# Model Expression

A `Model` is a component of Entity Context that defines how records are interpreted, categorized, and mapped to Record Context. It is paired with a `Policy` to define a `Domain`, and so all evaluation is known to be done against a specific data definition.

Before a record is evaluated through policy, its context needs to be established. If the record is already known, then its associated context is read and provided as input to policy evaluation. Otherwise, the Domain's Model is used to resolve important details about the Record. The Model defines a datastore-agnostic way of categorizing fields, and a set of mappings from record-field content to properties of context. The latter provides a stable ontology for policy to reference.

Connecting these domain-independent pieces are groups of "resolvers" that know how to work with domain-specific schema and structure. A specific resolver group is chosen, either by naming it as part of Datastore Export or as input to a decision API call, and the engine uses it to form context before proceeding with policy evaluation. The system uses each resolver in the group to see if a given field is present, resulting in full knowledge about the categories and context properties associated with the Record, and making this knowledge available to users via the Change Graph.

## Model Syntax

Models are defined as JSON-encoded objects. They are provisioned via the entity context APIs, and therefore are versioned over time. Unlike Policy, which is decomposed to make it simple to abstract common logic, Models are self-contained so it's possible to see the complete ontology that they define and ensure there are no inconsistencies or conflicts in the way terms are defined and mapped.

### ModelRoot

The root object of a model has the following structure:

| Field Name | Type | Required | Default |
|---|---|---|---|
| name | string | ✓ | |
| encodingVersion | SemVer | ✓ | |

| Field Name | Type | Required | Default |
|---|---|---|---|
| categories | array of Category | | |
| context | array of ContextEntry | | |
| resolverGroups | ResolverGroups | | |

The value of `name` is used to uniquely identify this `model` in Entity Context. The value of `encodingVersion` must be `1.0.0`.

## Category

| Field Name | Type | Required | Default |
|---|---|---|---|
| name | string | ✓ | |
| id | StableIdentifier | ✓ | |
| fieldSet | array of CategoryField | | |

A Category represents a set of fields that are used together (like "contact data"). The value of `name` is meant to be informational and human-readable, so it does not need to be unique or follow any specific requirements.

## StableIdentifier

A `Stable Identifier` is a `string` that must be unique and must not change once it has been assigned. The set of valid characters are the alpha-numerics and period, dash, and underscore.

## CategoryField

| Field Name | Type | Required | Default |
|---|---|---|---|
| name | string | ✓ | |
| id | StableIdentifier | ✓ | |

A `CategoryField` represents a single field within a `Category`, and follows the same convention for the value of `name`. If a category is named "contact data" then examples of field names might be "email" or "cell phone".

## ContextEntry

| Field Name | Type | Required | Default |
|---|---|---|---|
| name | StableIdentifier | ✓ | |
| type | `property` or `tag` | | `property` |
| options | ContextOptions | | |

A `ContextEntry` defines how a field from a record forms Record Context. If the value of `type` is `property` then both the entry's name and the record field's value are captured in-context. For `tag` entries only the entry's name is included.

## ContextOptions

| Field Name | Type | Required | Default |
|---|---|---|---|
| associative | boolean | | `false` |
| anonymous | boolean | | `false` |
| delegated | boolean | | `false` |
| retention | ISO 8601 Duration | | unbounded |
| score | integer | | `0` |

If the value of `associative` is `true` then this tag or property is applied to any other records that are part of the current session.

If the value of `anonymous` is `true` then the property value will be cryptographically hashed to hide the actual value. If these options are for a `tag` then `anonymous` is ignored.

If the value of `delegated` is `false` then the property can take on multiple values. If the value is `true` then each new value for this property replaces the previous value (as in, the previous owner delegates to the new owner of this property). Properties that are both `delegated` and `associative` have the effect of applying the consistent function to this property for every operation. If these options are for a `tag` then `delegated` is ignored.

The value of `retention` is included in the context graph to document the valid retention period for the associated record. This value can be referenced in policy as a duration using the `urn:tranquildata:attribute:retention` identifier in the `urn:tranquildata:category:record` category. The effect of this value can be referenced in policy as a boolean using the

`urn:tranquildata:attribute:expired` identifier in the `urn:tranquildata:category:record` category.

For each field in a record that matches a `Context Entry` the associated score is added to the total score for that record, and that total score is captured in-context. This value can be referenced in policy as an integer using the `urn:tranquildata:attribute:score` identifier in the `urn:tranquildata:category:record` category. Similarly, as records are accepted into context the active session maintains a cumulative score that can be referenced in policy through the `urn:tranquildata:category:session` category.

## ResolverGroups

| Field Name | Type | Required | Default |
|---|---|---|---|
| defaultGroup | StableIdentifier | ✓ | |
| groups | array of ResolverGroup | ✓ | |

The value of `defaultGroup` must name one of the resolver groups defined in `groups`. When no group is specified for evaluation, this this group is used.

## ResolverGroup

| Field Name | Type | Required | Default |
|---|---|---|---|
| name | StableIdentifier | ✓ | |
| resolvers | array of Resolver | ✓ | |

The value of `name` must be unique amongst all groups in this model.

## Resolver

| Field Name | Type | Required | Default |
|---|---|---|---|
| recordField | RecordFieldName or JoinStatement | ✓ | |
| resourceContext | boolean | | `false` |
| categoryKey | CategoryKey | | |
| contextKey | StableIdentifier | | |

| Field Name | Type | Required | Default |
|---|---|---|---|
| matchRules | map of RecordFieldName to string | | |

If the value of `recordField` matches a field name in the given record, then the field is "resolved" and category/context logic applied using the associated field value.

If the value of `resourceContext` is `true` then the value of a resolved record field identifies the Subject associated with the record. If that Subject is not present in context, or if multiple identifiers are resolved, then the record is considered invalid against the model and will fail evaluation. Otherwise, the associated Subject is included as part of Record Context and is available during policy evaluation for query expressions.

The values of `categoryKey` and `contextKey` are used (respectively) to map this field to a known Category or Context definition. When a record field is successfully resolved, the associated logic is applied. If either value is not present in the model then the model is considered invalid.

The values in `matchRules` provide additional requirments for resolving a record field. If `matchRules` is absent or empty then only `recordField` is used. If `matchRules` has values, however, then the keys are used to match record field names and the values from the map compared to the value of the matched field. At least one entry in `matchRules` must match for the record field to be resolved.

## RecordFieldName

A `RecordFieldName` is a `string` that expresses a relative or full path to a record field in a syntax meaningful to the underlying datastore or document format. For instance, for a record associated with a `Postgres` operation, a value of "id" would name a column in the current table. You could also use "testing.users.id" to specify the column "id" in the table "users" under the schema "testing". Case is ignored when matching each component of a `RecordFieldName`.

## JoinStatement

| Field Name | Type | Required | Default |
|---|---|---|---|
| fieldNames | array of RecordFieldName | ✓ | |
| delimiter | string | | |

Instead of matcing on a specific record field name to resolve the field value, a `JoinStatement` "joins" one or more field values to generate a value from the content of the record. Each of the

values of `fieldNames` is used in-order to match record field names, with the resulting set of field values joined using the given delimeter as a separator. If any one of the values in `fieldNames` cannot be matched then the field is not resolved.

As an example, suppose a record has a field named "type" with value "Patient", and "identifier" with value "37". If a `JoinStatement` is defined with `fieldNames` equal to "[type, identifier]" and `delimeter` set as "/" then the resulting record field value is successfully resolved as "Patient/37". This is often used to identify the Subject associated with a record when the components of the Subject's identifier are scattered across multiple fields.

### CategoryKey

| Field Name | Type | Required | Default |
|---|---|---|---|
| category | StableIdentifier | ✓ | |
| field | StableIdentifier | | |

A `CategoryKey` specifies the category, and optionally the field within that category, that a record field represents.

## Using Categories in Evaluation

Providing a mapping from record field name to category identifier validates record content and ensures that context captures useful details for later audit. It also serves two functional purposes from an evaluation perspective.

As input to policy evaluation, the set of category identifiers associated with any resolved record fields is provided as a bag of strings using the `urn:tranquildata:attribute:categories:requested` in the `urn:tranquildata:category:metadata` category. This supports policies that apply to specific categories of data, and logic that needs to compare the requested set of categories to a set kept elsewhere in context (e.g., as a property of a Subject Relationship or Group Member).

When policy decisions are returned, policy authors may include Obligations with the `urn:tranquildata:obligation` obligation identifier and the `urn:tranquildata:attribute:categories:allowed` attribute identifier. This communicates back to the engine the set of category identifiers that are allowed based on which policies applied to the request. If the engine is running in either `filter` or `redact` modes, then this obligation will be

used to (respectively) drop any records that contain categories outside the allowed set or redact any record fields not associated with the allowed categories.

## Example

This model defines a few contact fields, context needed to handle EU or Trial data, and one mapping from a specific DBMS structure to these general terms.

```
{
    "name": "urn:tranquildata:model:example",
    "encodingVersion": "1.0.0",

    "categories": [
        {
            "name": "contact",
            "id": "category.contact",
            "fieldSet": [
                {
                    "name": "email",
                    "id": "category.contact.email"
                },
                {
                    "name": "cell",
                    "id": "category.contact.cell"
                }
            ]
        }
    ],

    "context": [
      {
        "name": "EU-data",
        "type": "property",
        "options": {
            "associative": true,
            "score": 1
        }
      },
      {
        "name": "from-trial",
        "type": "tag",
        "options": {
            "retention": "P30D"
        }
      }
    ],

    "resolverGroups": {
      "defaultGroup": "accounts-db",
      "groups": [
```

```
        {
          "name": "accounts-db",
          "resolvers": [
            {
                "recordField": "users.login",
                "categoryKey": {
                    "category": "category.contact",
                    "field": "category.contact.email"
                }
            },
            {
                "recordField": "users.id",
                "resourceContext": true
            },
            {
                "recordField": "users.trial",
                "matchRules": {
                    "users.trial": "true"
                },
                "contextKey": "from-trial"
            },
            {
                "recordField": "contact.value",
                "matchRules": {
                    "contact.type": "cell"
                },
                "categoryKey": {
                    "category": "category.contact",
                    "field": "category.contact.phone"
                }
            },
            {
                "recordField": "users.country",
                "matchRules": {
                    "users.region": "EU",
                },
                "contextKey": "EU-data"
            }
          ]
        }
      ]
    }

}
```