Congress API Design

http://goo.gl/1E5MeY

- Background
- Data Model Overview
  - Policies
  - Data Sources
- REST Method Overview
- Detailed Type Definitions

Background

This document describes a basic API for interacting with the Congress policy-based management framework. The underlying implementation design can be found at http://goo.gl/YFd2Fr.

The API model is subject to change. A stable v1 proposal is expected after the alpha release.

Data Model Overview

Policies

Policies express the logic the user wishes to enforce on the cloud. Each 'policy' includes properties of the policy (e.g. 'owner'), and a collection of rules declared using the policy language grammar.

Initially, the system will support one policy with an ID of 'classification'. We expect to support multiple policies (multi-tenancy use cases) and associated RBAC in a post-alpha release.

Data Sources

Data sources represent an entity that provides a view of policy-significant data. Each data source instance corresponds to a configured data source plugin, or to a policy (which expresses intermediate and final results as data tables.)

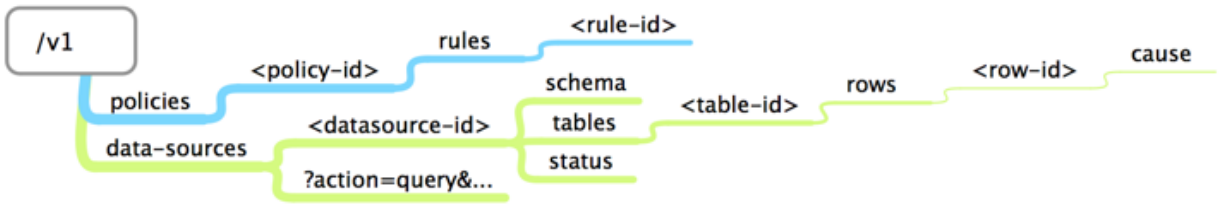
Each data source exposes its status (e.g. db connectivity), data schema, and zero to many 'tables'. Tables contains zero or more 'rows' (tuples) which are evaluated by the policy engine.

Built-in Data Sources:
For the initial release, we are planning to provide the following data source plugins in Congress:

- Neutron
- Nova

- “Internal” (better name pending) - Allows users to create/manage static data (stored in internal DB)

REST Method Overview



Policy (/v1/…)

GET	.../policies	List policies
POST	.../policies	Create policy (post alpha release)
GET	.../policies/<policy-id>	Read policy properties
PUT	.../policies/<policy-id>	Update policy properties (post alpha release)
DELETE	.../policies/<policy-id>	Delete policy (post alpha release)
POST	.../policies/<policy-id>?action=simulate &query=<query> &sequence=<sequence> &action_policy=<action_policy>	Simulate sequence of updates <sequence> and return results of <query> in resulting state. <query> is a string representing a rule. <sequence> is a string representing a sequence of state/rule updates or actions. <action_policy> is the name of a policy detailing the effects of actions.
POST	.../policies/<policy-id>?action=simulate &query=<query> &sequence=<sequence> &action_policy=<action_policy> &delta=true	Same as ‘simulate’ action except returns the change in query results caused by applying the sequence of updates.

Policy Rules (/v1/policies/<policy-id>/…)

GET	.../rules	List policy rules
POST	.../rules	Create policy rule
GET	.../rules/<rule-id>	Read policy rule
PUT	.../rules/<rule-id>	Update policy rule (post alpha release)
DELETE	.../rules/<rule-id>	Delete policy rule

Policy Tables (/v1/policies/<policy-id>/…)

GET	.../tables	List tables

GET	.../tables/<table-id>	Read table properties

Policy Table Rows (/v1/policies/<policy-id>/tables/<table-id>/…)

GET	.../rows	List rows
GET	.../rows?trace=true	List rows with explanation (use ‘printf’ to display)
GET	.../rows/<row-id>	Read row data

Data Source (/v1/…)

GET	.../data-sources	List data sources
POST	.../data-sources	Create (register) data source (post alpha release)
GET	.../data-sources/<ds-id>	Read data source properties
PUT	.../data-sources/<ds-id>	Update data source properties (post alpha release)
DELETE	.../data-sources/<ds-id>	Delete (unregister) data source (post alpha release)
POST	.../data-sources/<ds-id>?action=query	Execute a query (post alpha release)
GET	.../data-sources/<ds-id>/schema	Read data source schema (post alpha release)
GET	.../data-sources/<ds-id>/status	Read data source status (post alpha release)

Data Source Tables (/v1/data-sources/<ds-id>/…)

GET	.../tables	List tables
POST	.../tables	Create table (if supported by data source) (post alpha release)
GET	.../tables/<table-id>	Read table properties
PUT	.../tables/<table-id>	Update table properties (if supported by data source) (post alpha release)
DELETE	.../tables/<table-id>	Delete table (if supported by data source) (post alpha release)

Data Source Table Rows (/v1/data-sources/<ds-id>/tables/<table-id>/…)

GET	.../rows	List rows
POST	.../rows	Create (insert) row (if supported by data source) (post alpha release)
GET	.../rows/<row-id>	Read row data (post alpha release)
PUT	.../rows/<row-id>	Update row (if supported by data source) (post alpha release)

DELETE	.../rows/<row-id>	Delete row (if supported by data source) (post alpha release)
GET	.../rows/<row-id>/cause	Determine cause for row (data source specific: for policies, this will list associated rules for the output) TODO(pjb): may remodel as action on policy-engine

Detailed Type Definitions

Each REST method operates on a resource type that is described using JSON Schema semantics. The type definitions include necessary data descriptions for API documentation, and can be leveraged by the API framework (using a JSON Schema validation library) to validate API calls.

```
- !Type:
    id: PolicyProperties
    title: Policy Properties
    type: object
    properties:
        id:
            title: Policy ID
            type: string
            required: true
        owner_id:
            title: Policy owner
            type: string
            required: true
        description:
            title: Policy description
            type: string
            required: true
        type:
            title: Policy type
            type: string
            required: true
        abbreviation:
            title: Policy name abbreviation
            type: string
            required: false


- !Type:
    id: PolicyRule
    title: Policy rule
    type: object
    properties:
        id:
```

```
            title: Rule ID
            type: string
            required: true
        rule:
            title: Rule definition following policy grammar
            type: string
            required: true
        comment:
            title: User-friendly comment
            type: string
            required: false


- !Type:
    id: DataSourceProperties
    title: Data source properties
    type: object
    properties:
        id:
            title: Data source ID
            type: string
            required: true
        owner_id:
            title: Data source owner
            type: string
            required: true
        enabled:
            title: Enabled status of data source
            type: string
            required: false
            default: true
        type:
            title: Type of data source
            type: string
            enum: [‘plugin’, ‘policy’]
            required: true
        config:
            title: Data source driver specific configuration
            type: object
            required: false


- !Type:
    id: DataSourceSchema
    title: Data source schema
    type: object
    properties:
        tables:
            title: Tables exposed by data source
            type: array
```

```
            items: {$ref: TableSchema}
            required: true


- !Type:
    id: TableSchema
    title: Table schema
    type: object
    properties:
        table_id:
            title: Table ID
            type: string
            required: true

        columns:
            title: Data source table column spec
            type: array
            items: {
                type: object
                properties:
                    name:
                        type: string
                        required: true
                    description:
                        type: string
                        required: false
            }
            required: true


- !Type:
    id: DataSourceStatus
    title: Data source status
    type: object
    properties: {}  # To be defined


- !Type:
    id: DataSourceTableProperties
    title: Data source table properties
    type: object
    properties:
        id:
            title: Data source table ID
            type: string
            required: true


- !Type:
    id: DataSourceTableRow
```

```
    title: Data source properties
    type: object
    properties:
        id:
            title: Row ID
            type: string
            required: true
        data:
            title: Row data
            type: array
            items: {type: string}
            required: true



- !Type:
    id: ListResult
    title: Data source properties
    type: object
    properties:
        results:
            title: Result items
            type: array
            items: {}
            required: true
        # Future pagination-related properties to be added...
        #result_count:
        #    title: Count of results across all pages
        #    type: integer
        #    required: true

- !Type:
    id: DataSourceTableRowListResult
    title: List of DataSourceTableRows
    extends: {$ref: ListResult}
    properties:
        results:
          items: {$ref: DataSourceTableRow}
    additionalProperties: false
```