Subject Expression

As discussed in the previous section, Subject Context captures subjects (like a person or company), their relationships, associated groups (like an organization or team), and their members. The types of properties, relationships, and members are purposefully open-ended to support structure that maps to any given deployment requirements.

Structure

Each subject and group has a unique name that must be a valid URI, and an arbitrary number of key-value properties. The value of each property is one or more string.

A subject may have zero or more relationships, and a group zero or more members. A relationship has a `target` (who the relationship is with) and a `type` (what type of relationship is being described). This subject-type-target triple is unique, meaning that a given subject may only have one relationship of a given type with a given target, but may have many relationships of a given type or with a given target. A member must be uniquely named within a group.

The value of relationship and member properties is one or more string. The target of a relationship, and the member of a group, does not itself need to be a subject or a group, although that is typical for running mapped queries (below). See the Subject Context API definition for details about syntax and populating context.

Relationship Direction

Relationships are considered to be "one-way." That is, just because Alice has a relationship with Bob it does not mean that Bob has the same relationship with Alice. This is by design to give each subject ownership over their relationships. Policy authors are encouraged to consider this subject-centric approach in how they model rules, for instance, always verifying a relationship from the perspective of the owner of data instead of assuming that a relationship is mutual.

Query Definition

Each query is a single string that is used in the AttributeId field of an Attribute Designator to resolve values during policy evaluation. Because XACML only allows this string to define attribute

resolution, and because the identifier string must be a valid URI, the syntax of all queries follows a similar pattern.

Queries about the resource or query subject should be issued (respectively) in the category `urn:tranquildata:category:subject:resource` or `urn:tranquildata:category:subject:query`. Queries for a group should be issued in the category `urn:tranquildata:category:group`. If the syntax is not valid for the given category then an error is returned. If any component of a query cannot be resolved in-context then an empty bag is returned. Within a given session, the same query will *always* return the same value (or error).

All components provide the following properties:

Property	Description
`name`	the name of the component as it appears in-context
`timestamp`	the last time the component was updated

Subject Query

A Subject Query starts with the prefix `property` or `relationship`, followed by `:` (a colon). If the prefix is `property` then what follows is a string naming a property to retrieve. For instance, this query returns the name of a subject:

```
property:name
```

RelationshipQuery

If the query prefix is `relationship` then what follows is one or more key-value pairs, with each key and value separated by a colon. The key identifies the type of query criteria and the value is a string that defines a constraint on the query. Each of the keys may appear at most once in a relationship query. The supported keys are `type`, `target`, and `property`. The key `property` is required.

If `type` or `target` is absent, or if the value for either key is `*`, then any relationship type or target (respecitvely) is matched. For instance, this query returns the name of all family members for a given subject:

```
relationship:type:family-member:property:name
```

In addition to `name` and `timestamp` the following properties are available for relationships:

Property	Description
`target`	the target of the relationship (same as `name`)
`type`	the type of the relationship
`policy`	the policy (if specified) that governs this relationship

Group Query

A Group Query starts with the prefix `group:[NAME]` followed by `:` (a colon). This is followed by either `property` or `member`. If the prefix is followed by `property` then what follows is a string naming a property to retrieve. For instance, this query returns the name of a subject

```
group:team-one:property:name
```

In addition to `name` and `timestamp` the following properties are available for groups:

Property	Description
`members`	the names of all members of the group

MemberQuery

If the query prefix is followed by `member` then what follows is the name of the member, and then `property:[NAME]`. If the name of the member is `*` then the property value is retrieved for all members. For instance, this query is another way to return the name of all members of the group:

```
group:team-one:member:*:property:name
```

Query Mapping

The values returned from either a Subject or Group query may be *mapped* to another Subject query, where each result value is treated as the name of a Subject. For each name, if the associated Subject is known then the mapped query is run, and the values returned from all queries are returned as a single bag. If any given Subject is unknown then it is skipped.

The syntax for this query is a full Subject or Group query, followed by `:as-context-elements:subjects:`, followed by another Subject query. For instance, if all members of a group

are themselves expected to be Subjects, then here is yet another way to return the name of all members:

```
group:team-one:property:members:as-context-elements:subjects:property:name
```

You could now test the resulting bag versus the actual membership to see which members are not Subjects.

Best Practices

The role of Subject Context is not to replace a CRM or similar service that tracks wide knowledge about people and organizations. It is not designed or optimized for that breadth of data nor the rich kinds of queries and interoperability that is required. Instead, think of Subject Context as a place to track only the knowledge that is critical in determining which rules apply to a given action, or how to evaluate those rules. For instance, knowing where someone lives (city, state, country, etc.) may cause different policies to be chosen, or result in different decisions about data usage. Similarly, details like birth dates, family or employer relationships, or subscription tiers are likely to have similar effects.

As a rule of thumb, it's recommended to keep the amount of context about a subject minimal, as it will help with the design goal of creating clarity to anyone outside the system about what data is held and how that data is used to make decisions.