for Coordinator).
- The rule states: A Closed listing must have Closing Date.
- Default enforcement are suppressed.
- MLS Staff and Super Users get a Warning message "WARNING: You did not enter a Closing Date for a Closed Listing. Continue?"
- The rule is applied strictly (no Warning or Override) for salespersons and brokers. The error message is "A Closed Listing must have ClosingDate."
- By default, the rule is triggered when the OnSave functionality is executed.
- Additionally, the rule is triggered when the OnSearch functionality is executed.

## 8. THE RULEBOOK

### DESCRIPTION

The Rulebook is the container for all Rules for a specific MLS.

### SYNTAX

**RULEBOOK** ***aRuleBookName***

```
$[
  SCOPE ( aScope )
  _begin_Scope
        $[ A REBR Business or Process Rule ]$...
  _end_Scope
]$
_end_RuleBook
```

Where
*aScope* **is replaced by detailed specification of the Scope per the Scope statement syntax.**

### EXAMPLE

A RuleBook example is provided earlier in this document, in the section entitled "The Expressive Power of REBR – An Example RuleBook. "

## 9. ALLOW_EDIT [1] - FIELD

### DESCRIPTION

This rule specifies whether a specified listing data field value is modifiable.

### SYNTAX

```
ALLOW_EDIT
FIELD  aFieldName
[ YES | NO ]
//Yes="Must be editable"
//No="Must NOT be editable"
[ UNCONDITIONAL | IF aCondition ENDIF ]
;;  //Two semicolons
[aRULE-LEVEL-DIRECTIVE ] …
//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level
[ aENFORCEMENT ] … // Zero or more Enforcements. If omitted, default(s) must be provided.
[ aTRIGGER ] … // Zero or more Triggers. If omitted, default(s) must be provided.
_end_Rule  // Marks the end of this rule
```

### EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| PublicRemarks must be editable. | ALLOW_EDIT FIELD PublicRemarks<br>;;<br>_end_Rule |
| ListingId must be non- editable. | ALLOW_EDIT FIELD ListingId NO<br>;;<br>_end_Rule |
| CloseDate must be editable if Listing Status is changed to "Closed" from another ListingStatus. | ALLOW_EDIT FIELD CloseDate YES<br>IF<br>(CURRENT ListingStatus NOTEQUAL "Closed" ) AND<br>(NEW ListingStatus="Closed" )<br>ENDIF<br>;;<br>_end_Rule |
| PendingDate must be non-editable if it presently has a non-empty value. | ALLOW_EDIT FIELD PendingDate NO<br>IF (CURRENT PendingDate is not empty)<br>ENDIF<br>;;<br>_end_Rule |

## 10. ALLOW_EDIT [2] - FIELDGROUP

### DESCRIPTION

This rule specifies whether data field values for a specified group of Listing data fields can be modified. FieldGroup is defined using the DEFINITION clause.

### SYNTAX

**ALLOW_EDIT**

**FIELDGROUP** ***aFieldGroupName***

**[ YES | NO ] //Default = YES**

**//Yes="Must be editable"**

**//No="Must be non-editable"**

**[ UNCONDITIONAL | IF *aCondition* ENDIF ] //Default = UNCONDITIONAL**

**DEFINITION**

***aFieldGroupName* = *aSelectionOfFields***

**ENDDEFINITION**

**;; //Two semicolons**

**[*aRULE-LEVEL-DIRECTIVE* ] …**

**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**

**_end_Rule** **// Marks the end of this rule**

**where**
***aSelectionOfFields*** **=**

**INLIST ( *aListOfFieldNames*)** |
**NOTINLIST ( *aListOfFieldNames*)** |
**PASS ( *aCondition*)** |
**FAIL ( *aCondition*)** |
**MATCH ( *aPattern*)** **[DEFINITION *aPattern* ENDDEFINITION]** |
**NOTMATCH ( *aPattern*)** **[DEFINITION *aPattern* ENDDEFINITION]**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Each of the following fields must be editable if CommissionYN is "Yes" [which indicates the user's intent to provide "Commission To" information]:<br>• BuyerAgencyCompensation<br>• SubAgencyCompensation<br>• TransactionBrokerCompensation<br><br>**Alternate Statement:**<br>Each field in CommissionFieldGroup **must be editable** if CommissionYN is "Yes".<br><br>CommissionFieldGroup consists of the following fields:<br>• BuyerAgencyCompensation<br>• SubAgencyCompensation<br>• TransactionBrokerCompensation | ALLOW_EDIT<br>FIELDGROUP CommissionFieldGroup<br>IF CommissionYN = "Y" ENDIF<br>;;<br>DEFINITION<br>CommissionFieldGroup =<br>INLIST (BuyerAgencyCompensation,<br>SubAgencyCompensation,<br>TransactionBrokerCompensation)<br>ENDDEFINITION<br>;;<br>_end_Rule |

# 11. ALLOW_EDIT [3] - LISTING

## DESCRIPTION

This rule identifies Listings which can or cannot be modified.

## SYNTAX

**ALLOW_EDIT**

**LISTING**

**[ YES | NO ] //Default = YES**

**//Yes="Must be editable".**

**//No="Must NOT be editable"**

**[ UNCONDITIONAL | IF *a*C*ondition* ENDIF ] //Default = UNCONDITIONAL**

**;; //Two semicolons**

**[*aRULE-LEVEL-DIRECTIVE* ] …**

**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**

**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**

**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**

**_end_Rule // Marks the end of this rule**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| An Expired Listing must be editable up to 15 days after Expiration Date. | ALLOW_EDIT<br>LISTING<br>IF<br>(Listing Status = Expired) and<br>TODAY = ONORBEFORE (Expiration Date + 15 DAYS)<br>ENDIF<br>;;<br>_end_Rule |
| A Closed Listing must be non-editable. | ALLOW_EDIT<br>LISTING<br>NO<br>IF Status = Closed ENDIF<br>;;<br>_end_Rule |
| Disclaimer must be non-editable.<br><br>Analysis: The intent is to make it non-editable once it has a value. | ALLOW_EDIT FIELD Disclaimer NO<br>IF (CURRENT Disclaimer is not empty)<br>ENDIF<br>;;<br>_end_Rule |

# 12. REQUIRE_VALUE [1] - FIELD

## DESCRIPTION

This rule specifies whether a specified listing data field must have a value, i.e., it cannot be empty, and default value if any.

## SYNTAX

**REQUIRE_VALUE**
**FIELD** ***aFieldName***
**[ UNCONDITIONAL | IF** ***condition*** **ENDIF ]**
**[ DEFAULT** ***DefaultValue*** **]**
**;;** **//Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| RoomSize must have a value | REQUIRE_VALUE<br>FIELD RoomSize<br>;;<br>_end_Rule |
| State must have a value.<br>State has a default value of "Minnesota". | REQUIRE_VALUE<br>FIELD State<br>DEFAULT "Minnesota"<br>;;<br>_end_Rule |
| AllowAVM must have a value.<br>AllowAVM has a default value of "Yes". | REQUIRE_VALUE<br>FIELD AllowAVM<br>DEFAULT "Yes"<br>;;<br>_end_Rule |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| StreetNumber must have a value if Status does not have a value of "Partial". | REQUIRE_VALUE<br>FIELD StreetNumber<br>IF Status NOTINLIST "Partial" ENDIF<br>;;<br>_end_Rule |
| GarageSpaces must have a value if GarageYN has a value of "Y". | REQUIRE_VALUE<br>FIELD GarageSpaces<br>IF GarageYN = "Y" ENDIF<br>;;<br>_end_Rule |
| AssessmentPending must have a value if the Status does not have a value of "CompOnly". Enforcement: MLS Staff can override. Rule level enforcement augments scope level enforcement. [Not shown here – Scope Level enforcement default is Strict for all user roles] | REQUIRE_VALUE<br>FIELD AssessmentPending<br>IF Status NOTINLIST ("CompOnly") ENDIF<br>;;<br>RULELEVEL ENFORCEMENT<br>AUGMENTS SCOPELEVEL<br>;;<br>ENFORCEMENT OVERRRIDE<br>ROLE INLIST (MLS Staff)<br>;;<br>_end_Rule |
| Disclaimer must have a value. Disclaimer has the default value of "Information Deemed Reliable, But Not Guaranteed." | REQUIRE_VALUE<br>FIELD Disclaimer<br>DEFAULT "Information Deemed Reliable, But Not Guaranteed."<br>;;<br>_end_Rule |

## 13. REQUIRE_VALUE [2] - FIELDGROUP

### DESCRIPTION

This rule specifies whether fields in a specified group of fields must have a value. A min and max count can be used to specify the number of fields which must have a value.

### SYNTAX

**REQUIRE_VALUE**

**FIELDGROUP *aFieldGroupName***

**[ UNCONDITIONAL | IF *aCondition* ENDIF ]**

**[ MINCOUNT 1 | *aIntegerFrom0toMax* ] //DEFAULT = 1**
**//MINCOUNT is the minimum number of fields in the group which MUST have a value.**
**[ MAXCOUNT MAX | *aIntegerFrom0toMax* ] //DEFAULT = MAX**
**//"MAX" is shorthand for a count of total number of fields in the group**
**DEFINITION**
***aFieldGroupName* = *aSelectionOfFields***
**ENDDEFINITION**
**;; //Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule // Marks the end of this rule**

**where**
***aSelectionOfFields* =**
**INLIST ( *aListOfFieldNames*) |**
**NOTINLIST ( *aListOfFieldNames*) |**
**PASS ( *aCondition*) |**
**FAIL ( *aCondition*) |**
**MATCH ( *aPattern*) [DEFINITION *aPattern* ENDDEFINITION] |**
**NOTMATCH ( *aPattern*) [DEFINITION *aPattern* ENDDEFINITION]**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Exactly 1 field in the following list **must have a value** if CommissionYN is "Yes" [which indicates the user's intent to provide "Commission To" information]:<br>• BuyerAgencyCompensation<br>• SubAgencyCompensation<br>• TransactionBrokerCompensation<br><br>**Alternate Statement:**<br>Exacltly 1 field in CommissionFieldGroup **must have a value** if CommissionYN is "Yes".<br><br>CommissionFieldGroup consists of the following fields:<br>• BuyerAgencyCompensation<br>• SubAgencyCompensation<br>• TransactionBrokerCompensation | REQUIRE_VALUE<br>FIELDGROUP CommissionFieldGroup<br>IF CommissionYN = "Y" ENDIF<br>MINCOUNT 1<br>MAXCOUNT 1<br>DEFINITION<br>CommissionFieldGroup =<br>INLIST (BuyerAgencyCompensation, SubAgencyCompensation, TransactionBrokerCompensation)<br>ENDDEFINITION<br>;;<br>_end_Rule |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Every field in the UnitFieldsFieldGroup must have a value for at least one Unit, in a Listing of PropertyType Multi-family.<br><br>UnitFieldsFieldGroup includes:<br>DiningAreaLength, DiningAreaWidth, KitchenLength, KitchenWidth, LivingAreaLength, LivingAreaWidth, MasterBedLength, MasterBedWidth, NumberOfBathsFull, NumberOfBathsHalf, NumberOfBeds, NumberOfUnits, SqFt, Lease. | REQUIRE_VALUE<br>FIELDGROUP<br>(UnitFieldsFieldGroup for at least 1 Unit for this Listing)<br>IF<br>(PropertyType = MultiFamily ) AND (0 Units in this Listing have values for all fields in the UnitFieldsFieldGroup)<br>ENDIF<br>MINCOUNT MAX<br>//All fields in group required<br>MAXCOUNT MAX<br>DEFINITION<br>UnitFieldsFieldGroup = INLIST<br>(DiningAreaLength, DiningAreaWidth, KitchenLength, KitchenWidth, LivingAreaLength, LivingAreaWidth, MasterBedLength, MasterBedWidth, NumberOfBathsFull, NumberOfBathsHalf, NumberOfBeds, NumberOfUnits, SqFt, Lease)<br>ENDDEFINITION<br>;;<br>_end_Rule |

## 14. REQUIRE_VALUE [3] - OBJECT

### DESCRIPTION

This rule specifies whether a Listing must contain specified Object(s) and the minimum number of each such object. Examples of listing Objects are: Bedroom, Bathroom, Kitchen, etc., i.e., items which are not thought of as a data field but rather as a "thing" – often with its own data fields.

### SYNTAX

**REQUIRE_VALUE**

**OBJECT**

**[ UNCONDITIONAL | IF *aCondition* ENDIF ]**

**$[ *aObjectName* [MINCOUNT *aIntegerGTzero*] ]$**

**[ , *aObjectName* [MINCOUNT *aIntegerGTzero*] ]...**

**//Repetition is comma separated, e.g.: " Bedroom MINCOUNT 2, Bathroom MINCOUNT 2"**

**//MINCOUNT = min instances required**

**;;** **//Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| A SFR Listing with ListPrice greater than $3,000,000 must have at least:<br>2 Bedrooms, 2 Bathrooms, and 1 Kitchen. | SCOPE (<br>**Name = SFR;**<br>**//Other scope details go here**<br>**//..**<br>)<br>REQUIRE_VALUE OBJECT<br>IF<br>ListPrice GREATERTHAN $3M<br>ENDIF<br>Bedroom MINCOUNT 2,<br>Bathroom MINCOUNT 2,<br>Kitchen MINCOUNT 1<br>;; //Two semicolons<br>_end_Rule<br>//Other rules in this scope<br>_end_Scope |

# 15. REQUIRE_EMPTY [1] - FIELD

## DESCRIPTION

This rule specifies whether a data field is required to be empty valued, i.e., must have no value.

## SYNTAX

**REQUIRE_EMPTY**
**FIELD *aFieldName***
**[ UNCONDITIONAL | IF *aCondition* ENDIF ]**

**;;** **//Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

### EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| ReserveListPrice **must be empty valued** if this is not an auction listing. | REQUIRE_EMPTY<br>FIELD ReserveListPrice<br>IF (Listing is not an Auction Listing) ENDIF<br>;;<br>_end_Rule |
| LockBoxSerialNumber **must be empty valued** if Listing Status equals "closed" or "expired". | REQUIRE_EMPTY<br>FIELD LockBoxSerialNumber<br>IF Status INLIST ("closed", "expired")<br>ENDIF<br>;;<br>_end_Rule |

## 16. REQUIRE_EMPTY [2] - FIELDGROUP

### DESCRIPTION

This rule specifies whether all fields in a specified group of fields must be empty, i.e., have no value.

### SYNTAX

**REQUIRE_EMPTY**
**FIELDGROUP *aFieldGroupName***
**[UNCONDITIONAL | IF *aCondition* ENDIF ]**
**DEFINITION**
***aFieldGroupName* = *aSelectionOfFields***
**ENDDEFINITION**

**;;** **//Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

**where**
***aSelectionOfFields*** **=**

**INLIST ( *aListOfFieldNames*)** |
**NOTINLIST ( *aListOfFieldNames*)** |
**PASS ( *aCondition*)** |
**FAIL ( *aCondition*)** |
**MATCH ( *aPattern*) [DEFINITION *aPattern* ENDDEFINITION]** |
**NOTMATCH ( *aPattern*) [DEFINITION *aPattern* ENDDEFINITION]**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Each field in AssociationDetailsFieldGroup **must be empty valued** if a listed property does not have an Association.<br><br>AssociationDetailsFieldGroup consists of the following fields:<br>• AssociationName<br>• AssociationPhone<br>• AssociationFee<br>• AssociationFeeFrequency | REQUIRE_EMPTY<br>FIELDGROUP<br>AssociationDetailsFieldGroup<br>IF AssociationYN = "N" ENDIF<br>// REBR best practice<br>//is to avoid referring to fields by<br>// their implementation based<br>// fieldnames such as<br>// AssociationYN, especially if that<br>// makes the rule unclear to the<br>// business reader.<br>**// BETTER:**<br>**// IF (Property does not have an**<br>**// Association)**<br>**// ENDIF**<br>DEFINITION<br>AssociationDetailsFieldGroup =<br>INLIST(AssociationName,<br>AssociationPhone, AssociationFee,<br>AssociationFeeFrequency)<br>ENDDEFINITION |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| | |

## 17. ALLOW_VALUE [1] - FIELD

### DESCRIPTION

This rule specifies whether a specified listing data field is Optional, i.e., it may have a value or it may be empty.

### SYNTAX

**ALLOW_VALUE**
**FIELD *aFieldName***
**[ UNCONDITIONAL | IF *condition* ENDIF ]**
**// There is no ~~Default~~ ~~[ DEFAULT *DefaultValue* ]~~**
**;;** **//Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

### EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| RoomSize may have a value | ALLOW_VALUE<br>FIELD RoomSize<br>;;<br>_end_Rule |
| State may have a value. | ALLOW_VALUE<br>FIELD State<br>;;<br>_end_Rule |
| AllowAVM may have a value. | ALLOW_VALUE<br>FIELD AllowAVM<br>;;<br>_end_Rule |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| GarageSpaces may have a value if GarageYN has a value of "Y". | ALLOW_VALUE<br>FIELD GarageSpaces<br>IF GarageYN = "Y" ENDIF<br>;;<br>_end_Rule |
| Disclaimer may have a value. | ALLOW_VALUE<br>FIELD Disclaimer<br>;;<br>_end_Rule |

## 18. ALLOW_VALUE [2] - FIELDGROUP

### DESCRIPTION

This rule specifies whether fields in a specified group of fields are Optional (may or may not have a value). A ~~min and~~ max count can be used to specify the number of fields which ~~must~~ may have a value.

### SYNTAX

**ALLOW_VALUE**
**FIELDGROUP** *aFieldGroupName*
**[ UNCONDITIONAL | IF** *aCondition* **ENDIF ]**
**//There is no MINCOUNT ~~[ MINCOUNT 1 | aIntegerFrom0toMax ] //DEFAULT = 1~~**
**[ MAXCOUNT MAX |** *aIntegerFrom0toMax* **] //DEFAULT = MAX**
**//"MAX" is shorthand for a count of total number of fields in the group**
**DEFINITION**
*aFieldGroupName* **=** *aSelectionOfFields*
**ENDDEFINITION**
**;; //Two semicolons**
**[***aRULE-LEVEL-DIRECTIVE* **] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[** *aENFORCEMENT* **] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[** *aTRIGGER* **] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule // Marks the end of this rule**

**where**
*aSelectionOfFields* **=**
**INLIST (** *aListOfFieldNames***)**

**NOTINLIST ( *aListOfFieldNames*)** |
**PASS ( *aCondition*)** |
**FAIL ( *aCondition*)** |
**MATCH ( *aPattern*) [DEFINITION *aPattern* ENDDEFINITION]** |
**NOTMATCH ( *aPattern*) [DEFINITION *aPattern* ENDDEFINITION]**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Exactly 1 field in the following list **may have a value** if CommissionYN is "Yes" [which indicates the user's intent to provide "Commission To" information]:<br>• BuyerAgencyCompensation<br>• SubAgencyCompensation<br>• TransactionBrokerCompensation<br><br>**Alternate Statement:**<br>Exacltly 1 field in CommissionFieldGroup **may have a value** if CommissionYN is "Yes".<br><br>CommissionFieldGroup consists of the following fields:<br>• BuyerAgencyCompensation<br>• SubAgencyCompensation<br>• TransactionBrokerCompensation | ALLOW_VALUE<br>FIELDGROUP CommissionFieldGroup<br>IF CommissionYN = "Y" ENDIF<br>MAXCOUNT 1<br>DEFINITION<br>CommissionFieldGroup =<br>INLIST (BuyerAgencyCompensation,<br>SubAgencyCompensation,<br>TransactionBrokerCompensation)<br>ENDDEFINITION<br>;;<br>_end_Rule |

# 19. ALLOW_VALUE [3] - OBJECT

## DESCRIPTION

This rule specifies whether specified Object(s) may optionally be included in a Listing, and a max count. Examples of listing Objects are: Bedroom, Bathroom, Kitchen, etc., i.e., items which are not thought of as a data field but rather as a "thing" – often with its own data fields.

## SYNTAX

**ALLOW_VALUE**
**OBJECT**
**[ UNCONDITIONAL | IF *aCondition* ENDIF ]**
**$[ *aObjectName* [MAXCOUNT *aIntegerGTzero*] ]$**

**[ , *aObjectName* [MAXCOUNT *aIntegerGTzero*] ]...**

**//Repetition is comma separated, e.g.: " Bedroom MINCOUNT 2, Bathroom MINCOUNT 2"**
**//MINCOUNT = min instances required**

**;; //Two semicolons**

**[*aRULE-LEVEL-DIRECTIVE* ] …**

**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**

**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**

**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**

**_end_Rule // Marks the end of this rule**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| A SFR Listing with ListPrice less than $100,000 may have at most:<br>2 Bedrooms, 2 Bathrooms, and 1 Kitchen. | SCOPE (<br>**Name = SFR;**<br>**//Other scope details go here**<br>**//..**<br>)<br>ALLOW_VALUE OBJECT<br>IF<br>ListPrice LESSTHAN $100K<br>ENDIF<br>Bedroom MAXCOUNT 2,<br>Bathroom MAXCOUNT 2,<br>Kitchen MAXCOUNT 1<br>;; //Two semicolons<br>_end_Rule<br>//Other rules in this scope<br>_end_Scope |

# 20. DERIVE_VALUE [1] - SINGLE FIELD (COMPUTATION)

## DESCRIPTION

This rule specifies how a data field value is computed (e.g., Days On Market)

## SYNTAX

**DERIVE_VALUE**
**FIELD *aFieldName***
**$[**
**COMPUTATION *aComputationName* | *aComputationFormula***
**[ UNCONDITIONAL | IF *condition* ENDIF ]**
**DEFINITION *aComputationInstruction* ENDDEFINITION**
**]$ …**
**//There can be multiple computations based on different conditions**
**;; //Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule // Marks the end of this rule**

Where
***aComputationName*** = Text. Name of the computation to be used.

***aComputationFormula*** = Freeform text. For example: ListPrice *0.85

***aComputationInstruction*** = Freeform text detailing the computation. For example: TotalBathCalc = (TotalBathrooms = X.Y, where X=BathroomsFull, Y=BathroomsHalf)

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| TotalBathrooms **must be computed as** x+y, where x=BathroomsFull, y=BathroomsHalf. Example: If BathroomsFull = 3, BathroomsHalf = 2, then TotalBathrooms = 3+2. | DERIVE_VALUE<br>FIELD TotalBathrooms<br>COMPUTATION TotalBathCalc<br>DEFINITION<br>TotalBathCalc = (TotalBathrooms = X.Y, where X=BathroomsFull, Y=BathroomsHalf)<br>ENDDEFINITION<br>;;<br>_end_Rule |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| LotSizeSqFt **must be computed** as LotSizeAcres times 43560. | DERIVE_VALUE<br>FIELD LotSizeSqFt<br>COMPUTATION LotSizeSqFtCalc<br>DEFINITION<br>LotSizeSqFtCalc = (LotSizeAcres*43560)<br>ENDDEFINITION<br>;;<br>_end_Rule |

## 21. DERIVE_VALUE [2] - SINGLE FIELD (SYSTEM)

### DESCRIPTION

This rule specifies how a data field value is substituted by a System field.

### SYNTAX

**DERIVE_VALUE**

**FIELD** ***aFieldName***

**$[ SYSTEMVALUE** ***aSystemField*** **[ UNCONDITIONAL | IF *condition* ENDIF ] ]$ …**

**//Multiple system field sources allowed based on different conditions**

**;; //Two semicolons**

**[*aRULE-LEVEL-DIRECTIVE* ] …**

**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**

**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**

**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**

**_end_Rule // Marks the end of this rule**

### EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| DateCreated **must be assigned** system value of CurrentDateTime | DERIVE_VALUE<br>FIELD DateCreated<br>SYSTEMVALUE CurrentDateTime<br>;;<br>_end_Rule |

# 22. DERIVE_VALUE [3] - SINGLE FIELD (AUTOPOP)

## DESCRIPTION

This rule specifies how a data field is Autopopulated, and what is the source of the autopop data.

## SYNTAX

**DERIVE_VALUE**
**FIELD** ***aFieldName***
**$[ AUTOPOP** ***AutopopSourceNam***e **[ UNCONDITIONAL | IF** ***aCondition*** **ENDIF ] ]$ …**
**//Multiple Autopop sources allowed based on different conditions**
**[DEFINITION**
***AutopopSourceDescription***
**ENDDEFINITION**
**]…**
**;;** **//Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| ParcelNumber **must be auto populated**. CoreLogic is the data source If State is "NY". LPS is the data source if State is "NJ". CRS is the data source if State is "PA". | DERIVE_VALUE<br>FILED ParcelNumber<br>AUTOPOP CoreLogic IF State="NY" ENDIF<br>AUTOPOP LPS IF State="NJ" ENDIF<br>AUTOPOP CRS IF State="PA" ENDIF<br>;;<br>_end_Rule |

## 23. DERIVE_VALUE [4] - GROUP OF FIELDS (AUTOPOP)

### DESCRIPTION

This rule specifies how a group of fields is Autopopulated, and what is the source of the autopop data.

### SYNTAX

**DERIVE_VALUE**
**FIELDGROUP** *aFieldGroupName*
**$[**
**AUTOPOP** *AutopopSourceNam*e
**[ UNCONDITIONAL | IF** *aCondition* **ENDIF ]**
**]$ …**
**//Multiple Autopop sources allowed based on different conditions**
**[ DEFINITION** *aFieldGroupName* **=** *aSelectionOfFields* **ENDDEFINITION ]**
**[DEFINITION** *AutopopSourceDescription* **ENDDEFINITION ]…**
**;;** **//Two semicolons**
**[***aRULE-LEVEL-DIRECTIVE* **] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[** ***aENFORCEMENT*** **] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[** ***aTRIGGER*** **] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

**where**
*aSelectionOfFields* **=**
**INLIST (** *aListOfFieldNames***) |**
**NOTINLIST (** *aListOfFieldNames***) |**
**PASS (** *aCondition***) |**
**FAIL (** *aCondition***) |**
**MATCH (** *aPattern***) [DEFINITION** *aPattern* **ENDDEFINITION] |**
**NOTMATCH (** *aPattern***) [DEFINITION** *aPattern* **ENDDEFINITION]**

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Each field in TaxFieldGroup **must be must be auto populated.**<br>BKFS is the data source.<br>TaxFieldGroup consists of the following fields:<br>• APN<br>• StreetNumber<br>• StreetName<br>• County<br>• City<br>• State<br>• Zip<br>• Zoning<br>• Beds<br>• Baths<br>• SquareFeet | DERIVE_VALUE<br>FIELDGROUP TaxFieldGroup<br>AUTOPOP BKFS<br>DEFINITION<br>TaxFieldGroup =<br>INLIST(APN, StreetNumber,<br>StreetName, County,<br>City, State, Zip, Zoning,<br>Beds, Baths, SquareFeet)<br>ENDDEFINITION<br>DEFINITION<br>BKFS = (A description of BKFS,<br>including required connection<br>details, etc.)<br>ENDEFINITION<br>;;<br>_end_Rule |

## 24. AN IMPORTANT OBSERVATION ABOUT THE "CONSTRAIN_..." GROUP OF RULES

The "Constrain_..." group of rules includes the following rules:

- All Constrain_Date_Time rules
- All Constrain_Value rules
- All Constrain_Media rules

**IMPORTANT :**
**The constraint expressed in "Constrain_..." rules is NOT evaluated if the relevant field or object is empty valued.**

Example:

Rule #1: "List Price must be less than $3 Million".
Situation 1: List Price has a value = $4 million.
In this situation, Rule #1 is executed, and the result is that the rule failed.
Situation 2: List Price is empty valued.
In this case, Rule #1 is NOT executed.

If the intent is to ensure that List Price has a value, and that the value is less than $3M, then two separate rules are needed:

1. List Price must have a non-empty value
2. List Price must be less than $3 Million

The two rules can be expressed in REBR as:

```
//Rule #1: List Price must have a non-empty value
REQUIRE_VALUE FIELD ListPrice ;;
_end_Rule
//Fails if List Price is empty valued.
//Rule #2: List Price must be less than $3 Million
CONSTRAIN_VALUE FIELD ListPrice
BYVALUE LESSTHAN $3 Million;;
_end_Rule
```

## 25. CONSTRAIN_DATE_TIME [1] - SPECIFIC DATETIME, WITH OR WITHOUT OFFSET

### DESCRIPTION

This rule specifies the constraint that must be met by a Date or Datetime field. The constraint is in terms of a specific date/time, with or without offset.

### SYNTAX

**CONSTRAIN _DATE_TIME**

**FIELD**

**$[** *aDateFieldName* **|** *aDateTimeFieldName* **]$**

**[** **UNCONDITIONAL** **|** **IF** *aCondition* **ENDIF** **]**

**$[**

**EQUAL | NOTEQUAL |**

**BEFORE | ONORBEFORE |**

**AFTER | ONORAFTER**

**]$**

**$[** *aDate* **|** *aDateTime* **|**

**[New | Current | Prior]** *aDateFieldName* **|**

**[New | Current | Prior]** *aDateTimeFieldName*

**]$**

**[ OFFSET** *aDateOffset* **|** *aDateTimeOffset* **]**

**;;** **//Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

where
***aDateTimeOffset* =**
***aDateOffset aTimeOffset***

***aDateOffSet* = $[ + | - ]$ *aNumberGEzero* $[ DAYS | WEEKS | MONTHS | YEARS ]$**

***aTimeOffset* = $[ + | - ]$ *aNumberGEzero* $[ MILLISECS | SECONDS | MINUTES | HOURS ]$**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| ContractStatusChangeDate **must be** a date on or after ListingContractDate. | CONSTRAIN _DATE_TIME<br>FIELD ContractStatusChangeDate<br>ONORAFTER<br>ListingContractDate<br>;;<br>_end_Rule |
| ListingContractDate **must be** on or before today's date.<br><br>Analysis: This rule says that ListingContractDate must not be in the future. | CONSTRAIN_DATE_TIME<br>FIELD ListingContractDate<br>ONORBEFORE TODAY<br>;;<br>_end_Rule |
| YearBuilt **must be** on or after 1700. | CONSTRAIN_DATE_TIME<br>FIELD YearBuilt ONORAFTER 1/1/1700<br>;;<br>_end_Rule |
| Expiration Date **must be** no later than 1 year after Listing Contract Date. | CONSTRAIN_DATE_TIME<br>FIELD ExpirationDate<br>ONORBEFORE<br>(ListingContractDate + 1 YEAR)<br>;;<br>_end_Rule |

## 26. CONSTRAIN_DATE_TIME [2] - DATETIME RANGE, WITH OR WITHOUT OFFSET

### DESCRIPTION

This rule specifies the constraint that must be met by a Date or Datetime field. The constraint is in terms of a range of dates/times, with or without offset. NOTE: This syntax **requires both** Lowrange and Highrange to be specified. If there is only a single value, use the previous syntax.

### SYNTAX

**CONSTRAIN _DATE_TIME**

**FIELD**

**$[** *aDateFieldName* **|** *aDateTimeFieldName* **]$**

**[ UNCONDITIONAL | IF** *aCondition* **ENDIF ]**

**LOWRANGE**

**$[ ONORAFTER | AFTER ]$**

**$[** *aDate* **|** *aDateTime* **|**

**[New | Current | Prior]** *aDateFieldName* **|**

**[New | Current | Prior]** *aDateTimeFieldName*

**]$**

**[ OFFSET** *aDateOffset* **|** *aDateTimeOffset* **]**

**HIGHRANGE**

**$[ ONORBEFORE | BEFORE ]$**

**$[** *aDate* **|** *aDateTime* **|**

**[New | Current | Prior]** *aDateFieldName* **|**

**[New | Current | Prior]** *aDateTimeFieldName*

**]$**

**[ OFFSET** *aDateOffset* **|** *aDateTimeOffset* **]**

**;;** //Two semicolons

**[***aRULE-LEVEL-DIRECTIVE* **] …**

//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level

**[** *aENFORCEMENT* **] …** // Zero or more Enforcements. If omitted, default(s) must be provided.

**[** *aTRIGGER* **] …** // Zero or more Triggers. If omitted, default(s) must be provided.

**_end_Rule** // Marks the end of this rule

where

***aDateTimeOffset*** **=**

***aDateOffset*** ***aTimeOffset***

***aDateOffSet*** **= $[ + | - ]$** ***aNumberGEzero***

**$[ DAYS / WEEKS | MONTHS | YEARS ]$**

***aTimeOffset*****=$[ + | - ]$** ***aNumberGEzero***

**$[ MILLISECS | SECONDS | MINUTES | HOURS ]$**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| ExpirationDate **must be** after ListingContractDate and on or before ListingContractDate plus 365 days. | CONSTRAIN_DATE_TIME<br>FILED ExpirationDate<br>UNCONDITIONAL<br>LOWRANGE<br>AFTER ListingContractDate<br>HIGHRANGE ONORBEFORE<br>ListingContractDate + 365 days<br>;;<br>_end_Rule |

# 27. CONSTRAIN_VALUE [1] – SPECIFIC VALUE, WITH OR WITHOUT VALUE OFFSET

## DESCRIPTION

This rule specifies the constraint that must be met by a data field. The constraint is in terms of a specific data value, with or without offset.

## SYNTAX

**CONSTRAIN_VALUE**

**FIELD** ***aFieldName***

**[ UNCONDITIONAL | IF** ***aCondition*** **ENDIF ]**

**BYVALUE**

**$[**

**EQUAL | NOTEQUAL |**
**LESSTHANOREQUAL | LESSTHAN |**
**GREATERTHANOREQUAL | GREATERTHAN**
**]$**

**$[** ***aValue*** **|** ***aComputationInstruction*** **|**
**[New | Current | Prior]** ***aFieldName***
**]$**
**[ OFFSET** ***aValueOffset*** **]**
**;;** **//Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

**where**
***aValueOffset =***
**$[ + | - ]$** ***aValue | aFieldName | aComputationInstruction***

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| ListPriceLow **must be** greater than 85% of the value of ListPrice. | CONSTRAIN_VALUE<br>FIELD ListPriceLow<br>BYVALUE<br>GREATERTHAN ListPrice*.85<br>;;<br>_end_Rule |
| ParkingTotal **must be** greater than or equal to the value of RentedParkingSpaces | CONSTRAIN_VALUE<br>FIELD ParkingTotal<br>BYVALUE<br>GREATERTHANOREQUAL<br>RentedParkingSpaces<br>;;<br>_end_Rule |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| ClosePrice **must be** greater than or equal to ReserveListPrice if Auction is equal to "Yes". | CONSTRAIN_VALUE<br>FIELD ClosePrice<br>IF Auction = "Yes" ENDIF<br>BYVALUE<br>GREATERTHANOREQUAL<br>ReserveListPrice<br>;;<br>_end_Rule |
| Firm of Additional ListAgent must be the same as the Firm of the Primary List Agent. | CONSTRAIN_VALUE<br>FIED (Firm of Additional ListAgent)<br>BYVALUE<br>EQUAL (Firm of Primary ListAgent)<br>;;<br>_end_Rule |

## 28. CONSTRAIN_VALUE [2] – VALUE RANGE, WITH OR WITHOUT VALUE OFFSET

### DESCRIPTION

This rule specifies the constraint that must be met by a data field. The constraint is in terms of a range of values, with or without offset. NOTE: This syntax **requires** both Lowrange and Highrange to be specified. If there is only a single value, use the previous syntax.

### SYNTAX

**CONSTRAIN_VALUE**

**FIELD** *aFieldName*

**[ UNCONDITIONAL | IF** *aCondition* **ENDIF ]**

**LOWRANGE**

**$[ GREATERTHANOREQUAL | GREATERTHAN ]$**

**$[** *aValue* **|** *aComputationInstruction* **|**

**[New | Current | Prior]** *aFieldName*

**]$**

**[ OFFSET** *aValueOffset* **]**

**HIGHRANGE**

**$[ LESSTHANOREQUAL | LESSTHAN ]$**

**$[** *aValue* **|** *aComputationInstruction* **|**

**[New | Current | Prior]** *aFieldName*

**]$**

**[ OFFSET** *aValueOffset* **]**

**;; //Two semicolons**

**[*aRULE-LEVEL-DIRECTIVE* ] …**

**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**

**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**

**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**

**_end_Rule // Marks the end of this rule**

**where**

***aValueOffset =***

**$[ + | - ]$** *aValue* **|** *aFieldName* **|** *aComputationInstruction*

### EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| ListPrice **must be** greater than or equal to 1, and less than or equal to 50,000,000 | CONSTRAIN_VALUE<br>FIELD ListPrice<br>LOWRANGE<br>GREATERTHANOREQUAL 1<br>HIGHRANGE LESSTHANOREQUAL 50<br>Million<br>;;<br>_end_Rule |

## 29. CONSTRAIN_VALUE [3] – FILTER BY LIST, CONDITION OR PATTERN

### DESCRIPTION

This rule specifies the constraint that must be met by a data field. The constraint is in terms of values filtered by an inclusion or exclusion list, pass or fail conditions, or match or not match pattern(s).

### SYNTAX

**CONSTRAIN_VALUE**
**FIELD** ***aFieldName***
**[** **UNCONDITIONAL** **|** **IF** ***aCondition*** **ENDIF** **]**

**FILTER**
**$[**
**INLIST** **(*aListOfValues*)** **|**
**NOTINLIST** **(*aListOfValues*)** **|**
**PASS** **(** ***aCondition*** **)** **|**
**FAIL** **(** ***aCondition*** **)** **|**
**MATCH** **(** ***aPattern*** **)** **[DEFINITION** ***aPattern*** **ENDDEFINITION]** **|**
**NOTMATCH** **(*aPattern*)** **[DEFINITION** ***aPattern*** **ENDDEFINITION]**
**]$**
**;;** **//Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE*** **] …**

**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

where
***aListOfValues*** = ***aValue*** **[** , ***aValue*****]…**
**is a comma separated list of values with at least one value.**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| MemberEmail value **must match** the standard pattern for a single email address. | CONSTRAIN_VALUE<br>FIELD MemberEmail<br>FILTER<br>MATCH emailPatternRegExp<br>DEFINITION emailPatternRegExp =<br>"(?:[a-z0-9!#$%&'*+/=?^_`{\|}~-<br>]+(?:\.[a-z0-9!#$%&'*+/=?^_`{\|}~-<br>]+)*\|"(?:[\x01-\x08\x0b\x0c\x0e-<br>\x1f\x21\x23-\x5b\x5d-\x7f]\|\\[\x01-<br>\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-<br>z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-<br>9](?:[a-z0-9-]*[a-z0-9])?\|\[(?:(?:25[0-<br>5]\|2[0-4][0-9]\|[01]?[0-9][0-<br>9]?)\.){3}(?:25[0-5]\|2[0-4][0-<br>9]\|[01]?[0-9][0-9]?\|[a-z0-9-]*[a-z0-<br>9]:(?:[\x01-\x08\x0b\x0c\x0e-<br>\x1f\x21-\x5a\x53-\x7f]\|\\[\x01-<br>\x09\x0b\x0c\x0e-\x7f])+)\])"<br>ENDDEFINITION<br>;;<br>_end_Rule |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Status of an Active Listing of Residential Property Type **may only change** to one of the following: "Active", "Cancelled", "Extended", "Under Agreement", "Temporarily Withdrawn". MLS Enforcement: Staff or SuperUser may override.<br><br>**Analysis**: The above rule is applicable to residential listings whose **current** status is "Active". The rule limits the choice of **new** status value for the listing.<br><br>IMPORTANT: Note the use of keyword CURRENT to indicate that the rule applies to a **change** in value. | CONSTRAIN_VALUE<br>FIELD Status<br>IF<br>**CURRENT** Status = "Active" AND ListingPropertyType = "Residential"<br>ENDIF<br>FILTER<br>INLIST(<br>"Active","Cancelled","Extended","Under Agreement","Temporarily Withdrawn"<br>)<br>;;<br>RULELEVEL ENFORCEMENT AUGMENTS SCOPELEVEL<br>;;<br>ENFORCEMENT OVERRIDE<br>ROLE INLIST ("MLS STAFF", "SUPERUSER")<br>;;<br>// Staff can move an active listing to<br>// additional statuses like "Deleted" or<br>//"New" or straight to "Sold".<br>_end_Rule |

## 30. CONSTRAIN_VALUE [4] - PROHIBITED VALUES

### DESCRIPTION

This rule specifies values which are prohibited for a data field. Prohibited values are specified in terms of an inclusion or exclusion list, values filtered by pass or fail conditions, or values filtered by pattern(s).

Note: Best practice – If the intent is to prohibit the use of certain values, then use this form of the Constrain_Value rule. Do not use the "Constrain_Value Filter" form instead.

### SYNTAX

**CONSTRAIN_VALUE**
**FIELD** ***aFieldName***
**[ UNCONDITIONAL | IF** ***aCondition*** **ENDIF ]**
**PROHIBIT**
**$[**

**INLIST ( *aListOfValues*)** **|**
**NOTINLIST ( *aListOfValues* )** **|**
**PASS (*aCondition* )** **|**
**FAIL ( *aCondition* )** **|**
**MATCH ( *aPattern*)**
**[DEFINITION *aPattern* ENDDEFINITION]** **|**
**NOTMATCH (*aPattern* )**
**[DEFINITION *aPattern* ENDDEFINITION]**
**]$**
**;; //Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule // Marks the end of this rule**

where

***aListOfValues* = *aValue* [ , *aValue*]…**
**is a comma separated list of values with at least one value**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| PublicRemarks must not include any of the following terms: "curse1", "curse2". | CONSTRAIN_VALUE<br>FIELD PublicRemarks<br>PROHIBIT INLIST=("curse1", "curse2")<br>;;<br>_end_Rule |
| PublicRemarks must not include a website URL. | CONSTRAIN_VALUE<br>FIELD PublicRemarks<br>PROHIBIT MATCH (URL)<br>DEFINITION<br>URL = (Insert RegExp pattern for URL here)<br>ENDDEFINITION<br>;;<br>_end_Rule |

# 31. CONSTRAIN_VALUE [5] – SINGLE LOOKUP LISTS

## DESCRIPTION

This rule applies to fields which take a value from a single-choice lookup list. The rule specifies which lookup list will be used, default value(s) if any, and what choices are or are not permitted.

## SYNTAX

**CONSTRAIN_VALUE**

**FIELD** *aFieldName*

**[ UNCONDITIONAL | IF** *aCondition* **ENDIF ]**

**//Single Lookup**

**LOOKUP** *aLookupListName*

**[ CHOOSEDEFAULT (***aValue***) ]** **// Default value(s) must be in the lookup list.**

**[ CHOOSE** *aValue* **[UNCONDITIONAL | IF** *aCondition* **ENDIF] ]...**

**[ DONOTCHOOSE** *aValue* **[UNCONDITIONAL | IF** *aCondition* **ENDIF] ]...**

**[ DEFINITION** *aLookupListDescription* **ENDDEFINITION ]**

**// A lookup list definition is a free form**
**// text description of the lookuo list**
**;; //Two semicolons**
**[***aRULE-LEVEL-DIRECTIVE* **] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[** ***aENFORCEMENT*** **] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[** ***aTRIGGER*** **] … // Zero or more Triggers. If omitted, default(s) must be provided.**

**_end_Rule** **// Marks the end of this rule**

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| LockBoxType must be chosen from single-lookup list "LockBoxType". There is no default value. | CONSTRAIN_VALUE<br>FIELD LockBoxType<br>LOOKUP LockBoxType<br>;;<br>_end_Rule |
| LotSizeUnits must be chosen from single-lookup list "AreaUnits". Default=Acre | CONSTRAIN_VALUE<br>FIELD LotSizeUnits<br>LOOKUP AreaUnits<br>CHOOSEDEFAULT (Acre)<br>;;<br>_end_Rule |

## 32. CONSTRAIN_VALUE [6] – MULTI-LOOKUP LISTS

### DESCRIPTION

This rule applies to fields which take one or more values from a multi-choice lookup list. The rule specifies which multi-lookup list will be used, default value(s) if any, the min and max number of choices, and multi-lookup choices that are or are not permitted.

### SYNTAX

**CONSTRAIN_VALUE**

**FIELD** *aFieldName*

**[ UNCONDITIONAL | IF** *aCondition* **ENDIF ]**

**//Multi lookup**

**MULTILOOKUP** *aMultiLookupListName*

**[ CHOOSEMIN 1 |** *aIntegerFrom0ToMAX* **]**

**[ CHOOSEMAX MAX |** *aIntegerFrom0ToMAX* **]**

**// Multi default Min = 1: choose at least 1.**
**// Multi default Max =MAX: choose as many as you want**
**// Default value(s) must be in the lookup list.**

**[ CHOOSEDEFAULT (***aListOfValues***) ]**

**// More than 1 default value allowed only if multi-lookup**

**[ CHOOSE** *aSelectionOfValues* **[UNCONDITIONAL | IF** *aCondition* **ENDIF] ]...**

**[ DONOTCHOOSE** *aSelectionOfValues* **[UNCONDITIONAL | IF** *aCondition* **ENDIF] ]...**

**[ DEFINITION** *aMultiLookupListDescription* **ENDDEFINITION ]**

**// Each lookup list definition is a free form**
**// text description of the lookuo list**
**;; //Two semicolons**
**[***aRULE-LEVEL-DIRECTIVE* **] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[** *aENFORCEMENT* **] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[** *aTRIGGER* **] … // Zero or more Triggers. If omitted, default(s) must be provided.**

**_end_Rule** **// Marks the end of this rule**

Where

*aListOfValues* = *aValue* **[ ,** *aValue***]...**

**is a comma separated list of values with at least one value**

***aSelectionOfValues*** **=**

**INLIST ( *aListOfValues*) |**
**NOTINLIST (*aListOfValues*) |**
**PASS (*aCondition* ) |**
**FAIL ( *aCondition* ) |**
**MATCH ( *aPattern*) [DEFINITION *aPattern* ENDDEFINITION] |**
**NOTMATCH (*aPattern* ) [DEFINITION *aPattern* ENDDEFINITION]**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| AgriculturalWater must have at least 0 values and at most 8 values, each selected from multi-lookup list AgriculturalWater_SF | CONSTRAIN_VALUE<br>FIELD AgriculturalWater<br>MULTILOOKUP AgriculturalWater_SF<br>CHOOSEMIN 0<br>CHOOSEMAX 8<br>;;<br>_end_Rule |
| CommonWalls must have at least one and at most 3 values, each selected from multi-lookup list CommonWallsLookup.<br><br>CommonWalls must be "NoCommonWalls" if PropertyAttachedYN is "N"<br><br>CommonWalls must not be "NoCommonWalls" if PropertyAttachedYN is "Y" | CONSTRAIN_VALUE<br>FIELD CommonWalls<br>MULTILOOKUP CommonWallsLookup<br>CHOOSEMIN 1<br>CHOOSEMAX 3<br>CHOOSE INLIST ("NoCommonWalls")<br>IF PropertyAttachedYN = "N" ENDIF<br>DONOTCHOOSE INLIST ("NoCommonWalls")<br>IF PropertyAttachedYN = "Y" ENDIF<br>;;<br>_end_Rule |
| BasementFeatures values must belong to multi-lookup list ListOfBasementFeatures such that exactly one of the following is true:<br>1. BasementFeatures value is in the sublist: "None". (A single value)<br>2. BasementFeatures values are in the sublist which excludes: "None". (1 or more values) | CONSTRAIN_VALUE<br>FIELD BasementFeatures<br>UNCONDITIONAL<br>MULTILOOKUP ListOfBasementFeatures<br>DONOTCHOSE INLIST ("NONE")<br>IF (Any value other than "None" is chosen)<br>ENDIF<br>DONOTCHOOSE INLIST (All values except "NONE")<br>IF ("None" is chosen)<br>ENDIF<br>;;<br>_end_Rule |

## 33. CONSTRAIN_MEDIA

### DESCRIPTION

Specifies what media can be accepted – file counts, types and size; Image name and size.

### SYNTAX

**CONSTRAIN_MEDIA** *aMediaType*

**[ UNCONDITIONAL | IF** *aCondition* **ENDIF ]**
**MEDIAFILETYPE =**
**INLIST (***aListOfMediaFileTypes***) ;**
**[ MEDIAFILECOUNT** //Max or Min, or both
**$[ MAX =** *aIntegerGEzero* **]$ |**
**$[ MIN =** *aIntegerGEzero* **]$ |**
**$[ MAX =** *aIntegerGEzero* **MIN =** *aIntegerGEzero* **]$**
**;**
**]**
**[ MAXFILESIZE =** *aFileSize* **$[Bytes | KBytes | MB | GB ]$ ; ]**

**[**
**IMAGESIZENAME =** *ImageSizeName* **;**
**IMAGESIZEUNIT =**
**$[Pixel | inch | cm | mm]$ ;**
**IMAGEWIDTH**
**MAX =** *MaxWidth*
**MIN =** *MinWidth* **;**
**IMAGEHEIGHT**
**MAX =** *MaxHeight*
**MIN =** *MinHeight* **;**
**]...** //Can be repeated. For example, for ImageSizeName = Thumbnail, Small and Large sizes.
**; ;** //Two semicolons
**[***aRULE-LEVEL-DIRECTIVE* **] …**
//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level

**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule // Marks the end of this rule**

Where
***aListOfMediaFileTypes* = *aMediaFileType* [ , *aMEdiaFileType* ] …**

is a list of comma separated media file types.

***aMediaFileType*** = a filetype used for media files such as .jpg, .png, .mov, .mp3, .mp4, etc.

### EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Media in the category of "Photo" **must be** of MediaType "JPG" or "PNG". Filesize **must be** less than or equal to 20MB. Thumbnail image width **must be** 100 and height **must be** 30. Small image width **must be** 300 and height **must be** 200. Total number of Photos **must be** less than or equal to 50. | **CONSTRAIN_MEDIA** Photo<br>MediaFileType= INLIST("JPG","PNG");<br>MEDIAFILECOUNT MAX = 50;<br>MAXFILESIZE = 20 MB;<br>ImageSizeUnit=Pixel;<br><br>**ImageSizeName** = **Thumbnail**;<br>ImageWidth MAX=100 MIN=100;<br>ImageHeight= MAX=30 MIN=30;<br><br>**ImageSizeName**= **Small**;<br>ImageWidth MAX=300 MIN=300;<br>ImageHeight= MAX=200 MIN=200;<br>;;<br>**_end_Rule** |

## 34. ALLOW_DELETE_LISTING

### DESCRIPTION

A system may allow a listing to be effectively deleted under certain circumstances, by certain user roles.

Note: implementation may be to physically delete the listing and associated records from the database but more often it is to flag the listing in some way (e.g. a listing status of "deleted") so it no longer shows up in any display.

### SYNTAX

**ALLOW_DELETE_LISTING**

**[ UNCONDITIONAL | IF *aCondition* ENDIF ]**
**LISTINGSTATUS**
**$[**
**ALL | INLIST (*aListOfStatuses*) |**
**NOTINLIST (*aListOfStatuses*)**
**]$**
**USERROLE $[**
**ALL | INLIST (*aListOfUserRoles*) |**
**NOTINLIST (*aListOfUserRoles*)**
**]$**
**;; //Two semicolons**
**[*aRULE-LEVEL-DIRECTIVE* ] …**
**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**
**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**
**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule // Marks the end of this rule**

Where
***aListOfStatuses = aListingStatus [ , aListingStatus ] …***

is a list of comma separated listing statuses.

***aListOfUserRoles = aUserRole [ , aUserRole ] …***

is a list of comma separated user roles.

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| An Active listing may be deleted by a user with user role of MLS Staff. | ALLOW_DELETE_LISTING<br>LISTINGSTATUS NOTINLIST ("Active")<br>USERROLE INLIST ("MLS Staff")<br>;;<br>_end_Rule |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| A listing with a status that is not "Partial" **may only** be deleted by a user with one of the following roles: Listing Owner, MLS Staff, Super User. | ALLOW_DELETE_LISTING<br>LISTINGSTATUS NOTINLIST ("Partial")<br>USERROLE INLIST ("LISTING OWNER", "MLS Staff", "Super User")<br>;;<br>_end_Rule |

## 35. WATERMARK_PHOTO

### DESCRIPTION

This rule specifies the conditions under which the system will watermark a photo, using an image file and/or text, the location and opacity for the watermark, and font specifications for text watermark.

### SYNTAX

**WATERMARK_PHOTO**

**[ UNCONDITIONAL | IF *aCondition* ENDIF ]**

**$[**

**[*aIMAGEMARK*] | //Image used as watermark**

**[*aTEXTMARK*] | //Text used as watermark**

**[*aIMAGEMARK* *aTEXTMARK*] //Image and Text used as watermark**

**]$**

**;; //Two semicolons**

**[*aRULE-LEVEL-DIRECTIVE* ] …**

**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**

**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**

**[ *aTRIGGER* ] … // Zero or more Triggers. If omitted, default(s) must be provided.**

**_end_Rule // Marks the end of this rule**

**Where:**

***aIMAGEMARK =***

**IMAGEMARK //Image used as watermark**

**(**

**FILENAME = *filename ;***

**DISPLAYLOCATION** = **[ RANDOMIZE ]** **INLIST ( *aDisplayLocationList*)** **;**
**OPACITY % =** ***aInteger0to100***
**)**

***aTEXTMARK =***

**TEXTMARK** //Text used as watermark
**(**
**TEXT=*someText*** ;
**DISPLAYLOCATION** = **[ RANDOMIZE ]** **INLIST ( *aDisplayLocationList*)** **;**
**FONT= *aFont*** **;**
**FONTSIZE = *aFontSize*** **;**
**FONTCOLOR = *aFontColor*** **;**
**OPACITY % =** ***aInteger0to100***
**)**

***aDisplayLocationList* = *aDisplayLocation* [ , *aDisplayLocation*] …**
**Is a comma separated list of display locations.**
***aDisplayLocation*** = **$[ TopRight | TopLeft | BottomRight | BottomLeft]$**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Photos must be visibly watermarked if all of the following are true:<br>1. Image size is larger than 400x300 pixels (W x H)<br>2. image size is smaller than 3000x2000 pixels (W x H)<br><br>The watermark image filename is "MLSlogo.jpg"<br>Display Location is bottom-right<br>Opacity is 50%<br><br>The text "(C) [Current Year] [ListingCopyrightHolder]" is included.<br>Display Location is bottom-right<br>Font = Arial<br>Font-size = 12 point<br>Font-color =black<br>Opacity is 100% | **WATERMARK_PHOTO**<br>IF<br>(Photo size is GT 400x300 pixels [WxH]) AND<br>(Photo size is LT 3000x2000 pixels [WxH])<br>ENDIF<br>**IMAGEMARK** (<br>FILEENAME="MLSlogo.jpg"<br>DISPLAYLOCATION ="BottomRight"<br>OPACITY="50%"<br>)<br>**TEXTMARK** (<br>TEXT="©[Current Year] [CopyrightHolder]"<br>DISPLAYLOCATION ="BottomRight"<br>FONT=Arial<br>FONTSIZE=12pt<br>FONTCOLOR=Black<br>OPACITY="100%"<br>)<br>;;<br>**_end_Rule** |
| Photos must be visibly watermarked if all of the following are true: | IMAGE_WATERMARK<br>IF |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| 1. Photo is Virtually Staged<br>2. Image size is larger than 400x300 pixels (W x H)<br>3. image size is smaller than 3000x2000 pixels (W x H)<br>4. BrokerWatermarkPreference="SAMPLE MLS"<br><br>Watermark Text = "Virtually Staged<br>Display Location = Randomized in one of the following locations: top-right, top-left.<br>Font is Arial<br>Font-size is 12 point<br>Font-color is black<br>Opacity is 100% | (Photo is Virtually Staged) AND<br>(Photo size is GT 400x300 pixels [WxH]) AND<br>(Photo size is LT 3000x2000 pixels [WxH])<br>AND( BrokerWatermarkPreference=<br>"SAMPLE MLS")<br><br>**TEXTMARK** (<br>TEXT="Virtually Staged"<br>DISPLAYLOCATION = RANDOMIZE<br>INLIST (TopRight, TopLeft)<br>FONT=Arial<br>FONTSIZE=12pt<br>FONTCOLOR=Black<br>OPACITY="100%"<br>)<br>;;<br>_end_Rule |

## 36. AUTO_UPDATE LISTINGSTATUS

### DESCRIPTION

This rule specifies the conditions under which the system will automatically update the Listing Status. The trigger clause specifies a trigger based on a date or time, a condition, an event, or a user action.

### SYNTAX

**AUTO_UPDATE**

**LISTINGSTATUS**

**TO** ***aToListingStatus***

**//This is the "change to" status**

**[ UNCONDITIONAL | IF *aCondition* ENDIF ]**

**// IF condition identifies the set of Listings**

**// whose status is to be changed.**

**;; //Two semicolons**

**[*aRULE-LEVEL-DIRECTIVE* ] …**

**//Up to two. Indicating if rule-level enforcement and trigger replaces or augments scope-level**

**[ *aENFORCEMENT* ] … // Zero or more Enforcements. If omitted, default(s) must be provided.**

**// Trigger is Required. Hence, it is explicitly shown in the Auto_Update syntax.**

**TRIGGER**

**$[**

***aUserAction* |**

***aSpecifiedTime* |**

***aEvent*** |
***aIfEndif***
**]$...**
**// Zero or more Triggers. If omitted, default(s) must be provided.**
**_end_Rule** **// Marks the end of this rule**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| Listing Status must be set to Expired on or after Expiration Date if Current Listing Status is NOT one of the following: ( Expired, Pending, Sold, Leased). | AUTO_UPDATE<br>LISTINGSTATUS<br>TO "Expired"<br>IF ListingStatus NOTINLIST ("Expired", "Pending", "Sold", "Leased")<br>ENDIF<br>;;<br>TRIGGER<br>IF<br>TODAY is on or after Expiration Date ENDIF<br>;;<br>_end_Rule |
| Listing Status must be set to Pending Over, 4 months past Last Status Change Date, if Current Listing Status is Pending. | AUTO_UPDATE LISTINGSTATUS<br>TO "Pending Over"<br>IF ListingStatus = "Pending"<br>ENDIF<br>;;<br>TRIGGER<br>IF<br>TODAY GE (Last Status Change Date + 4 Months)<br>ENDIF<br>;;<br>_end_Rule |
| Listing Status must be set to Active, 3 days past Last Status Change Date, if Current Listing Status is in (New, Extended, Back on Market, Price Changed, Reactivated). | AUTO_UPDATE LISTINGSTATUS<br>TO "Active"<br>IF ListingStatus INLIST ("New", "Extended", "Back on Market", "Price Changed", "Reactivated")<br>ENDIF<br>;;<br>TRIGGER<br>IF<br>TODAY GE (StatusChangeDate + 3 DAYS)<br>ENDIF<br>;;_end_Rule |

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| A Listing in Incomplete Status must be set to Deleted Status 90 days after last edit date. | AUTO_UPDATE LISTINGSTATUS<br>TO "Deleted"<br>IF ListingStatus = "Incomplete"<br>ENDIF<br>;;<br>TRIGGER<br>IF TODAY GE (Most Recent Listing Modification Date + 90 Days)<br>ENDIF<br>;;<br>_end_Rule |

# 37. APPENDIX – DEFINITIONS OF KEYWORDS USED IN RULE SYNTAX

## Current vs. Historical Values

To differentiate between current and historical values, the following keywords are used where necessary.

| Keyword | Description |
|---|---|
| **CURRENT** ***aFieldName*** | CURRENT ***aFieldName*** is defined as the currently persisted value of ***aFieldName***. |
| **CURRENT** ***aSystemFieldName*** | CURRENT ***aSystemFieldName*** is defined as the System value of ***aSystemFieldName*** at the time the system is queried for that field. |
| **PRIOR** ***aFieldName*** | PRIOR ***aFieldName*** is defined as the historical value of ***aFieldName*** immediately prior to the CURRENT value. |
| **NEW** ***aFieldName*** | NEW ***aFieldName*** is defined as the value of ***aFieldName*** which, if persisted, will replace CURRENT ***aFieldName***.<br>[Think of this as the value the user entered on the screen]. |

## Selections

The following keywords are used to define selections.

| Keyword | Description |
|---|---|
| **INLIST** ( ***aListOfItems***) | Enumerates a comma separated Inclusion list. Examples:<br>• INLIST (CommisssionField1, CommissionField2, CommissionField3)<br>• INLIST ("SFR", "Condo") |

| Keyword | Description |
|---|---|
| **NOTINLIST (*aListOfItems*)** | Enumerates a comma separated Exclusion list. Examples:<br>• NOTINLIST (CommisssionField1, CommissionField2, CommissionField3)<br>• NOTINLIST ("SFR", "Condo") |
| **PASS ( *aCondition* )** | Filters according to specified condition being passed. **Example:**<br>• GT 200 |
| **FAIL** ( ***aCondition*** ) | Filters according to specified condition being failed. |
| **MATCH ( *aPattern*)** | Filters according to a pattern(s) being matched. Used for text values. Well-known patterns include:<br>• URL<br>• Email<br>• U.S. Phone Number (any valid format)<br>• Date Time (any valid format)<br>Additionally, any user defined pattern may be used. |
| **NOTMATCH ( *aPattern* )** | Filters according to a pattern(s) being not matched. Used for text values. Well-known patterns include:<br>• URL<br>• Email<br>• U.S. Phone Number (any valid format)<br>• Date Time (any valid format)<br>Additionally, any user defined pattern may be used. |

## Comparison Operators

| Keyword | Description |
|---|---|
| **EQUAL, or EQ, or =** | Used to compare Values or Dates |
| **NOTEQUAL, or NE, or <>** | Used to compare Values or Dates |
| **LESSTHAN, or LT, or <** | Used to compare Values |
| **LESSTHANOREQUAL, or LTEQ, or <=** | Used to compare Values |
| **GREATERTHAN, or GT, or >** | Used to compare Values |
| **GREATERTHANOREQUAL, or GTEQ, or >=** | Used to compare Values |
| **BEFORE** | Used to compare Dates |
| **ONORBEFORE** | Used to compare Dates |
| **AFTER** | Used to compare Dates |
| **ONORAFTER** | Used to compare Dates |

### Miscellaneous

| Keyword | Description |
|---|---|
| **ALL** | All items in a group or selection |
| **UNCONDITIONAL** | Unconditionally |
| **MAX** | Count of all items in a group or selection |
| **YES, NO** | Self explanatory |
| **REQUIRED, OPTIONAL** | Self explanatory |

## 38. APPENDIX – DEFINITIONS OF VARIABLES USED IN RULE SYNTAX

### Date & Time Offset

| Term | Definition & Syntax |
|---|---|
| ***aDateOffset*** | A specified date offset.<br>***aDateOffset =***<br>**$[ + \| - ]$ *aInteger***<br>**$[DAYS \| WEEKS \| MONTHS \| YEARS]$**<br><br>**Examples:**<br>• + 3 DAYS<br>• - 2 YEARS |
| ***aDateTimeOffset*** | A specified date and time offset<br>***aDateTimeOffset = $[ aDateOffset aTimeOffset ]$***<br><br>**Examples:**<br>• + 2 DAYS - 10 HOURS |
| ***aTimeOffset*** | A specified time offset.<br>***aTimeOffset =***<br>**$[ + \| - ]$ *aInteger***<br>**$[ MILLISECS \|SECONDS \| MINUTES \| HOURS ]$**<br><br>**Examples:**<br>• + 10 MILLISECS<br>• - 2 HOURS |

## Value Offset

| Term | Definition & Syntax |
|---|---|
| ***aValueOffset*** | A specified value offset<br>***aValueOffset*** **=**<br>**$[ + \| - ]$**<br>**$[** ***aValue*** **\|** ***aFieldName*** **\|** ***aComputedValue]$***<br><br>**Examples:**<br>• + 90<br>• - DaysOnMarket<br>• + 5% of ListPrice |

## An Instance of a single-valued Entity

| Term | Definition & Syntax |
|---|---|
| ***aComputedValue*** | A value which is computed using one or more ***aValue*** and ***aFieldName*** and arithmetic or text manipulation operators.<br>**Examples:**<br>• Calculation of DOM, ADOM, CDOM<br>• Area is a concatenation of MLS Map # + MLS Coordinate<br>• Street field is created from a set of concatenated sub-fields (Street Number, Street Name, etc.), possibly with punctuation between, and formatting (e.g. uppercase) applied. |
| ***aDate*** | A date value, expressed in any valid date format e.g., 12/31/2015 or Nov. 3, 1997. |

| | |
|---|---|
| ***aDateFieldName*** | Name of a field which contains a date. During rule execution, this will be translated to the date value of that field.<br>**Examples:**<br>• ListDate<br>• ExpirationDate<br>• ClosingDate<br>• TODAY (a system value for current day's date) |
| ***aFieldName*** | Name of a Data field. During rule execution, this will be translated to the value of that data field, within the execution context. For instance, when validating user input, the value is what is entered on the input screen. In other cases, it will most likely be the same as CURRENT value. |
| ***aInteger*** | An integer numeric value, e.g., 15. |
| ***aMessage*** | A text string, containing a warning or error message. |
| ***aPattern*** | A regular expression, or a defined and/or commonly understood pattern.<br>**Examples:**<br>• URL<br>• U.S. Phone Number 1-nnn-nnn-nnnn<br>• 5 digit zip code xxxxx<br>• A Local APN format XXX-NNNNNNNNNNN |
| ***aSystemFieldName*** | Name of a field whose value is provided by the system. During rule execution, this will be translated to the system value of that field.<br>**Examples:**<br>• Current Date and Time<br>• A sequential number |
| ***aListing*** | Some unique identifier for a Listing, e.g., a Listing ID. |
| ***aValue*** | A data value, matching in type and format to the value expected in that usage context. |

## An Instance of a Group or Collection

| Term | Definition & Syntax |
|---|---|
| ***aFieldGroup*** | A group of fields, specified per the DEFINITION clause in the associated Rule.<br>**Example:**<br>• CommissionFieldGroup is composed of "BuyerAgencyCompensation", "SubAgencyCompensation", and "TransactionBrokerCompensation" |
| ***aListingGroup*** | A group of Listings, specified per the DEFINITION clause in the associated Rule.<br>**Examples:**<br>• All My Listings (i.e., logged-in user's listings)<br>• All listings for an Salesperson, Broker, or Office<br>• All Partial, or Sold, or Expired Listings<br>• All Listings that were created in a date range |

## Miscellaneous

| Term | Definition & Syntax |
|---|---|
| ***aCondition*** | An *expression* or *assertion* that evaluates to True or False<br><br>Where,<br>*Assertion* = A positive statement or declaration;<br>*Expression* = a combination of one or more explicit values, constants, variables, operators, and functions that evaluates to another value |
| ***aComputationInstruction*** | Freeform text. A rule or set of rules which specifies how a value is computed.<br>**Examples:**<br>• Address = Uppercased concatenation of Street, City, State, Zip.<br>• Total Baths is computed as F.H where F = number of full baths and H = number of half baths. |

| Term | Definition & Syntax |
|---|---|
| ***aAutopopSource*** | Specifies the 3rd party data source from which the autopop values are obtained. |

## 39. APPENDIX. WORK IN PROGRESS - RULES NOT YET FINALIZED

The following proposed rules need further analysis before they can be finalized.

1. **COPY_LISTING**
   - **Clone field values and media from existing listing into new listing**
2. **AUTO_UPDATE MEDIA**
   - **For example, delete all non-primary photos one year after listing is off-market**
3. **ALLOW_DUPLICATE_LISTING**
   - **Address duplicate listing cross-listing**
4. **Rule which incorporates RETS Query specifications**

### COPY_LISTING

#### DESCRIPTION

Copy ("clone") field values and media from existing listing into new listing.

#### SYNTAX

```
COPY_LISTING
{ YES | NO }
OWNEDBY { ANY | <ownedBy1>,… } //Who owns source listing
LISTINGSTATUS { ANY | INLIST <aListingStatus1>, … | NOTINLIST <aListingStatus1>, }
//What source Listing Status(es) can be copied from
{
//Between which pair(s) of source/target Listing Classes
FROMLISTINGCLASS
{ ANY | INLIST <aListingClass1>,… | NOTINLIST <aListingClass1>, … }
// Source Listing Class(es)
TOLISTINGCLASS
// Target Listing Class(es)
{ ANY | INLIST <aListingClass1>,… | NOTINLIST <aListingClass1>, … }
//From and To pairings can be repeated
FIELDS //These source fields can be copied to target
{ Two ways to describe which fields to copy
//Method 1:
// Using one of REBR's Selection Syntaxes. For example, INLIST, NOTINLIST, etc.
```

**//Method 2:**
**//Using a reference to a Spreadsheet or Decision Table**
**}**
**} …**

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| A listing can be copied from a source listing to a new listing.<br>The following fields cannot be copied:<br>• PublicRemarks<br>• PrivateRemarks<br>• Directions<br>• ListPrice<br>• ListDate<br>• ExpireDate<br>• CloseDate<br>• Photos<br>• Documents | COPY_LISTING YES<br>FIELDS NOTINLIST ("PublicRemarks", "PrivateRemarks", "Directions", "ListPrice", "ListDate", "ExpireDate", "CloseDate", PHOTOS, DOCUMENTS)<br><br>TBD - REVIEW HOW PHOTOS AND DOCUMENTS ARE REFERENCED IN REBR |
| A listing can be copied from a source listing to a new listing. The source listing must belong to the SALESPERSON, or to the same Broker. Non on-market listings can be copied.<br>• FROM Single Family TO (Single Family, Condo), and<br>• FROM Farm TO (Land, Single Family)<br>The following fields can be copied: | COPY_LISTING<br>YES<br>OWNEDBY (Me, MyBroker)<br>LISTINGSTATUS NOTINLIST OnMarket }<br>FROMLISTINGCLASS INLIST SFR<br>TOLISTINGCLASS INLIST (SFR, Condo, Townhouse)<br>FIELDS<br>FROMLISTINGCLASS INLIST Farm<br>TOLISTINGCLASS INLIST (SFR, Land)<br>FIELDS<br>(See DecisionTable xyz) |

## AUTO_UPDATE MEDIA

How do we want to structure this rule??

| | |
|---|---|
| A Listing must have only the primary photo one year after current off-market status change date. | AUTO_UPDATE MEDIA<br>**INSTRUCTION**<br>(Delete all photos, except Primary Photo)<br>IF (Status is Off Market) ENDIF<br>TRIGGER<br>IF TODAY GE (Off Market Status Change Date + 1 YEAR)<br>ENDIF |

## ALLOW_DUPLICATE_LISTING

### DESCRIPTION

Address duplicate listing and cross-listing.

### SYNTAX

```
ALLOW_DUPLICATE_LISTING
{ YES | NO }
LISTINGCLASS
{ ANY | < INLIST [ <aListingClass1> , <aListingClass2>, … ] }
// Duplicate listings are allowed between the above Listing Classes
//At least two Listing Classes required
IF {
OWNEDBY { ANY | <ownedBy1>,… } //e.g. ListingOwner, Co-Lister
[Other conditions]
}
//Listing owner default if not specified

LISTING_IS_DUP_IF //What criteria are used to detect duplicate listings?
{
 LISTINGCLASS // Listings in which listing classes can be compared?
{ ANY | INLIST [ <aListingClass1> , <aListingClass2>, … ] }
// At least two Listing Classes across which listings are compared
[Other conditions]
 [AND]
MATCHON // What must be same between two listings
            {
            [<aDataField>, … ] // At least one field to match on
            |
            [ <aCondition> ] // Any condition other than a match on fields
            } …
} … // MATCHON can be repeated
```

## EXAMPLES

| RuleSpeak "Structured English" | REBR Notation |
|---|---|
| A listing may be cross-listed by the listing owner only across residential and farm classes. A listing is considered a duplicate if there is a match on either address field or Tax ID. | ALLOW_DUPLICATE_LISTING<br>YES<br>LISTINGCLASS INLIST (RES , FRM)<br><br>LISTING_IS_DUP_IF<br>{ LISTINGCLASS<br>INLIST (RES , FRM, COMMERCIAL, CONDO)<br>MATCHON Street, City, State, Zip<br>MATCHON TaxID<br>} |
| A listing may be cross-listed by the listing owner only across residential and farm classes and across sale and lease transaction types. A listing is considered a duplicate if there is a match on Tax ID with one of three Tax ID fields: TaxID, TaxID2, or TaxID3. | ALLOW_DUPLICATE_LISTING<br>YES<br>LISTINGCLASS INLIST (Res, SFR Townhome)<br>LISTINGTXNTYPE INLIST (Sale, Lease)<br><br>LISTING_IS_DUP_IF<br>{<br>LISTINGCLASS<br>INLIST (Sale or Lease; Res, SFR Townhome)<br>MATCHON (TaxID matches one of TaxID, TaxID2, TaxID3) |

## INCORPORATING QUERY SYNTAX INTO A business RULE

In some *extremely* rare cases, it may be needed to integrate query syntax into a business rule. For example, let's say one wanted to constrain a listing's MLS Area to "MLS Area 12" if the listing is located inside the geographic boundary of a particular polygon. There's no good way to describe that polygon in pure Structured English.

To reference a RETS query in a business rule condition, use similar syntax to how such a search is implemented in the Web API. Use the keyword "PROPERTY SEARCH FILTER" in Structured English to specify a RETS query of a property resource. In REBR, just use the RETS syntax as illustrated below.

**Structured English:**

> Area must be equal to "12" if PROPERTY SEARCH FILTER is equal to geo.intersects(Location,POLYGON((-127.02 45.08,-127.02 45.38,-127.32 45.38,-127.32 45.08,-127.02 45.08)))

**REBR:**

```
CONSTRAIN_VALUE Area
IF Property?$filter=geo.intersects(Location,POLYGON((-127.02 45.08,-127.02 45.38,-127.32 45.38,-127.32
45.08,-127.02 45.08)))
ENDIF
EQUAL "12"
```

## AUTHORS

**The REBR standard was formulated and refined by:**

- Amod Singhal (RPR)
- Matt Cohen (Clareity Consulting)
- Jeremy Crawford (RESO)
- Susie Cass (RPR)
- *With the input of members of the Research & Development Business Rules sub-group*