

公共交通情報流通連携基盤システム

実装仕様書

2013年3月29日

株式会社横須賀テレコムリサーチパーク

1. システム構成

公共交通情報流通連携基盤システム

本システムは、Linux OS (Kernel 2.6.26 以上) を搭載したクラウドサーバ上で動作する。具体的なシステム構成は以下の通りである。

◆基盤FrontEndサーバー

基盤フロントエンドサーバーは、外部に情報配信を行うデータを扱うサーバーである。Web Application FrameworkとしてRailsを使用し、REST APIなどをこのサーバーに実装する。この環境は、AWS環境にてAutoScalingすることを前提として構築を行う。したがって、実際のDB本体はこの中では稼働せず、全て後述するDBサーバーを参照するように実装される。

本サーバーのソフトウェアの構成を以下に示す。

- ・ Ubuntu Server 12.04 LTS 64bit
 - DHCPにてIP取得
 - ファイヤーウォール設定無し
- ・ nginx PPA
- ・ rbenv
 - JRuby > v1.7.2
 - Ruby on Rails > v3.2.2

◆基盤MongoDBサーバー

基盤MongoDBサーバーは、情報流通連携基盤にて扱うデータをストアするためのDBサーバーである。MongoDBが稼働しており、情報流通連携基盤で扱うデータ形式でデータを保持する。

本サーバーのソフトウェアの構成を以下に示す。

- ・ Ubuntu Server 12.04 LTS 64bit
 - DHCPにてIP取得
 - ファイヤーウォール設定無し
- ・ MongoDB PPA

◆基盤PostGISサーバー

基盤PostGISサーバー(バックエンドサーバー)は、内部的に用いるデータをストアするために用いる。PostgreSQLとPostGISが稼働しており、下記に示す国土地理院発行の国土数値情報をストアする。

本サーバーのソフトウェアの構成を以下に示す。

- ・ Ubuntu Server 12.04 LTS 64bit
 - DHCPにてIP取得
 - ファイヤーウォール設定無し
- ・ APT
 - PostgreSQL
 - PostGIS
 - SSH Server

2. コンフィグレーション情報

公共交通情報流通連携基盤システム

本システムのコンフィグレーション情報は以下の通りである。

2-1. application

```
require File.expand_path('../boot', __FILE__)

# Pick the frameworks you want:
# require "active_record/railtie"
require "action_controller/railtie"
require "action_mailer/railtie"
require "active_resource/railtie"
require "sprockets/railtie"
# require "rails/test_unit/railtie"

if defined?(Bundler)
  # If you precompile assets before deploying to production, use this line
  Bundler.require(*Rails.groups(:assets => %w(development test)))
  # If you want your assets lazily compiled in production, use this line
  # Bundler.require(:default, :assets, Rails.env)
end

module Opendata
  class Application < Rails::Application
    # Settings in config/environments/* take precedence over those specified here.
    # Application configuration should go into files in config/initializers
    # -- all .rb files in that directory are automatically loaded.

    # Custom directories with classes and modules you want to be autoloadable.
    # config.autoload_paths += %W(#{config.root}/extras)

    # Only load the plugins named here, in the order given (default is alphabetical).
    # :all can be used as a placeholder for all plugins not explicitly named.
    # config.plugins = [ :exception_notification, :ssl_requirement, :all ]
```

```

# Activate observers that should always be running.
# config.active_record.observers = :cacher, :garbage_collector, :forum_observer

# Set Time.zone default to the specified zone and make Active Record auto-convert
to this zone.
# Run "rake -D time" for a list of tasks for finding time zone names. Default is
UTC.
# config.time_zone = 'Central Time (US & Canada)'

# The default locale is :en and all translations from config/locales/*.rb, yml are
auto loaded.
# config.i18n.load_path += Dir[Rails.root.join('my', 'locales', '*.{rb, yml}').to_s]
# config.i18n.default_locale = :de

# Configure the default encoding used in templates for Ruby 1.9.
config.encoding = "utf-8"

# Configure sensitive parameters which will be filtered from the log file.
config.filter_parameters += [:password]

# Enable escaping HTML in JSON.
config.active_support.escape_html_entities_in_json = true

# Use SQL instead of Active Record's schema dumper when creating the database.
# This is necessary if your schema can't be completely dumped by the schema dumper,
# like if you have constraints or database-specific column types
# config.active_record.schema_format = :sql

# Enforce whitelist mode for mass assignment.
# This will create an empty whitelist of attributes available for mass-assignment
for all models
# in your app. As such, your models will need to explicitly whitelist or
blacklist accessible
# parameters by using an attr_accessible or attr_protected declaration.
# config.active_record.whitelist_attributes = true

# Enable the asset pipeline
config.assets.enabled = true

# Version of your assets, change this if you want to expire all your assets
config.assets.version = '1.0'
end
end

```

2-2. boot

```
require 'rubygems'

# Set up gems listed in the Gemfile.
ENV['BUNDLE_GEMFILE'] ||= File.expand_path('../../Gemfile', __FILE__)

require 'bundler/setup' if File.exists?(ENV['BUNDLE_GEMFILE'])
```

2-3. environment

```
# Load the rails application
require File.expand_path('../application', __FILE__)

# Initialize the rails application
Opendata::Application.initialize!
```

2-4. routes

```
Opendata::Application.routes.draw do
  # The priority is based upon order of creation:
  # first created -> highest priority.

  # Sample of regular route:
  # match 'products/:id' => 'catalog#view'
  # Keep in mind you can assign values other than :controller and :action

  # Sample of named route:
  # match 'products/:id/purchase' => 'catalog#purchase', :as => :purchase
  # This route can be invoked with purchase_url(:id => product.id)

  # Sample resource route (maps HTTP verbs to controller actions automatically):

  resources :datapoints, :places, :diagram

  resource :sparql, :rawdata
  resource "rdf-graph-store", :controller => 'graphstores', :as => 'graphstores'

  # Sample resource route with options:
  # resources :products do
  #   member do
  #     get 'short'
  #     post 'toggle'
  #   end
```

```

#
#   collection do
#     get 'sold'
#   end
# end

# Sample resource route with sub-resources:
#   resources :products do
#     resources :comments, :sales
#     resource :seller
#   end

# Sample resource route with more complex sub-resources
#   resources :products do
#     resources :comments
#     resources :sales do
#       get 'recent', :on => :collection
#     end
#   end

# Sample resource route within a namespace:
#   namespace :admin do
#     # Directs /admin/products/* to Admin::ProductsController
#     # (app/controllers/admin/products_controller.rb)
#     resources :products
#   end

# You can have the root of your site routed with "root"
# just remember to delete public/index.html.
# root :to => 'welcome#index'

# See how all your routes lay out with "rake routes"

# This is a legacy wild controller route that's not recommended for RESTful
applications.
# Note: This route will make all actions in every controller accessible via GET
requests.
# match ':controller(/:action(/:id))(.:format)'
end

```

3. 環境情報

公共交通情報流通連携基盤システム

本システムの環境情報は以下の通りである。

3-1. development

```
Opendata::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb

  # In the development environment your application's code is reloaded on
  # every request. This slows down response time but is perfect for development
  # since you don't have to restart the web server when you make code changes.
  config.cache_classes = false

  # Log error messages when you accidentally call methods on nil.
  config.whiny_nils = true

  # Show full error reports and disable caching
  config.consider_all_requests_local = true
  config.action_controller.perform_caching = false

  # Don't care if the mailer can't send
  config.action_mailer.raise_delivery_errors = false

  # Print deprecation notices to the Rails logger
  config.active_support.deprecation = :log

  # Only use best-standards-support built into browsers
  config.action_dispatch.best_standards_support = :builtin

  # Do not compress assets
  config.assets.compress = false

  # Expands the lines which load the assets
```



```
    config.assets.debug = true
end
```

3-2. production

```
Opendata::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb

  # Code is not reloaded between requests
  config.cache_classes = true

  # Full error reports are disabled and caching is turned on
  config.consider_all_requests_local = false
  config.action_controller.perform_caching = true

  # Disable Rails's static asset server (Apache or nginx will already do this)
  config.serve_static_assets = false

  # Compress JavaScripts and CSS
  config.assets.compress = true

  # Don't fallback to assets pipeline if a precompiled asset is missed
  config.assets.compile = false

  # Generate digests for assets URLs
  config.assets.digest = true

  # Defaults to nil and saved in location specified by config.assets.prefix
  # config.assets.manifest = YOUR_PATH

  # Specifies the header that your server uses for sending files
  # config.action_dispatch.x_sendfile_header = "X-Sendfile" # for apache
  # config.action_dispatch.x_sendfile_header = 'X-Accel-Redirect' # for nginx

  # Force all access to the app over SSL, use Strict-Transport-Security, and use
  # secure cookies.
  # config.force_ssl = true

  # See everything in the log (default is :info)
  # config.log_level = :debug

  # Prepend all log lines with the following tags
  # config.log_tags = [ :subdomain, :uuid ]

  # Use a different logger for distributed setups
  # config.logger = ActiveSupport::TaggedLogging.new(SyslogLogger.new)
```

```

# Use a different cache store in production
# config.cache_store = :mem_cache_store

# Enable serving of images, stylesheets, and JavaScripts from an asset server
# config.action_controller.asset_host = "http://assets.example.com"

# Precompile additional assets (application.js, application.css, and all non-JS/CSS
are already added)
# config.assets.precompile += %w( search.js )

# Disable delivery errors, bad email addresses will be ignored
# config.action_mailer.raise_delivery_errors = false

# Enable threaded mode
# config.threadsafe!

# Enable locale fallbacks for I18n (makes lookups for any locale fall back to
# the I18n.default_locale when a translation can not be found)
config.i18n.fallbacks = true

# Send deprecation notices to registered listeners
config.active_support.deprecation = :notify

end

```

3-3. test

```

Opendata::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb

  # The test environment is used exclusively to run your application's
  # test suite. You never need to work with it otherwise. Remember that
  # your test database is "scratch space" for the test suite and is wiped
  # and recreated between test runs. Don't rely on the data there!
  config.cache_classes = true

  # Configure static asset server for tests with Cache-Control for performance
  config.serve_static_assets = true
  config.static_cache_control = "public, max-age=3600"

  # Log error messages when you accidentally call methods on nil
  config.whiny_nils = true

  # Show full error reports and disable caching
  config.consider_all_requests_local = true
  config.action_controller.perform_caching = false

```

```
# Raise exceptions instead of rendering exception templates
config.action_dispatch.show_exceptions = false

# Disable request forgery protection in test environment
config.action_controller.allow_forgery_protection = false

# Tell Action Mailer not to deliver emails to the real world.
# The :test delivery method accumulates sent emails in the
# ActionMailer::Base.deliveries array.
config.action_mailer.delivery_method = :test

# Print deprecation notices to the stderr
config.active_support.deprecation = :stderr
end
```

4. 初期化情報

公共交通情報流通連携基盤システム

本システムの初期化情報は以下の通りである。

4-1. backtrace_silencers

```
# Be sure to restart your server when you modify this file.

# You can add backtrace silencers for libraries that you're using but don't wish to
# see in your backtraces.
# Rails.backtrace_cleaner.add_silencer { |line| line =~ /my_noisy_library/ }

# You can also remove all the silencers if you're trying to debug a problem that
# might stem from framework code.
# Rails.backtrace_cleaner.remove_silencers!
```

4-2. inflections

```
# Be sure to restart your server when you modify this file.

# Add new inflection rules using the following format
# (all these examples are active by default):
# ActiveSupport::Inflector.inflections do |inflect|
#   inflect.plural /^(ox)$/i, '¥1en'
#   inflect.singular /^(ox)en/i, '¥1'
#   inflect.irregular 'person', 'people'
#   inflect.uncountable %w( fish sheep )
# end
#
# These inflection rules are supported but not enabled by default:
# ActiveSupport::Inflector.inflections do |inflect|
#   inflect.acronym 'RESTful'
# end
```

4-3. mime_types

Be sure to restart your server when you modify this file.

```
# Add new mime types for use in respond_to blocks:
# Mime::Type.register "text/richtext", :rtf
# Mime::Type.register_alias "text/html", :iphone
Mime::Type.register_alias "application/json", :geojson
```

4-4. secret_token

Be sure to restart your server when you modify this file.

```
# Your secret key for verifying the integrity of signed cookies.
# If you change this key, all old signed cookies will become invalid!
# Make sure the secret is at least 30 characters and all random,
# no regular words or you'll be exposed to dictionary attacks.
Opendata::Application.config.secret_token = '
610ab6a762a9b81a90c1916b1fd34a87a456018c219b98797fd173b94c7539d063cbd3db0c23966361d9-
9bfb36bffa8d501ea73820b7ab4a28c0415e0a668f9d'
```

4-5. session_store

Be sure to restart your server when you modify this file.

```
Opendata::Application.config.session_store :cookie_store, key: '_opendata_session'
```

```
# Use the database for sessions instead of the cookie-based default,
# which shouldn't be used to store highly confidential information
# (create the session table with "rails generate session_migration")
# Opendata::Application.config.session_store :active_record_store
```

4-6. wrap_parameters

Be sure to restart your server when you modify this file.

#

```
# This file contains settings for ActionController::ParamsWrapper which
# is enabled by default.
```

```
# Enable parameter wrapping for JSON. You can disable this by setting :format to an
```

```
empty array.  
ActiveSupport.on_load(:action_controller) do  
  wrap_parameters format: [:json]  
end
```

以下の各章では、本システムで提供する標準APIについて説明する。

5. SPARQL-based Command

公共交通情報流通連携基盤システム

SPARQL-based Commandは、SPARQL 1.1プロトコルに準拠した、オープンデータの登録・更新・削除・閲覧・検索機能を提供する。

――[5-1: SPARQL1.1準拠のクエリ発行(GET メソッド)]――

<機能概要>

HTTP GET メソッドを利用して、SPARQL1.1準拠のクエリを発行する。

<メソッド>

GET

<URLパス>

/api/v1/sparql?query=<query>

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する(以下の表)。

[表5-1] SPARQL1.1準拠のクエリ発行(GET メソッド)のリクエストパラメータ

パラメータ名	デフォルト値	説明
query	xsd:string	検索対象パラメータ名

<リクエストヘッダ>

レスポンスの形式を、リクエストヘッダのAcceptで設定する。SPARQL 1.1のSELECTオペレーションを発行する場合に指定できるパラメータ、CONSTRUCT またはDESCRIBEオペレーションを発行する場合に指定できるパラメータ、ASK オペレーションを発行する場合に指定できるパラメータは以下の表の通りである。

[表5-2] SELECTオペレーションのレスポンス形式を指定するAcceptヘッダ値

パラメータ値	説明
application/sparql-results+xml	SPARQL Query Results XML Formatに基づくレスポンス
application/sparql-results+json	SPARQL Query Results JSON Formatに基づくレスポンス

[表5-3] レスポンスのRDFグラフ表現形式を指定するAcceptヘッダ値

パラメータ値	説明
application/rdf+xml	RDF/XML
text/plain	N-Triples
text/rdf+n3	Notation3

[表5-4] レスポンスのバイナリ値形式を指定するAcceptヘッダ値

パラメータ値	説明
application/sparql-results+xml	SPARQL Query Results XML Formatに基づくレスポンス
text/boolean	テキスト表現(true/false)

<レスポンス>

レスポンスは以下の通りである。

- ・ SELECTオペレーションのレスポンスは、acceptヘッダ値に基づき、以下のいずれかである。
 - SPARQL Query Results JSON Format に基づくレスポンス
 - SPARQL Query Results XML Format に基づくレスポンス
- ・ CONSTRUCT、DESCRIBE オペレーションのレスポンスは、RDFグラフデータである。このフォーマットは、acceptヘッダ値で指定した通りである。
- ・ ASK オペレーションのレスポンスは、acceptヘッダ値に基づき、以下のいずれかである。
 - SPARQL Query Results XML Format に基づくレスポンス
 - true またはfalse の文字列

——[5-2: SPARQL1.1準拠のクエリ発行(POSTメソッド)]——

<機能概要>

HTTP POSTメソッドを利用して、SPARQL1.1準拠のクエリを発行する。

<メソッド>

POST

<URLパス>

/api/v1/sparql/

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、メッセージボディ部に格納する。

[表5-5] SPARQL1.1準拠のクエリ発行(POSTメソッド)のリクエストパラメータ

パラメータ名	デフォルト値	説明
query	xsd:string	SPARQLクエリ文をURLエンコーディングしたもの

<リクエストヘッダ>

レスポンスの形式を、リクエストヘッダのAcceptで設定する。その方法は、「3.1.1 SPARQL1.1準拠のクエリ発行(GETメソッド)」と同じである。(以下の表を参照のこと)

<レスポンス>

レスポンスは以下の通りである。

- ・ SELECTオペレーションのレスポンスは、acceptヘッダ値に基づき、以下のいずれかである。
 - SPARQL Query Results JSON Format に基づくレスポンス
 - SPARQL Query Results XML Format に基づくレスポンス
- ・ CONSTRUCT、DESCRIBE オペレーションのレスポンスは、RDFグラフデータである。このフォーマットは、acceptヘッダ値で指定した通りである。
- ・ ASK オペレーションのレスポンスは、acceptヘッダ値に基づき、以下のいずれかである。
 - SPARQL Query Results XML Format に基づくレスポンス
 - true またはfalse の文字列

[5-3: RDFグラフの閲覧]

<機能概要>

RDFグラフを閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/rdf-graph-store?graph=<graph>
/api/v1/rdf-graph-store?default

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する

[表5-6] RDFグラフの閲覧のリクエストパラメータ

パラメータ名	デフォルト値	説明
graph	(指定なし)	閲覧するRDFグラフの識別URI

<リクエストヘッダ>

レスポンスの形式を、リクエストヘッダのAcceptで設定する。その方法は、「3.1.1 SPARQL1.1準拠のクエリ発行(GETメソッド)」のCONSTRUCTオペレーションと同じである。(別表を参照のこと)

<レスポンス>

Acceptヘッダで指定した形式でエンコードされた、RDFグラフ表現。

――[5-4: RDFグラフの追加]――

<機能概要>

RDFグラフを追加する。

<メソッド>

POST

<URLパス>

/api/v1/rdf-graph-store?graph=<graph>

/api/v1/rdf-graph-store?default

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのURL部に格納する。追加するRDFグラフは、メッセージボディ部に格納する。

[表5-7] RDFグラフの閲覧のリクエストパラメータ

パラメータ名	デフォルト値	説明
graph	(指定なし)	追加対象のRDFグラフを識別するURI

<リクエストヘッダ>

追加するRDFグラフの形式を、リクエストヘッダのContent-type で設定する。指定できるパラメータ値とその意味は、「3.1.1 SPARQL1.1準拠のクエリ発行(GETメソッド)」内の別表を参照のこと。

<レスポンス>

成功時、レスポンスボディは空である。

[5-5: RDFグラフの更新]

<機能概要>

RDFグラフを更新する。本クエリの操作完了後に登録されているRDFグラフは、本クエリで指定したものである。本クエリに含まれないRDFグラフは削除される。

<メソッド>

PUT

<URLパス>

/api/v1/rdf-graph-store?graph=<graph>
/api/v1/rdf-graph-store?default

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、PUTメソッドのURL部に格納する。更新するRDFグラフは、メッセージボディ部に格納する。

[表5-8] RDFグラフの更新のリクエストパラメータ

パラメータ名	デフォルト値	説明
graph	(指定なし)	更新対象のRDFグラフを識別するURI

<リクエストヘッダ>

更新するRDFグラフの形式を、リクエストヘッダのContent-type で設定する。指定できるパラメータ値とその意味は、「3.1.1 SPARQL1.1準拠のクエリ発行(GET メソッド)」内の別表を参照のこと。

<レスポンス>

成功時、レスポンスボディは空である。

[5-6: RDFグラフの削除]

<機能概要>

RDFグラフを削除する。本クエリの実施後、RDFグラフは空になる。

<メソッド>

DELETE

<URLパス>

/api/v1/rdf-graph-store?graph=<graph>
/api/v1/rdf-graph-store?default

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、メッセージボディ部に格納する。

[表5-9] RDFグラフの削除のリクエストパラメータ

パラメータ名	デフォルト値	説明
graph	(指定なし)	削除対象のRDFグラフを識別するURI

<レスポンス>

成功時、レスポンスボディは空である。

——[5-7: Triple の閲覧]——

<機能概要>

指定された主語・述語・目的語をもつTriple を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/rawdata?subject=<subject>&predicate=<predicate>&object=<object>&graph=<graph>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表5-10] Triple の閲覧のリクエストパラメータ

パラメータ名	デフォルト値	説明
subject	(指定なし)	出力対象のTriple の主語
predicate	(指定なし)	出力対象のTriple の述語
object	(指定なし)	出力対象のTriple の目的語
graph	default	閲覧するRDFグラフを識別するURI

<リクエストヘッダ>

レスポンスの形式を、リクエストヘッダのAcceptで設定する。その方法は、「3.1.1 SPARQL1.1準拠のクエリ発行(GETメソッド)」のCONSTRUCTオペレーションと同じである。(別表を参照のこと)

<レスポンス>

subject、predicate、object で指定されたリソースをもつTriple に対するRDFグラフ表現。出力されるグラフの形式は、Acceptヘッダで指定されたものである。

[5-8: Triple の追加]

<機能概要>

Triple を追加する。情報流通連携基盤システムに既に登録されているTriple と主語、述語、目的語のすべてが一致するTriple を、追加登録することはできない。

<メソッド>

POST

<URLパス>

/api/v1/rawdata?graph=<graph>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。追加するTripleは、リクエストボディに記述する。

[表5-11] Triple の追加のリクエストパラメータ

パラメータ名	デフォルト値	説明
graph	default	追加対象のRDFグラフを識別するURI

<リクエストヘッダ>

追加するTriple を表現する形式を、リクエストヘッダのContent-type で設定する。指定できるパラメータ値とその意味は、「3. 1. 1 SPARQL1. 1準拠のクエリ発行(GET メソッド)」内の別表を参照のこと。

<レスポンス>

成功時、レスポンスボディは空である。

[5-9: Triple の更新]

<機能概要>

Triple を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/rawdata?graph=<graph>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、PUTメソッドのURL部に格納する。更新するtripleは、メッセージボディに格納する。

[表5-12] Triple の更新のリクエストパラメータ

パラメータ名	デフォルト値	説明
graph	(指定なし)	更新対象のRDFグラフを識別するURI

更新は、削除+追加の順に実施する。更新するTriple は、この順にリクエストボディに記述する。削除するTriple と追加するTriple の間に1行以上の空行を入れること。更新するTripleの表現形式がRDF/XMLでない場合、削除するTriple、追加するTriple それぞれが1行以上の空行を含んではない。

<リクエストヘッダ>

更新するTriple の表現形式を、リクエストヘッダのContent-type で設定する。指定できるパラメータ値とその意味は、「3.1.1 SPARQL1.1準拠のクエリ発行(GETメソッド)」内の別表を参照のこと。

<レスポンス>

成功時、レスポンスボディは空である。

――[5-10: Triple の削除]――

<機能概要>

指定したTriple を削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/rawdata?graph=<graph>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、DELETEメソッドのURL部に格納する。

[表5-13] Triple の削除のリクエストパラメータ

パラメータ名	デフォルト値	説明
graph	(指定なし)	削除対象のRDFグラフを識別するURI

<レスポンス>

成功時、レスポンスボディは空である。

6. Traceability/RealTimeData Mngt Command

公共交通情報流通連携基盤システム

Traceability/RealTimeData Management Commandは、トレーサビリティに代表されるイベント管理に必要な機能を提供する。トレーサビリティ管理の対象となる事象をイベント(event)と呼び、それを基本的にucodeで識別する。

――[6-1: イベントの検索]――

<機能概要>

イベントを検索する。

<メソッド>

GET

<URLパス>

/api/v1/events?<param1>=<value1>&<param2>=<value2>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GET メソッドのURL部に格納する。複数のパラメータを指定した場合、AND 検索となる。

[表6-1] イベントの検索のリクエストパラメータ

パラメータ名	デフォルト値	説明
paramN	(指定なし)	検索対象パラメータ名
valueN	(指定なし)	検索対象パラメータ値

offsetとlimitを指定したクエリは、検索結果をイベント発生時刻(ev:date)の新しい順に並べたときのoffset番目からlimit個分を要求することを意味する。ただし、レスポンスがイベント発生時刻(ev:date)順に並んでいることは保証しない。

[表6-2] イベントの検索のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	イベント対象 (ev:target、ev:source、ev:destination) の識別子
source	xsd:anyURI []	イベント発生源 (ev:source) の識別子
destination	xsd:anyURI []	イベント発生の結果生成された (ev:destination) 識別子
owner	xsd:anyURI []	イベント発生者 (ev:owner、ev:startOwner、ev:endOwner) の識別子
after	xsd:dateTime	イベント発生時刻 (ev:date) がこの値より後である
before	xsd:dateTime	イベント発生時刻 (ev:date) がこの値より前である
place	xsd:anyURI []	イベント発行場所 (ev:place) の識別子
description	xsd:string	イベント説明文 (ev:description/部分一致検索)
stream	xsd:integer	Stream API に基づくコネクションを指定された秒数継続する
offset	xsd:integer	検索結果のオフセット値。省略時は最初から返す。
limit	xsd:integer	検索結果の返却数。省略時は情報流通基盤システムが設定する限界数。

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表6-3] イベントの検索のレスポンスパラメータ

パラメータ名	型	説明
events	RDF	イベントのリスト。レスポンス形式にXMLを指定した場合、各イベント情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各イベント情報はRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれないイベントがある場合はtrue、ない場合はfalse。

――[6-2: イベントの新規登録]――

<機能概要>

イベントを新規登録する。発生日時が指定されていない場合、現時刻を発生日時とする。

<メソッド>

POST

<URLパス>

/api/v1/events

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのボディ部に、JSONまたはXML形式で格納する。
(A)か(B)のどちらか1種類のみを指定できるが、(A)を推奨する。

[表6-4] イベントの新規登録のリクエストパラメータ (A)

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSONで記述されたRDFデータ。urn:ucode:_{<val>}という形のURIを含めることにより、ucodeの自動発行を要求できる。

[表6-5] イベントの新規登録のリクエストパラメータ (B)

パラメータ名	型	説明
target	xsd:anyURI	登録するイベント識別子
params	struct[]	キーが登録パラメータ名、値が登録値であるリスト。
numofDestination	xsd:integer	イベント発生の結果発行されるucodeの数

<レスポンス>

クエリパラメータが(A)の形式であった場合のレスポンスは、以下の表(A)に示す構造データをJSONまたはXML形式で表現したものである。

クエリパラメータが(B)の形であった場合のレスポンスは、以下の表(B)に示す構造データをJSONまたはXML形式で表現したものである。

[表6-6] イベントの新規登録のレスポンスパラメータ (A)

パラメータ名	型	説明
ucode	struct[]	キーが指定された変数名、値が発行されたucodeであるリスト。

[表6-7] イベントの新規登録のレスポンスパラメータ (B)

パラメータ名	型	説明
ucode	xsd:anyURI	新規登録されたイベントのucode。
destination	xsd:anyURI []	このイベントにより発行された対象物のucode。

――[6-3: イベントの閲覧]――

<機能概要>

イベントを閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/events/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。また、クエリパラメータとして?streamを指定できる。このときは、Stream APIに基づくコネクションを指定された秒数継続する。

[表6-8] イベントの閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	イベントの識別子

<レスポンス>

レスポンスは、以下の表の構造データをJSONまたはXML形式で表現したものである。

[表6-9] イベントの閲覧のレスポンスパラメータ

パラメータ名	型	説明
events	RDF	イベントのリスト。レスポンス形式にXMLを指定した場合、各イベント情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各イベント情報はRDF/JSONで表現される。

――[6-4: イベントの閲覧(プロパティ指定)]――

<機能概要>

プロパティ値を指定してイベントを閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/events/<target>/<property>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。また、クエリパラメータとして?stream を指定できる。このときは、Stream API に基づくコネクションを指定された秒数継続する。

[表6-10] イベントの閲覧(プロパティ指定)のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	イベントの識別子
property	xsd:anyURI []	プロパティの識別子

<レスポンス>

指定されたイベントに結びついている属性。イベントの識別子もプロパティの識別子も1 種類である場合は、該当する属性値のリストを返す。そうでない場合は、以下の表に示す構造データをJSON またはXML形式で表現したものである。

[表6-11] イベントの閲覧(プロパティ指定)のレスポンスパラメータ

パラメータ名	型	説明
events	RDF	イベントのリスト。レスポンス形式にXMLを指定した場合、各イベント情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各イベント情報はRDF/JSONで表現される。

――[6-5: イベントの更新]――

<機能概要>

イベントを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/events/<target>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子はURI 部分に指定する。それ以外のパラメータ(以下の表)は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表6-12] イベントの更新のリクエストパラメータ

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSON形式で記した更新情報。更新情報のsubject は、<target> と一致していること。コマンド終了後、更新情報に含まれるpredicate に対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。更新情報に含まれないpredicate に関する値は変化しない。
params	struct[]	キーが登録パラメータ名、値が登録値であるリスト。コマンド終了後、キーに指定されたプロパティに対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。リクエストパラメータのキーにないプロパティに関する値は変化しない。

<レスポンス>

成功時、レスポンスボディは空である。

――[6-6: イベントの更新(プロパティ指定)]――

<機能概要>

プロパティ値を指定してイベントを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/events/<target>/<property>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子とプロパティ値はURL 部分に指定する。更新するパラメータ値は、PUTメソッドのボディ部に格納する。

<レスポンス>

成功時、レスポンスボディは空である。

[6-7: イベントの削除]

<機能概要>

イベントを削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/events/<target>

<リクエストパラメータ>

削除対象のイベント識別子はURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[6-8: イベントの削除(プロパティ指定)]

<機能概要>

プロパティ値を指定してイベントを削除する。指定したプロパティ以外のイベント情報は残る。

<メソッド>

DELETE

<URLパス>

/api/v1/events/<target>/<property>

<リクエストパラメータ>

削除対象のイベント識別子とプロパティはURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[6-9: トレースの実施]

<機能概要>

指定したtarget を起点とするトレースフォワード/トレースバックを実施し、その結果得られたイベントのリストを返す。

<メソッド>

GET

<URLパス>

/api/v1/trace/<target>?direction=<direction>&limit=<limit>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表6-13] トレースの実施のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI	トレースの起点となるucode。(イベントまたはイベント対象物ucode)
direction	xsd:string	トレースのパラメータ。forward:(トレースフォワード)またはback:(トレースバック)の値をとる。省略時はforward。
limit	xsd:integer	トレースする階層数。省略時は1階層。

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

7. Geographical Data Management Command

公共交通情報流通連携基盤システム

Geographical Data Management Commandは、GIS 等の地理情報処理に必要な機能を提供するコマンドである。場所や地図は、基本的にucodeで識別する。

――[7-1: 場所情報の検索]――

<機能概要>

場所情報を検索する。

<メソッド>

GET

<URLパス>

/api/v1/places?<param1>=<value1>&<param2>=<value2>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GET メソッドのURL部に格納する。複数のパラメータを指定した場合、AND 検索となる。

[表7-1] 場所情報の検索のリクエストパラメータ

パラメータ名	デフォルト値	説明
paramN	(指定なし)	検索対象パラメータ名
valueN	(指定なし)	検索対象パラメータ値

[表7-2] 場所情報の検索のリクエストパラメータ

パラメータ名	型	説明
lat	xsd:double	WGS84での緯度。省略できない。
lon	xsd:double	WGS84での経度。省略できない。
floor	xsd:double[]	階数。下限値と上限値をカンマで区切って指定する。下限値と上限値が等しい場合は省略した場合は指定なし。(altとどちらか1つのみ指定)
alt	xsd:double[]	高度[m]。下限値と上限値をカンマで区切って指定する。省略した場合は指定なし。(floor とどちらか1つのみ指定)
radius	xsd:double	検索半径[m]。省略できない。

[表7-3] 場所情報の検索のリクエストパラメータ

パラメータ名	型	説明
intersect	xsd:string(WKT)	パラメータ値が指定する領域と重なりを持つ
within	xsd:string(WKT)	パラメータ値が指定する領域に完全に含まれる
contains	xsd:string(WKT)	パラメータ値が指定する領域を完全に含む

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-4] 場所情報の検索のレスポンスパラメータ

パラメータ名	型	説明
place	RDF	場所のリスト。レスポンス形式にXMLを指定した場合、各場所情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各場所情報はRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれない場所がある場合はtrue、ない場合はfalse。

――[7-2: 場所情報の新規登録]――

<機能概要>

場所情報を新規登録する。

<メソッド>

POST

<URLパス>

/api/v1/places

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのボディ部に、JSONまたはXML形式で格納する。
(A)か(B)のどちらか1種類のみを指定できるが、(A)を推奨する。

[表7-5] 場所情報の新規登録のリクエストパラメータ(A)

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSONで記述されたRDFデータ。urn:ucode:_?<val>という形のURIを含めることにより、ucodeの自動発行を要求できる。

[表7-6] 場所情報の新規登録のリクエストパラメータ(B)

パラメータ名	型	説明
target params num	xsd:anyURI [] struct[] xsd:integer	場所情報の識別子 キーがプロパティのURI、値が登録値であるリスト。 新規発行するucodeの個数。省略時は1。

<レスポンス>

クエリパラメータが(A)の形式であった場合のレスポンスは、以下の表(A)に示す構造データをJSONまたはXML形式で表現したものである。
クエリパラメータが(B)の形であった場合のレスポンスは、以下の表(B)に示す構造データをJSONまたはXML形式で表現したものである。

[表7-7] 場所情報の新規登録のレスポンスパラメータ(A)

パラメータ名	型	説明
ucode	struct	キーが指定された変数名、値が発行されたucodeのURI表現であるリスト。

[表7-8] 場所情報の新規登録のレスポンスパラメータ (B)

パラメータ名	型	説明
ucode	xsd:anyURI []	作成された場所情報の識別ucode。

[7-3: 場所情報の閲覧]

<機能概要>

場所情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/places/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-9] 場所情報の閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	場所情報の識別子

<レスポンス>

以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-10] 場所情報の閲覧のレスポンスパラメータ

パラメータ名	型	説明
place	RDF	指定された場所情報のリスト。レスポンス形式にXMLを指定した場合、各場所情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各場所情報はRDF/JSONで表現される。

――[7-4: 場所情報の閲覧(プロパティ指定)]――

<機能概要>

プロパティを指定して、場所情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/places/<target>/<property>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-11] 場所情報の閲覧(プロパティ指定)のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	場所情報の識別子
property	xsd:anyURI []	プロパティ値

<レスポンス>

指定されたプロパティをもつ目的語値。場所の識別子もプロパティの識別子も1種類である場合は、該当する目的語値のリストを返す。そうでない場合は、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-12] 場所情報の閲覧(プロパティ指定)のレスポンスパラメータ

パラメータ名	型	説明
place	RDF	指定された場所情報のリスト。レスポンス形式にXMLを指定した場合、各場所情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各場所情報はRDF/JSONで表現される。

――[7-5: 場所情報の更新]――

<機能概要>

場所情報を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/places/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)のうち、更新対象の識別子はURI 部分に指定する。それ以外のパラメータは、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表7-13] 場所情報の更新のリクエストパラメータ

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSON形式で記した更新情報。更新情報のsubject は、<target> と一致していること。コマンド終了後、リクエストパラメータに含まれるpredicateに対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。
params	struct[]	キーが登録パラメータ名、値が登録値であるリスト。コマンド終了後、キーに指定されたプロパティに対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。リクエストパラメータのキーにないプロパティに関する値は変化しない。

<レスポンス>

成功時、レスポンスボディは空である。

――[7-6: 場所情報の更新(プロパティ指定)]――

<機能概要>

プロパティを指定して、場所情報を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/places/<target>/<property>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子とプロパティはURI 部分に指定する。

更新する値は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

本コマンド終了後、更新対象の識別子URI とプロパティをもつオブジェクトは、リクエストパラメータに指定した値のみとなる。

<レスポンス>

成功時、レスポンスボディは空である。

[7-7: 場所情報の削除]

<機能概要>

場所情報を削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/places/<target>

<リクエストパラメータ>

削除対象の識別子はURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-8: 場所情報の属性削除]

<機能概要>

プロパティを指定して、場所情報の属性を削除する。指定したプロパティ以外の場所情報は残る。

<メソッド>

DELETE

<URLパス>

/api/v1/places/<target>/<property>

<リクエストパラメータ>

削除対象の識別子とプロパティはURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-9: 場所情報の包含関係移設]

<機能概要>

場所情報の包含関係を移設する。これは、「3. 3. 6 場所情報の更新(プロパティ指定)」の特殊ケースである。

<メソッド>

PUT

<URLパス>

/api/v1/places/<target>/ug_consistsOf

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子はURI 部分に指定する。

更新する値は、PUTメソッドのボディ部に格納する。

本コマンドの終了後、<target> に含まれる (ug:consistsOf の関係をもつ) 場所の識別子は、個数を含めてリクエストパラメータに指定した値と完全に一致する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-10: 地図情報の検索]

<機能概要>

地図情報を検索する。

<メソッド>

GET

<URLパス>

/api/v1/maps?<param1>=<value1>&<param2>=<value2>

<リクエストパラメータ>

リクエストパラメータ (以下の表) は、GETメソッドのURL部に格納する。複数のパラメータを指定した場合、AND 検索となる。

[表7-14] 地図情報の検索のリクエストパラメータ

パラメータ名	デフォルト値	説明
paramN	(指定なし)	検索対象パラメータ名
valueN	(指定なし)	検索対象パラメータ値

[表7-15] 地図情報の検索のリクエストパラメータ

パラメータ名	型	説明
lat	xsd:double	WGS84での緯度。省略できない。
lon	xsd:double	WGS84での経度。省略できない。
floor	xsd:double[]	階数。下限値と上限値をカンマで区切って指定する。下限値と上限値が等しい場合は省略した場合は指定なし。(altとどちらか1つのみ指定)
alt	xsd:double[]	高度[m]。下限値と上限値をカンマで区切って指定する。省略した場合は指定なし。(floor とどちらか1つのみ指定)
radius	xsd:double	検索半径[m]。省略できない。

[表7-16] 地図情報の検索のリクエストパラメータ

パラメータ名	型	説明
intersect	xsd:string(WKT)	パラメータ値が指定する領域と重なりを持つ
within	xsd:string(WKT)	パラメータ値が指定する領域に完全に含まれる
contains	xsd:string(WKT)	パラメータ値が指定する領域を完全に含む

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-17] 地図情報の検索のレスポンスパラメータ

パラメータ名	型	説明
maps	RDF	地図のリスト。レスポンス形式にXMLを指定した場合、各地図情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各地図情報はRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれない地図がある場合はtrue、ない場合はfalse。

——[7-11: 地図情報の新規登録]——

<機能概要>

地図情報を新規登録する。

<メソッド>

POST

<URLパス>
/api/v1/maps

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのボディ部に、JSONまたはXML形式で格納する。
(A)か(B)のどちらか1種類のみを指定できるが、(A)を推奨する。

[表7-18] 地図情報の新規登録のリクエストパラメータ(A)

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSONで記述されたRDFデータ。urn:ucode: _?<val>という形のURIを含めることにより、ucodeの自動発行を要求できる。

[表7-19] 地図情報の新規登録のリクエストパラメータ(B)

パラメータ名	型	説明
target params num	xsd:anyURI [] struct[] xsd:integer	地図情報の識別子 キーがプロパティのURI、値が登録値であるリスト。 新規発行するucodeの個数。省略時は1。

<レスポンス>

クエリパラメータが(A)の形式であった場合のレスポンスは、以下の表(A)に示す構造データをJSONまたはXML形式で表現したものである。
クエリパラメータが(B)の形であった場合のレスポンスは、以下の表(B)に示す構造データをJSONまたはXML形式で表現したものである。

[表7-20] 地図情報の新規登録のレスポンスパラメータ(A)

パラメータ名	型	説明
ucode	struct	キーが指定された変数名、値が発行されたucodeのURI表現であるリスト。

[表7-21] 地図情報の新規登録のレスポンスパラメータ (B)

パラメータ名	型	説明
ucode	xsd:anyURI []	作成された地図情報の識別ucode。

[7-12: 地図情報の閲覧]

<機能概要>

地図情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/maps/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-22] 地図情報の閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	地図情報の識別子

<レスポンス>

以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-23] 地図情報の閲覧のレスポンスパラメータ

パラメータ名	型	説明
maps	RDF	指定された地図情報のリスト。レスポンス形式にXMLを指定した場合、各地図情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各地図情報はRDF/JSONで表現される。

[7-13: 地図情報の閲覧(プロパティ指定)]

<機能概要>

プロパティを指定して、地図情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/maps/<target>/<property>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-24] 地図情報の閲覧(プロパティ指定)のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	地図情報の識別子
property	xsd:anyURI []	プロパティ値

<レスポンス>

指定されたプロパティをもつ目的語値。場所の識別子もプロパティの識別子も1種類である場合は、該当する目的語値のリストを返す。そうでない場合は、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-25] 地図情報の閲覧(プロパティ指定)のレスポンスパラメータ

パラメータ名	型	説明
places	RDF	指定された地図情報のリスト。レスポンス形式にXMLを指定した場合、各地図情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各地図情報はRDF/JSONで表現される。

[7-14: 地図情報の更新]

<機能概要>

地図情報を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/maps/<target>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子はURI 部分に指定する。それ以外のパラメータは、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表7-26] 地図情報の更新のリクエストパラメータ

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSON形式で記した更新情報。更新情報のsubject は、<target> と一致していること。コマンド終了後、更新情報に含まれるpredicate に対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。更新情報に含まれないpredicate に関する値は変化しない。
params	struct[]	キーが登録パラメータ名、値が登録値であるリスト。コマンド終了後、キーに指定されたプロパティに対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。リクエストパラメータのキーにないプロパティに関する値は変化しない。

<レスポンス>

成功時、レスポンスボディは空である。

――[7-15: 地図情報の更新(プロパティ指定)]――

<機能概要>

プロパティを指定して、地図情報を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/maps/<target>/<property>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子とプロパティはURI 部分に指定する。更新する値は、PUTメソッドのボディ部に格納する。

本コマンド終了後、更新対象の識別子URI とプロパティをもつオブジェクトは、リクエストパラメータに指定した値のみとなる。

<レスポンス>

成功時、レスポンスボディは空である。

[7-16: 地図情報の削除]

<機能概要>

地図情報を削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/maps/<target>

<リクエストパラメータ>

削除対象の識別子はURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-17: 地図情報の属性削除]

<機能概要>

プロパティを指定して、地図情報の属性を削除する。指定したプロパティ以外の地図情報は残る。

<メソッド>

DELETE

<URLパス>

/api/v1/maps/<target>/<property>

<リクエストパラメータ>

削除対象の識別子とプロパティはURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-18: ルールの検索]

<機能概要>

ルールを検索する。

<メソッド>

GET

<URLパス>

/api/v1/rules?<param1 >=<value1 >¶m2=<value2 >

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。複数のパラメータを指定した場合、AND 検索となる。

[表7-27] ルールの検索のリクエストパラメータ

パラメータ名	デフォルト値	説明
paramN	(指定なし)	検索対象パラメータ名
valueN	(指定なし)	検索対象パラメータ値

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-28] ルールの検索のレスポンスパラメータ

パラメータ名	型	説明
rules	RDF	ルールのリスト。レスポンス形式にXMLを指定した場合、各ルール情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各ルール情報はRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれないルールがある場合はtrue、ない場合はfalse。

――[7-19: ルールの新規登録]――

<機能概要>

ルールを新規登録する。

<メソッド>

POST

<URLパス>

/api/v1/rules

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのボディ部に、JSONまたはXML形式で格納する。

(A)か(B)のどちらか1種類のみを指定できるが、(A)を推奨する。

[表7-29] ルールの新規登録のリクエストパラメータ (A)

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSONで記述されたRDFデータ。urn:ucode: _?<val>という形のURIを含めることにより、ucodeの自動発行を要求できる。

[表7-30] ルールの新規登録のリクエストパラメータ (B)

パラメータ名	型	説明
target params num	xsd:anyURI [] struct [] xsd:integer	ルールの識別子 キーがプロパティのURI、値が登録値であるリスト。 新規発行するucodeの個数。省略時は1。

<レスポンス>

クエリパラメータが(A)の形式であった場合のレスポンスは、以下の表(A)に示す構造データをJSONまたはXML形式で表現したものである。

クエリパラメータが(B)の形であった場合のレスポンスは、以下の表(B)に示す構造データをJSONまたはXML形式で表現したものである。

[表7-31] ルールの新規登録のレスポンスパラメータ (A)

パラメータ名	型	説明
ucode	struct	キーが指定された変数名、値が発行されたucodeのURI表現であるリスト。

[表7-32] ルールの新規登録のレスポンスパラメータ (B)

パラメータ名	型	説明
ucode	xsd:anyURI []	作成されたルールの識別ucode。

——[7-20: ルールの閲覧]——

<機能概要>

ルールを閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/rules/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-33] ルールの閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	ルールの識別子

<レスポンス>

以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-34] ルールの閲覧のレスポンスパラメータ

パラメータ名	型	説明
rules	RDF	指定されたルール情報のリスト。レスポンス形式にXMLを指定した場合、各ルール情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各ルール情報はRDF/JSONで表現される。

——[7-21: ルールの閲覧(プロパティ指定)]——

<機能概要>

プロパティを指定して、ルールを閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/rules/<target>/<property>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。property パラメータとして、ルール文書XMLのキーを指定してもよい。

[表7-35] ルールの閲覧(プロパティ指定)のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	ルール情報の識別子
property	xsd:anyURI []	プロパティ値

<レスポンス>

指定されたプロパティをもつ目的語値。

ルールの識別子もプロパティの識別子も1種類である場合は、該当する目的語値のリストを返す。

そうでない場合は、以下の表の構造データをJSONまたはXML形式で表現したものである。

[表7-36] ルールの閲覧(プロパティ指定)のレスポンスパラメータ

パラメータ名	型	説明
place	RDF	指定されたルール情報のリスト。レスポンス形式にXMLを指定した場合、各ルール情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各ルール情報はRDF/JSONで表現される。

——[7-22: ルールの更新]——

<機能概要>

ルールを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/rules/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)のうち、更新対象の識別子はURI 部分に指定する。それ以外のパラメータは、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表7-37] ルールの更新のリクエストパラメータ

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSON形式で記した更新情報。更新情報のsubject は、<target> と一致していること。コマンド終了後、リクエストパラメータに含まれるpredicateに対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。
rdf	RDF	RDF/XMLまたはRDF/JSON形式で記した更新情報。更新情報のsubject は、<target> と一致していること。コマンド終了後、更新情報に含まれるpredicate に対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。更新情報に含まれないpredicate に関する値は変化しない。

<レスポンス>

成功時、レスポンスボディは空である。

——[7-23: ルールの更新(プロパティ指定)]——

<機能概要>

プロパティを指定して、ルールを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/rules/<target>/<property>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子とプロパティはURI 部分に指定する。プロパティ名として、ルールXML文書のキーを指定してもよい。更新する値は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

本コマンド終了後、更新対象の識別子URI とプロパティをもつオブジェクトは、リクエストパラメータに指定した値のみとなる。

<レスポンス>

成功時、レスポンスボディは空である。

[7-24: ルールの削除]

<機能概要>

ルールを削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/rules/<target>

<リクエストパラメータ>

削除対象の識別子はURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-25: ルール属性の削除]

<機能概要>

プロパティを指定して、ルールの属性情報を削除する。指定したプロパティ以外のルール属性情報は残る。

<メソッド>

DELETE

<URLパス>

/api/v1/rules/<target>/<property>

<リクエストパラメータ>

削除対象の識別子とプロパティはURI 部分に指定する。プロパティとして、ルールXML文書のキーを指定してもよい。ただしpermission は指定できない。

<レスポンス>

成功時、レスポンスボディは空である。

[7-26: ルールの適用先閲覧]

<機能概要>

ルールの適用先を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/rules/<target>/applied

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-38] ルールの適用先閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	ルールの識別子

<レスポンス>

このルールを適用されているオープンデータの識別子のリストを、XMLまたはJSON形式で返す。

――[7-27: ルールの適用]――

<機能概要>

ルールをオープンデータに適用する。これと同じことは、オープンデータの属性値として dcterms:accessRights を追加するオペレーションを利用しても実現できる。

<メソッド>

PUT

<URLパス>

/api/v1/rules/<target>/applied

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子はURI 部分に指定する。適用先のオープンデータ識別子のリストは、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。
本コマンド終了後、適用先のオープンデータ識別子のリストは、リクエストパラメータに指定した値のみとなる。

<レスポンス>

成功時、レスポンスボディは空である。

[7-28: ルールの適用解除]

<機能概要>

指定したオープンデータを、このルールの適用対象から外す。これと同じことは、オープンデータの属性値としてdcterms:accessRights を更新するオペレーションを利用しても実現できる。

<メソッド>

DELETE

<URLパス>

/api/v1/rules/<target>/applied

<リクエストパラメータ>

ルールの識別子はURI 部分に指定する。適用解除対象のオープンデータの識別子は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-29: ユーザの検索]

<機能概要>

ユーザを検索する。

<メソッド>

GET

<URLパス>

/api/v1/users?<param1 >=<value1 >¶m2=<value2 >

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。複数のパラメータを指定した場合、AND 検索となる。

[表7-39] ユーザの検索のリクエストパラメータ

パラメータ名	デフォルト値	説明
paramN	(指定なし)	検索対象パラメータ名
valueN	(指定なし)	検索対象パラメータ値

<レスポンス>

レスポンスは、以下の表の構造データをJSONまたはXML形式で表現したものである。

[表7-40] ユーザの検索のレスポンスパラメータ

パラメータ名	型	説明
place	RDF	ユーザのリスト。レスポンス形式にXMLを指定した場合、各ユーザ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各ユーザ情報はRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれないユーザがある場合はtrue、ない場合はfalse。

――[7-30: ユーザの新規登録]――

<機能概要>

ユーザを新規登録する。

<メソッド>

POST

<URLパス>

/api/v1/users

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのボディ部に、JSONまたはXML形式で格納する。
(A)か(B)のどちらか1種類のみを指定できるが、(A)を推奨する。

[表7-41] ユーザの新規登録のリクエストパラメータ (A)

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSONで記述されたRDFデータ。urn:ucode: _?<val>という形のURIを含めることにより、ucodeの自動発行を要求できる。

[表7-42] ユーザの新規登録のリクエストパラメータ (B)

パラメータ名	型	説明
target params num	xsd:anyURI [] struct [] xsd:integer	ユーザの識別子 キーがプロパティのURI、値が登録値であるリスト。 新規発行するucodeの個数。省略時は1。

<レスポンス>

クエリパラメータが(A)の形式であった場合のレスポンスは、以下の表(A)に示す構造データをJSONまたはXML形式で表現したものである。

クエリパラメータが(B)の形であった場合のレスポンスは、以下の表(B)に示す構造データをJSONまたはXML形式で表現したものである。

[表7-43] ユーザの新規登録のレスポンスパラメータ (A)

パラメータ名	型	説明
ucode	struct	キーが指定された変数名、値が発行されたucodeのURI表現であるリスト。

[表7-44] ユーザの新規登録のレスポンスパラメータ (B)

パラメータ名	型	説明
ucode	xsd:anyURI []	作成されたユーザの識別ucode。

——[7-31: ユーザ情報の閲覧]——

<機能概要>

ユーザ情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/users/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-45] ユーザ情報の閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	ユーザの識別子

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-46] ユーザ情報の閲覧のレスポンスパラメータ

パラメータ名	型	説明
users	RDF	指定されたユーザのリスト。レスポンス形式にXMLを指定した場合、各ユーザ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各ユーザ情報はRDF/JSONで表現される。

——[7-32: ユーザ情報の閲覧(プロパティ指定)]——

<機能概要>

プロパティを指定して、ユーザ情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/users/<target>/<property>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-47] ユーザ情報の閲覧(プロパティ指定)のリクエストパラメータ

パラメータ名	型	説明
target property	xsd:anyURI [] xsd:anyURI []	ユーザの識別子 プロパティ値

<レスポンス>

指定されたプロパティをもつ目的語値。ユーザの識別子もプロパティの識別子も1種類である場合は、該当する目的語値のリストを返す。そうでない場合は、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-48] ユーザ情報の閲覧(プロパティ指定)のレスポンスパラメータ

パラメータ名	型	説明
users	RDF	指定されたユーザのリスト。レスポンス形式にXMLを指定した場合、各ユーザ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各ユーザ情報はRDF/JSONで表現される。

――[7-33: ユーザの更新]――

<機能概要>

ユーザを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/users/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)のうち、更新対象の識別子はURI 部分に指定する。それ以外のパラメータは、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表7-49] ユーザの更新のリクエストパラメータ

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSON形式で記した更新情報。更新情報のsubject は、<target> と一致していること。コマンド終了後、更新情報に含まれるpredicate に対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。更新情報に含まれないpredicate に関する値は変化しない。
params	struct[]	キーが登録パラメータ名、値が登録値であるリスト。コマンド終了後、キーに指定されたプロパティに対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。リクエストパラメータのキーにないプロパティに関する値は変化しない。

<レスポンス>

成功時、レスポンスボディは空である。

——[7-34: ユーザの更新(プロパティ指定)]——

<機能概要>

プロパティを指定して、ユーザを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/users/<target>/<property>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子とプロパティはURI 部分に指定する。更新する値は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

本コマンド終了後、更新対象の識別子URI とプロパティをもつオブジェクトは、リクエストパラメータに指定した値のみとなる。

<レスポンス>

成功時、レスポンスボディは空である。

[7-35: ユーザの削除]

<機能概要>

ユーザを削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/users/<target>

<リクエストパラメータ>

削除対象の識別子はURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-36: ユーザの属性情報削除]

<機能概要>

プロパティを指定して、ユーザの属性情報を削除する。指定したプロパティ以外のユーザ情報は残る。

<メソッド>

DELETE

<URLパス>

/api/v1/users/<target>/<property>

<リクエストパラメータ>

削除対象の識別子とプロパティはURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-37: グループの検索]

<機能概要>

グループを検索する。

<メソッド>

GET

<URLパス>

/api/v1/groups?<param1 >=<value1 >¶m2=<value2 >

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。複数のパラメータを指定した場合、AND 検索となる。

[表7-50] グループの検索のリクエストパラメータ

パラメータ名	デフォルト値	説明
paramN	(指定なし)	検索対象パラメータ名
valueN	(指定なし)	検索対象パラメータ値

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-51] グループの検索のレスポンスパラメータ

パラメータ名	型	説明
groups	RDF	グループのリスト。レスポンス形式にXMLを指定した場合、各グループ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各グループ情報はRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれない場所がある場合はtrue、ない場合はfalse。

――[7-38: グループの新規登録]――

<機能概要>

グループを新規登録する。

<メソッド>

POST

<URLパス>

/api/v1/groups

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのボディ部に、JSONまたはXML形式で格納する。
(A)か(B)のどちらか1種類のみを指定できるが、(A)を推奨する。

[表7-52] グループの新規登録のリクエストパラメータ (A)

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSONで記述されたRDFデータ。urn:ucode:_?<val>という形のURIを含めることにより、ucodeの自動発行を要求できる。

[表7-53] グループの新規登録のリクエストパラメータ (B)

パラメータ名	型	説明
target params num	xsd:anyURI [] struct [] xsd:integer	グループの識別子 キーがプロパティのURI、値が登録値であるリスト。 新規発行するucodeの個数。省略時は1。

<レスポンス>

クエリパラメータが(A)の形式であった場合のレスポンスは、以下の表(A)に示す構造データをJSONまたはXML形式で表現したものである。

クエリパラメータが(B)の形であった場合のレスポンスは、以下の表(B)に示す構造データをJSONまたはXML形式で表現したものである。

[表7-54] グループの新規登録のレスポンスパラメータ (A)

パラメータ名	型	説明
ucode	struct	キーが指定された変数名、値が発行されたucodeのURI表現であるリスト。

[表7-55] グループの新規登録のレスポンスパラメータ (B)

パラメータ名	型	説明
ucode	xsd:anyURI []	作成されたグループの識別ucode。

——[7-39: グループ情報の閲覧]——

<機能概要>

グループ情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/groups/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-56] グループ情報の閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	グループの識別子

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-57] グループ情報の閲覧のレスポンスパラメータ

パラメータ名	型	説明
groups	RDF	指定されたグループのリスト。レスポンス形式にXMLを指定した場合、各グループ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各グループ情報はRDF/JSONで表現される。

——[7-40: グループ情報の閲覧(プロパティ指定)]——

<機能概要>

プロパティを指定して、グループを閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/groups/<target>/<property>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表7-58] グループ情報の閲覧(プロパティ指定)のリクエストパラメータ

パラメータ名	型	説明
target property	xsd:anyURI [] xsd:anyURI []	グループの識別子 プロパティ値

<レスポンス>

指定されたプロパティをもつ目的語値。グループの識別子もプロパティの識別子も1種類である場合は、該当する目的語値のリストを返す。そうでない場合は、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表7-59] グループ情報の閲覧(プロパティ指定)のレスポンスパラメータ

パラメータ名	型	説明
groups	RDF	指定されたグループのリスト。レスポンス形式にXMLを指定した場合、各グループ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各グループ情報はRDF/JSONで表現される。

――[7-41: グループの更新]――

<機能概要>

グループを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/groups/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)のうち、更新対象の識別子はURI 部分に指定する。それ以外のパラメータは、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表7-60] グループの更新のリクエストパラメータ

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSON形式で記した更新情報。更新情報のsubject は、<target> と一致していること。コマンド終了後、更新情報に含まれるpredicate に対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。更新情報に含まれないpredicate に関する値は変化しない。
params	struct[]	キーが登録パラメータ名、値が登録値であるリスト。コマンド終了後、キーに指定されたプロパティに対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。リクエストパラメータのキーにないプロパティに関する値は変化しない。

<レスポンス>

成功時、レスポンスボディは空である。

——[7-42: グループの更新(プロパティ 指定)]——

<機能概要>

プロパティを指定して、グループを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/groups/<target>/<property>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象の識別子とプロパティはURI 部分に指定する。更新する値は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

本コマンド終了後、更新対象の識別子URI とプロパティをもつオブジェクトは、リクエストパラメータに指定した値のみとなる。

<レスポンス>

成功時、レスポンスボディは空である。

[7-43: グループの削除]

<機能概要>

グループを削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/groups/<target>

<リクエストパラメータ>

削除対象の識別子はURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[7-44: グループの属性情報削除]

<機能概要>

プロパティを指定して、グループの属性情報を削除する。指定したプロパティ以外のユーザ情報は残る。

<メソッド>

DELETE

<URLパス>

/api/v1/groups/<target>/<property>

<リクエストパラメータ>

削除対象の識別子とプロパティはURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

8. Notification Management Command

公共交通情報流通連携基盤システム

Notification とは、対象としているオープンデータが登録・更新され、指定した条件を満たした場合にユーザプログラムにコールバックする仕組みである。コールバック先は、URLで指定する。

――[8-1: Notification の検索]――

<機能概要>

Notification を検索する。

ただし、閲覧権限のないNotification は検索できない。

<メソッド>

GET

<URLパス>

/api/v1/triggers?<param1 >=<value1 >&<param2 >=<value2 >

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表8-1] Notification の検索のリクエストパラメータ

パラメータ名	デフォルト値	説明
paramN	(指定なし)	検索対象パラメータ名
valueN	(指定なし)	検索対象パラメータ値

<レスポンス>

レスポンスは、以下の表の構造データをJSONまたはXML形式で表現したものである。

[表8-2] Notification の検索のレスポンスパラメータ

パラメータ名	型	説明
triggers	struct[]	検索の結果得られたNotification。
—ucode	xsd:anyURI	Notification のucode。
—name	xsd:string	Notification 名。
—target	xsd:anyURI []	発動対象のオープンデータ識別子
—type	xsd:string	Notification の発動条件。詳細は別表を参照のこと。
—value	xsd:string	発動条件の閾値。
—callback	xsd:string	コールバックURL。
—run	xsd:boolean	稼働中ならばtrue、そうでなければfalse
—(*)	(*)	以下、該当するNotificationに結びついている属性を返す。パラメータ名はNotification属性を示すプロパティである。
remains	xsd:boolean	検索条件にマッチするが、レスポンスに含めていないNotificationがあればtrue、なければfalse

——[8-2: Notification の新規作成]——

<機能概要>

Notification を新規作成する。

<メソッド>

POST

<URLパス>

/api/v1/triggers

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表8-3] Notification の新規作成のリクエストパラメータ

パラメータ名	型	説明
params	struct[]	キーが登録パラメータ名、値が登録値であるリスト。

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表8-4] Notification の新規作成のレスポンスパラメータ

パラメータ名	型	説明
ucode	xsd:anyURI []	作成されたNotification のucode。

――[8-3: Notification 情報の閲覧]――

<機能概要>

Notification 情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/triggers/<ucode>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表8-5] Notification 情報の閲覧のリクエストパラメータ

パラメータ名	型	説明
ucode	xsd:anyURI []	Notification の識別子

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表8-6] Notification 情報の閲覧のレスポンスパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	発動対象のオープンデータ識別子
type	xsd:string	Notification の発動条件。詳細は別表を参照のこと。
value	xsd:string	発動条件の閾値。
callback	xsd:string	コールバックURL。
run	xsd:boolean	稼働していればtrue、そうでなければfalse
(*)	(*)	以下該当するNotificationに結びついている属性を返す。パラメータ名はNotification属性を示すプロパティである。

[8-4: Notification 情報の更新]

<機能概要>

Notification 情報を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/triggers/<ucode>

<リクエストパラメータ>

リクエストパラメータ(以下の表)のうち、更新対象Notification のucodeはURL 部分に指定する。
それ以外の下記パラメータは、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表8-7] Notification 情報の更新のリクエストパラメータ

パラメータ名	型	説明
params	struct[]	キーがパラメータ名、値が登録値であるリスト。

<レスポンス>

成功時、レスポンスボディは空である。

[8-5: Notification の削除]

<機能概要>

Notification を削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/triggers/<ucode>

<リクエストパラメータ>

削除対象Notification のucodeはURL 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[8-6: Notification の起動・再開]

<機能概要>

Notification を起動または再開する。

<メソッド>

PUT

<URLパス>

/api/v1/triggers/<ucode>/run

<リクエストパラメータ>

リクエストパラメータのうち、更新対象Notification のucodeはURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[8-7: Notification の停止]

<機能概要>

Notification を停止する。

<メソッド>

DELETE

<URLパス>

/api/v1/triggers/<ucode>/run

<リクエストパラメータ>

削除対象Notification のucodeはURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

9. Vocabulary Management Command

公共交通情報流通連携基盤システム

Vocabulary Management Commandは、ボキャブラリ管理機能を実現するためのコマンドである。ボキャブラリは、RDF Schema 形式に基づいて入出力する。

――[9-1: ボキャブラリの検索]――

<機能概要>

ボキャブラリを検索する。

<メソッド>

GET

<URLパス>

/api/v1/vocabularies?<param1 >=<value1 >&<param2 >=<value2 >

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表9-1] ボキャブラリの検索のリクエストパラメータ

パラメータ名	デフォルト値	説明
paramN	(指定なし)	検索対象パラメータ名
valueN	(指定なし)	検索対象パラメータ値

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表9-2] ボキャブラリの検索のレスポンスパラメータ

パラメータ名	型	説明
vocabularies	RDF	RDF Schema 形式に基づくボキャブラリのリスト。レスポンス形式にXMLを指定した場合、各ボキャブラリ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各ボキャブラリ情報はRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれないボキャブラリがある場合はtrue、ない場合はfalse。

――[9-2: ボキャブラリの新規作成]――

<機能概要>

ボキャブラリを新規作成する。

<メソッド>

POST

<URLパス>

/api/v1/vocabularies

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのボディ部に、JSONまたはXML形式で格納する。
(A)か(B)のどちらか1種類のみを指定できるが、(A)を推奨する。

[表9-3] ボキャブラリの新規作成のリクエストパラメータ(A)

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSONで記述されたRDFデータ。urn:ucode:_{<val>}という形のURIを含めることにより、ucodeの自動発行を要求できる。

[表9-4] ボキャブラリの新規作成のリクエストパラメータ (B)

パラメータ名	型	説明
target	xsd:anyURI []	ボキャブラリの識別子
schema	xsd:string	RDF schema で記述されたボキャブラリ定義文書。
params	struct[]	キーがプロパティのURI、値が登録値であるリスト。
num	xsd:integer	新規発行するucodeの個数。省略時は1。

<レスポンス>

クエリパラメータが(A)の形式であった場合のレスポンスは、以下の表(A)に示す構造データをJSONまたはXML形式で表現したものである。

クエリパラメータが(B)の形であった場合のレスポンスは、以下の表(B)に示す構造データをJSONまたはXML形式で表現したものである。

[表9-5] ボキャブラリの新規作成のレスポンスパラメータ (A)

パラメータ名	型	説明
ucode	struct	キーが指定された変数名、値が発行されたucodeのURI表現であるリスト。

[表9-6] ボキャブラリの新規作成のレスポンスパラメータ (B)

パラメータ名	型	説明
ucode	xsd:anyURI []	作成されたボキャブラリの識別ucode。

――[9-3: ボキャブラリの閲覧]――

<機能概要>

ボキャブラリ情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/vocabularies/<target>

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。

[表9-7] ボキャブラリの閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	ボキャブラリの識別子

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表9-8] ボキャブラリの閲覧のレスポンスパラメータ

パラメータ名	型	説明
vocabularies	RDF	RDF Schema に基づく、指定されたボキャブラリのリスト。レスポンス形式にXMLを指定した場合、各ボキャブラリ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各ボキャブラリ情報はRDF/JSONで表現される。

――[9-4: ボキャブラリ情報の閲覧(プロパティ指定)]――

<機能概要>

ボキャブラリ情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/vocabularies/<target>/<property>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表9-9] ボキャブラリ情報の閲覧(プロパティ指定)のリクエストパラメータ

パラメータ名	型	説明
target property	xsd:anyURI [] xsd:anyURI []	ボキャブラリの識別子 プロパティ値

<レスポンス>

指定されたプロパティをもつ目的語値。ボキャブラリの識別子もRDF schema のpredicateも1種類である場合は、該当する目的語値のリストを返す。そうでない場合は、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表9-10] ボキャブラリ情報の閲覧(プロパティ指定)のレスポンスパラメータ

パラメータ名	型	説明
vocabularies	RDF	指定されたボキャブラリ情報のリスト。レスポンス形式にXMLを指定した場合、各ボキャブラリ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各ボキャブラリ情報はRDF/JSONで表現される。

――[9-5: ボキャブラリ情報の更新]――

<機能概要>

ボキャブラリ情報を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/vocabularies/<target>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象ボキャブラリ<target> はURI 部分に指定する。それ以外のパラメータ(以下の表)は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表9-11] ボキャブラリ情報の更新のリクエストパラメータ

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSON形式で記した更新情報。更新情報のsubject は、<target> と一致していること。コマンド終了後、更新情報に含まれるpredicate に対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。更新情報に含まれないpredicate に関する値は変化しない。
params	struct[]	キーが登録パラメータ名、値が登録値であるリスト。コマンド終了後、キーに指定されたpredicate に対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。リクエストパラメータのキーにないpredicate に関する値は変化しない。
schema	xsd:string	RDF schema で記述されたボキャブラリ定義文書。コマンド終了後のボキャブラリ情報は、指定したボキャブラリ定義文書と完全に一致する。指定したボキャブラリ定義文書にないpredicate に関する値は削除される。

<レスポンス>

成功時、レスポンスボディは空である。

――[9-6: ボキャブラリ情報の更新(プロパティ指定)]――

<機能概要>

プロパティを指定して、ボキャブラリ情報を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/vocabularies/<target>/<property>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象ボキャブラリ<target>とプロパティ<property>はURI 部分に指定する。更新する値を、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。本コマンド終了後、更新対象ボキャブラリ<target>とプロパティ<property>をもつオブジェクトは、リクエストパラメータに指定した値のみとなる。

<レスポンス>

成功時、レスポンスボディは空である。

[9-7: ボキャブラリの削除]

<機能概要>

ボキャブラリを削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/vocabularies/<target>

<リクエストパラメータ>

削除対象ボキャブラリはURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[9-8: 同意語の検索]

<機能概要>

指定したボキャブラリの同意語(owl:sameAs で結ばれているボキャブラリ)を検索する。

<メソッド>

GET

<URLパス>

/api/v1/vocabularies/<target>/synonyms

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表9-12] 同意語の検索のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI	ボキャブラリの識別子

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表9-13] 同意語の検索のレスポンスパラメータ

パラメータ名	型	説明
vocabularies	xsd:anyURI []	指定したボキャブラリの同意語のリスト。

[9-9: 同意語情報の更新]

<機能概要>

ボキャブラリの同意語情報を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/vocabularies/<target>/synonyms

<リクエストパラメータ>

リクエストパラメータのうち、更新対象ボキャブラリ<target> はURI 部分に指定する。それ以外のパラメータ(以下の表)は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表9-14] 同意語情報の更新のリクエストパラメータ

パラメータ名	型	説明
synonyms	xsd:anyURI []	指定したボキャブラリの同意語ucodeのリスト。

<レスポンス>

成功時、レスポンスボディは空である。

[9-10: 親ボキャブラリの検索]

<機能概要>

指定したボキャブラリの親ボキャブラリ(rdfs:subPropertyOf, rdfs:subClassOf の先にあるリソース)を検索する。

<メソッド>

GET

<URLパス>

/api/v1/vocabularies/<target>/parents

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表9-15] 親ボキャブラリの検索のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI	ボキャブラリの識別子。

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表9-16] 親ボキャブラリの検索のレスポンスパラメータ

パラメータ名	型	説明
vocabularies	xsd:anyURI []	指定したボキャブラリの親ボキャブラリのリスト。

——[9-11: 親ボキャブラリ情報の更新]——

<機能概要>

ボキャブラリの親ボキャブラリ(rdfs:subPropertyOf, rdfs:subClassOf の先にあるリソース)を更新する。

<メソッド>

PUT

<URLパス>

/api/v1/vocabularies/<target>/parents

<リクエストパラメータ>

リクエストパラメータのうち、更新対象ボキャブラリ<target> はURI 部分に指定する。親ボキャブラリは、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

本コマンド終了後、更新対象ボキャブラリの親ボキャブラリは、リクエストパラメータに指定した値のみとなる。

<レスポンス>

成功時、レスポンスボディは空である。

——[9-12: 子ボキャブラリの検索]——

<機能概要>

指定したボキャブラリの子ボキャブラリ (rdfs:subPropertyOf, rdfs:subClassOf の前にあるリソース) を検索する。

<メソッド>

GET

<URLパス>

/api/v1/vocabularies/<target>/children

<リクエストパラメータ>

リクエストパラメータ (以下の表) は、GETメソッドのURL部に格納する。

[表9-17] 子ボキャブラリの検索のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI	ボキャブラリの識別子。

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表9-18] 子ボキャブラリの検索のレスポンスパラメータ

パラメータ名	型	説明
vocabularies	xsd:anyURI []	指定したボキャブラリの子ボキャブラリのリスト。

10. Triple Management Command

公共交通情報流通連携基盤システム

Triple Management Commandは、センサやスマートメータなどの小型機器がRDFモデルの主語・述語・目的語からなるTriple を効率的に扱うために、利用者プログラムが「標準データ規格」を簡素化したオープンデータ操作を行うためのコマンドである。

——[10-1: オープンデータ検索]——

<機能概要>

オープンデータを検索する。

<メソッド>

GET

<URLパス>

/api/v1/datapoints?<param1 >=<value1 >&<param2 >=<value2 >

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表10-1] オープンデータ検索のリクエストパラメータ

パラメータ名	デフォルト値	説明
paramN	(指定なし)	検索対象パラメータ名
valueN	(指定なし)	検索対象パラメータ値

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表10-2] オープンデータ検索のレスポンスパラメータ

パラメータ名	型	説明
results	RDF	検索の結果得られたオープンデータ。レスポンス形式にXMLを指定した場合、データはRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、データはRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれない場所がある場合はtrue、ない場合はfalse。

——[10-2: オープンデータの新規作成]——

<機能概要>

オープンデータを新規作成する。

<メソッド>

POST

<URLパス>

/api/v1/datapoints

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのボディ部に、JSONまたはXML形式で格納する。
(A)か(B)のどちらか1種類のみを指定できるが、(A)を推奨する。

[表10-3] オープンデータの新規作成のリクエストパラメータ(A)

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSONで記述されたRDFデータ。urn:ucode:_?<val>という形のURIを含めることにより、ucodeの自動発行を要求できる。

[表10-4] オープンデータの新規作成のリクエストパラメータ (B)

パラメータ名	型	説明
target	xsd:anyURI []	オープンデータの識別子 キーがプロパティのURI、値が登録値であるリスト。 新規発行するucodeの個数。省略時は1。
params	struct[]	
num	xsd:integer	

<レスポンス>

クエリパラメータが(A)の形式であった場合のレスポンスは、以下の表(A)に示す構造データをJSONまたはXML形式で表現したものである。

クエリパラメータが(B)の形であった場合のレスポンスは、以下の表(B)に示す構造データをJSONまたはXML形式で表現したものである。

[表10-5] オープンデータの新規作成のレスポンスパラメータ (A)

パラメータ名	型	説明
ucode	struct	キーが指定された変数名、値が発行されたucodeのURI表現であるリスト。

[表10-6] オープンデータの新規作成のレスポンスパラメータ (B)

パラメータ名	型	説明
ucode	xsd:anyURI []	作成されたオープンデータの識別ucode。

——[10-3: オープンデータの閲覧]——

<機能概要>

オープンデータを閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/datapoints/<target>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

また、クエリパラメータとして?stream を指定できる。このときは、Stream API に基づくコネクションを指定された秒数継続する(「2.8 Streams API」参照)。

[表10-7] オープンデータの閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	オープンデータの識別子

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表10-8] オープンデータの閲覧のレスポンスパラメータ

パラメータ名	型	説明
results	RDF	指定されたデータのリスト。レスポンス形式にXMLを指定した場合、各データはRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各データはRDF/JSONで表現される。

——[10-4: オープンデータの閲覧(プロパティ指定)]——

<機能概要>

プロパティを指定して、オープンデータを閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/datapoints/<target>/<property>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。また、クエリパラメータとして?stream を指定できる。このときは、Stream API に基づくコネクションを指定された秒数継続する(「2.8 Streams API」参照)。

[表10-9] オープンデータの閲覧(プロパティ指定)のリクエストパラメータ

パラメータ名	型	説明
target property	xsd:anyURI [] xsd:anyURI []	オープンデータの識別子 プロパティ値

<レスポンス>

指定されたプロパティをもつ目的語値。オープンデータの識別子もプロパティの識別子も1種類である場合は、該当する目的語値のリストを返す。そうでない場合は、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表10-10] オープンデータの閲覧(プロパティ指定)のレスポンスパラメータ

パラメータ名	型	説明
results	RDF	指定されたデータのリスト。レスポンス形式にXMLを指定した場合、各データはRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各データはRDF/JSONで表現される。

——[10-5: オープンデータの更新]——

<機能概要>

オープンデータを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/datapoints/<target>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象オープンデータのucodeはURI 部分に指定する。それ以外のパラメータ(以下の表)は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表10-11] オープンデータの更新のリクエストパラメータ

パラメータ名	型	説明
rdf	RDF	RDF/XMLまたはRDF/JSON形式で記した更新情報。更新情報のsubject は、<target> と一致していること。コマンド終了後、リクエストパラメータに含まれるpredicateに対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。
params	struct[]	キーが登録パラメータ名、値が登録値であるリスト。コマンド終了後、キーに指定されたプロパティに対する値は、個数を含めてリクエストパラメータが指定した情報と完全に一致する。リクエストパラメータのキーにないプロパティに関する値は変化しない。

<レスポンス>

成功時、レスポンスボディは空である。

——[10-6: オープンデータの更新(プロパティ指定)]——

<機能概要>

プロパティを指定して、オープンデータを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/datapoints/<target>/<property>

<リクエストパラメータ>

リクエストパラメータのうち、更新対象オープンデータ識別子<target> とプロパティ<property> はURI 部分に指定する。更新値は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

本コマンド終了後、更新対象の識別子URI とプロパティをもつオブジェクトは、リクエストパラメータに指定した値のみとなる。

<レスポンス>

成功時、レスポンスボディは空である。

[10-7: オープンデータの削除]

<機能概要>

オープンデータを削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/datapoints/<target>

<リクエストパラメータ>

削除対象オープンデータ<target> はURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

[10-8: オープンデータの属性削除]

<機能概要>

オープンデータの指定した属性を削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/datapoints/<target>/<property>

<リクエストパラメータ>

削除対象オープンデータのucodeと属性はURI 部分に指定する。

<レスポンス>

成功時、レスポンスボディは空である。

11. Identification Resolution Command

公共交通情報流通連携基盤システム

Identification Resolution Commandは、利用者プログラムが、モノ・場所・データの識別子から、その識別子が指し示す対象に関するオープンデータが格納されているサーバを得るための、ディレクトリ型検索の機能を提供するコマンドである。

――[11-1: 簡易ucode解決]――

<機能概要>

簡易ucode解決プロトコル に基づいたucode解決機能を提供する。すなわち、ucodeに結びついた情報の参照先を取得する。

<メソッド>

GET

<URLパス>

/api/v1/rs/<ucode>?<param1 >=<value1 >&<param2 >=<value2 >

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表11-1] 簡易ucode解決のリクエストパラメータ

パラメータ名	型	説明
ucode	xsd:anyURI	対象のucode
paramN	std:string	解決パラメータ

本コマンドが使用する具体的な値を以下の表に列記する。

[表11-2] ucode解決のパラメータ

パラメータ名	デフォルト値	説明
X-UIDC-GWMODE	resolveall	解決モード
X-UIDC-QUERYMASK	all_1	識別子解決のマスク値
X-UIDC-QUERYATTRIBUTE	UIDC_ATTR_ANONYMOUS	取得する解決情報の属性
resolveall:	識別子解決(多段解決)	
resolve:	識別子解決(1階級解決)	
redirect:	識別子解決とHTTPリダイレクト	
UIDC_ATTR_ANONYMOUS:	指定しない	
UIDC_ATTR_RS:	解決サーバ	
UIDC_ATTR_IS:	情報サーバ	
UIDC_ATTR_USER:	ユーザ定義情報	

<レスポンス>

レスポンスは、以下の表の構造データをJSONまたはXML形式で表現したものである。ただし、X-UIDC-GWMODE パラメータにredirect を指定した場合は、解決先URLにリダイレクトする。

[表11-3] 簡易ucode解決のレスポンスパラメータ

パラメータ名	型	説明
results	struct[]	解決情報のリスト。
—X-UIDC-DATA	xsd:string	解決結果データ
—X-UIDC-DATAVERSION	xsd:integer	解決結果データのバージョン
—X-UIDC-DATATYPE	xsd:integer	解決結果のデータタイプ
—X-UIDC-RETURNMASK	xsd:string	解決結果のビットマスク
—X-UIDC-TTL	xsd:integer	解決結果の有効期限
—X-UIDC-RESOLVEMODE	xsd:integer	解決モード

——[11-2: ucode解決(ucodeからオープンデータの参照先の取得)]——

<機能概要>

ucodeに結びつけられたオープンデータの参照先を取得する。

<メソッド>

GET

<URLパス>

/api/v1/resolve/<ucode>?<param1 >=<value1 >&<param2 >=<value2 >

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、GETメソッドのURL部に格納する。

[表11-4] ucode解決(ucodeからオープンデータの参照先の取得)のリクエストパラメータ

パラメータ名	型	説明
ucode paramN	xsd:anyURI std:string	対象のucode 解決パラメータ

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表11-5] ucode解決(ucodeからオープンデータの参照先の取得)のレスポンスパラメータ

パラメータ名	型	説明
results	struct[]	解決情報のリスト。各情報は以下の構造をもつ。
X-UIDC-ATTRIBUTE	xsd:string	解決結果の属性値
X-UIDC-DATA	xsd:integer	解決結果データ
X-UIDC-RETURNMASK	xsd:string	解決結果のビットマスク
X-UIDC-RESOLVEMODE	xsd:integer	解決モード

――[11-3: ucode解決情報の新規作成]――

<機能概要>

ucodeに対して、オープンデータの参照先を結びつける。

<メソッド>

POST

<URLパス>

/api/v1/resolve

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、POSTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表11-6] ucode解決情報の新規作成のリクエストパラメータ

パラメータ名	型	説明
target params	xsd:anyURI struct[]	対象のucode キーが登録パラメータ名、値が登録値であるリスト。

<レスポンス>

レスポンスは、以下の表に示す構造データをJSONまたはXML形式で表現したものである。

[表11-7] ucode解決情報の新規作成のレスポンスパラメータ

パラメータ名	型	説明
ucode	xsd:anyURI []	作成されたucode解決情報を識別するucode。

――[11-4: ucode解決情報の更新]――

<機能概要>

ucodeに対して、オープンデータの参照先を結びつけを更新する。

<メソッド>

PUT

<URLパス>

/api/v1/resolve/<ucode>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表11-8] ucode解決情報の更新のリクエストパラメータ

パラメータ名	型	説明
ucode	xsd:anyURI	ucode解決情報を識別するucode
target params	xsd:anyURI struct[]	対象のucode キーが登録パラメータ名、値が登録値であるリスト。

<レスポンス>

成功時、レスポンスボディは空である。

[11-5: ucode解決情報の削除]

<機能概要>

ucodeに対して、オープンデータと参照先との結びつけを削除する。

<メソッド>

DELETE

<URLパス>

/api/v1/resolve/<ucode>

<リクエストパラメータ>

リクエストパラメータ(以下の表)は、PUTメソッドのボディ部に、JSONまたはXML形式で格納する。

[表11-9] ucode解決情報の削除のリクエストパラメータ

パラメータ名	型	説明
ucode	xsd:anyURI	ucode解決情報を識別するucode

<レスポンス>

成功時、レスポンスボディは空である。

12. Traffic Extension（交通実証拡張）

公共交通情報流通連携基盤システム

Traffic Extension（交通実証拡張）は、公共交通に関する情報を統一した形で保持・運用するための機能である。ここでは、公共交通機関の情報の中で、静的な情報を扱う。具体的な範囲を以下に挙げる。

- ・ 駅
- ・ 鉄道路線
- ・ バス停
- ・ バス路線
- ・ 時刻表

本機能で取り扱うデータは以下の通りである。

- ・ リアルタイムなデータ
 - － 走行位置データ
 - － 遅延データ
 - － 運休データ
 - － ダイヤ変更データ
 - － 臨時ダイヤデータ
 - － その他緊急情報に係るデータ
 - － リアルタイムな公共交通施設情報に係るデータ
- ・ 静的なデータ
 - － 静的な運行情報に係るデータ
 - － 路線データ
 - － 駅・停留所に係るデータ
 - － 公共交通関連施設データ

——[12-1: 公共交通に関する地物情報の検索]——

<機能概要>

公共交通情報として扱うエンティティの検索を行う。ここで扱う情報はすべて地物であるため、情報流通連携基盤におけるGeographical Data Management Command に準拠した検索API を提供する。

<メソッド>

GET

<URLパス>

/api/v1/places?lat=<lat>&lon=<lon>&radius=<radius>&...

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。複数のパラメータを指定した場合、AND検索となる。

[表12-1] 公共交通に関する地物情報の検索のリクエストパラメータ

パラメータ名	値の型	説明
lat	xsd:double	[必須] WGS84 での緯度
lon	xsd:double	[必須] WGS84 での経度
radius	xsd:double	[必須] 検索半径[m]
floor	xsd:double[]	階数、上下限値をカンマ区切り、地上(屋外)は0。
alt	xsd:double[]	高度、上下限値をカンマ区切り。
offset	xsd:integer	lat、lon で指定した地点からの距離が近い順に並べた場合にoffset番目からlimit個分のデータを要求する。
limit	xsd:integer	(前項参照)

<レスポンス>

レスポンスは、以下の構造データを JSONまたはXML形式で表現したものである。

[表12-2] 公共交通に関する地物情報の検索のレスポンスパラメータ

パラメータ名	型	説明
places	RDF	場所の連想配列。レスポンス形式にXMLを指定した場合、各エンティティ情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各エンティティ情報はRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれない場所がある場合はtrue、ない場合はfalse。

レスポンスに含まれるプロパティ(predicate)とその意味は以下の通りである。

プロパティ	値域	意味
rdf:type	rdfs:Class	公共交通データのクラス。現状ではug:Station またはug:BusStop のいずれか。
dc:title	xsd:string	地物の名称(駅名やバス停名)
ug:region	xsd:string	鉄道・軌道(索道は除く)沿いに設置される駅やバス停の幾何情報(GeoJSON形式)
w3cgeo:lat	xsd:double	駅やバス停の代表点の緯度
w3vgeo:long	xsd:double	駅やバス停の代表点の経度

——[12-2: 静的公共交通情報の検索]——

<機能概要>

静的公共交通情報を検索する。

<メソッド>

GET

<URLパス>

/api/v1/datapoints?<param1>=<value1>&<param2>=<value2>...

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。

[表12-3] 静的公共交通情報の検索のリクエストパラメータ

パラメータ名	説明
paramN	検索対象パラメータ名
valueN	検索対象パラメータ値

<レスポンス>

レスポンスは、以下の構造データを JSONまたはXML形式で表現したものである。

[表12-4] 静的公共交通情報の検索のレスポンスパラメータ

パラメータ名	型	説明
results	RDF	静的公共交通情報の連想配列。レスポンス形式にXMLを指定した場合、各情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各情報はRDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれない情報がある場合はtrue、ない場合はfalse。

レスポンスに含まれるプロパティ (predicate) とその意味は以下の表の通りである。

プロパティ	値域	意味
rdf:type	rdfs:Class	静的公共交通情報のクラス。現状では表2に挙げたクラスのいずれか。
dc:title	xsd:string	静的公共交通情報の名称(駅名やバス停名)
ug:region	xsd:string	駅やバス停の幾何情報(GeoJSON形式)
pt:operatorCode	xsd:string	管轄会社の識別コード
pt:nameOfRailway	xsd:string	路線名
pt:operator	xsd:string	管轄会社名
pt:yearOfTogether	xsd:integer	静的公共交通情報の対象に対する供用が開始された年
pt:yearOfBegin	xsd:integer	静的公共交通情報の対象を設置開始した年
pt:yearOfEnd	xsd:integer	静的公共交通情報の対象を設置終了した年

クラス名	alias_URI	ucodeの付与対象 [例]
駅	ug:Station	管轄会社毎に付与する [JR五反田駅、東急五反田駅、都営五反田駅]
鉄道路線	ug:Railway	路線毎に付与する [山手線、浅草線、池上線]
バス停	ug:BusStop	バス停毎に付与する [五反田駅前、戸越銀座]
バス路線	ug:BusRoute	路線毎に付与する [31六本木循環]
時刻表	pg:Diagram	時刻表情報毎に付与する [「駅名, {発or 着}, 時刻」の組]

——[12-3: 静的公共交通情報の閲覧]——

<機能概要>

静的公共交通情報を閲覧する。

<メソッド>

GET

<URLパス>

/api/v1/datapoints/<target>

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。

[表12-5] 静的公共交通情報の閲覧のリクエストパラメータ

パラメータ名	型	説明
target	xsd:anyURI []	静的公共交通情報の識別子

<レスポンス>

レスポンスは、以下の構造データを JSONまたはXML形式で表現したものである。

[表12-6] 静的公共交通情報の閲覧のレスポンスパラメータ

パラメータ名	型	説明
results	RDF	静的公共交通情報の連想配列。レスポンス形式にXMLを指定した場合、各情報はRDF/XMLで表現される。レスポンス形式にJSONを指定した場合、各情報はRDF/JSONで表現される。

レスポンスに含まれるプロパティ (predicate) とその意味は、前の表の通りである。

——[12-4: 公共交通に関するエンティティの絞り込み検索]——

<機能概要>

公共交通情報として扱うエンティティの検索を行う。

<メソッド>

GET

<URLパス>

/api/v1/places?lat=<lat>&lon=<lon>&radius=<radius>&...

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。複数のパラメータを指定した場合、AND検索となる。

[表12-7] 公共交通に関するエンティティの絞り込み検索のリクエストパラメータ

パラメータ名	値の型	説明
lat	xsd:double	[必須] WGS84 での緯度
lon	xsd:double	[必須] WGS84 での経度
radius	xsd:double	[必須] 検索半径[m]
floor	xsd:double[]	階数、上下限値をカンマ区切り、地上(屋外)は0。
alt	xsd:double[]	高度、上下限値をカンマ区切り。
offset	xsd:integer	lat、lon で指定した地点からの距離が近い順に並べた場合にoffset番目からlimit個分のデータを要求する。
limit	xsd:integer	(前項参照)
hops	xsd:integer	URI をたどるホップカウント数。
rdf_type	xsd:anyURI	対象となるエンティティのタイプ(rdf:type)

rdf_type パラメータ値として指定できるクラスを以下に示す。

クラス	説明
puti:Train	電車
ug:Station	駅
ug:RailWay	線路
puti:Bus	バス
ug:BusStop	バス停
ug:BusRoute	バス路線
puti:Diagram	時刻表
puti:Sensor	センサ

<レスポンス>

レスポンスは、以下の構造データを JSONまたはXML形式で表現したものである。

[表12-8] 公共交通に関するエンティティの絞り込み検索のレスポンスパラメータ

パラメータ名	型	説明
results	RDF	エンティティ情報の連想配列。RDF/JSONで表現される。
remains	xsd:boolean	条件にマッチするが、レスポンスに含まれない場所がある場合はtrue、ない場合はfalse。

——[12-5: 時刻表の検索]——

<機能概要>

時刻表のデータを取得する。

<メソッド>

GET

<URLパス>

/api/v1/puti/diagram/<ucode>

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。

[表12-9] 時刻表の検索のリクエストパラメータ

パラメータ名	値の型	説明
ucode	xsd:anyURI	時刻表の識別子

<レスポンス>

レスポンスは、以下の構造データからなる配列を JSONまたはXML形式で表現したものである。

[表12-10] 時刻表の検索のレスポンスパラメータ

パラメータ名	型	説明
time	Time	時刻
destination	String	行き先(オプション)
isArrival	Bool	着時刻である場合はtrue、省略時はfalse
isNonStepBus	Bool	ノンステップバスである場合はtrue、省略時はfalse

――[12-6: 走行情報・遅延情報の取得]――

<機能概要>

列車・バスの走行情報・遅延情報を取得する。

<メソッド>

GET

<URLパス>

/api/v1/datapoints?rdf_type=<rdf_type>&...

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。

[表12-11] 走行情報・遅延情報の取得のリクエストパラメータ

パラメータ名	値の型	説明
lat	xsd:double	[必須] WGS84 での緯度
lon	xsd:double	[必須] WGS84 での経度
radius	xsd:double	[必須] 検索半径[m]
floor	xsd:double[]	階数、上下限値をカンマ区切り、地上(屋外)は0。
alt	xsd:double[]	高度、上下限値をカンマ区切り。
offset	xsd:integer	lat、lon で指定した地点からの距離が近い順に並べた場合にoffset番目からlimit個分のデータを要求する。
limit	xsd:integer	(前項参照)
rdf_type	xsd:anyURI	[必須] ug_Train またはug_Bus に固定
stream	xsd:integer	0 を指定した場合、ストリーム配信を開始

<レスポンス>

rdf_type にputi:Train を指定した場合のレスポンスは、以下のプロパティをもつRDF/JSONデータである。

[表12-12] 走行情報・遅延情報の取得のレスポンスパラメータ

プロパティ名	値域	意味
rdf:type	—	puti:Train に固定
dc:title	xsd:string	列車番号
puti:nameOfRailway	xsd:string	鉄道路線の名称
ug:railway	ug:Railway	鉄道路線のucode
puti:nameOfDestination	xsd:string	行き先名
puti:destination	ug:Station	行き先のucode
puti:diagrams	xsd:string	時刻表データが格納されたURL
w3cgeo:lat	xsd:double	代表点の緯度。10進数表記
w3cgeo:long	xsd:double	代表点の経度。10進数表記

rdf_type にputi:Bus を指定した場合のレスポンスは、以下のプロパティをもつRDF/JSONデータである。

プロパティ名	値域	意味
rdf:type	—	puti:Bus に固定
dc:title	xsd:string	列車番号
puti:nameOfBusRoutes	xsd:string	バス路線の名称
ug:busRoute	ug:BusRoute	バス路線ucode
puti:nameOfDestination	xsd:string	行き先名
puti:destination	ug:BusStop	行き先のucode
puti:diagrams	xsd:string	時刻表データが格納されたURL
w3cgeo:lat	xsd:double	代表点の緯度。10進数表記
w3cgeo:long	xsd:double	代表点の経度。10進数表記

Entity	属するクラス	ucode の付与対象 [例]
列車	puti:Train	列車番号毎に付与 [3510M, 2453M]
バス車両	puti:Bus	運行番号毎に付与 [M234, M4343]

ある列車編成の遅延情報はその列車編成に振られたucode のrelation として表現される。本コマンドは、そのrelation のリストを取り出す。

——[12-7: 在線情報の取得]——

<機能概要>

列車・バスの在線情報を取得する。

<メソッド>

GET

<URLパス>

/api/v1/datapoints?rdf_type=<rdf_type>&...

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。

[表12-13] 在線情報の取得のリクエストパラメータ

パラメータ名	値の型	説明
rdf_type	xsd:anyURI	[必須] 取得対象の駅・停留所・路線のクラス。 ug_Station、ug_Railway、ug_Busstop、ug_Busroute に固定。
dc_title stream	xsd:string xsd:integer	[必須] 情報を取得したい駅・停留所・路線の名称。 0 を指定した場合、ストリーム配信を開始

<レスポンス>

rdf_type にputi:Train を指定した場合のレスポンスは、以下のプロパティをもつRDF/JSONデータである。

[表12-14] 在線情報の取得のレスポンスパラメータ

プロパティ名	値域	意味
dc:time puti:Trains puti:Buses	xsd:datetime puti:Train[] puti:Bus[]	その状態である時刻 列車のリスト バスのリスト
Entity	属するクラス	ucode の付与対象 [例]
駅	puti:Train	駅と管轄会社毎に付与 [JR五反田駅、東急五反田駅、 都営五反田駅]
路線	puti:Bus	路線毎に付与 [山手線、都営浅草線、東急池上線]
バス停	ug:BusStop	バス停毎に付与 [五反田駅前、戸越銀座]
バス路線	—	— [31 六本木循環]

例えば、五反田駅の情報を取得すると、五反田駅に停車していると観測された列車一覧が取得される。また山手線の路線を指定すると、その区間を走行中の列車一覧が取得される。

——[12-8: 運行情報の取得]——

<機能概要>

列車・バスの運行情報を取得する。

<メソッド>

GET

<URLパス>

/api/v1/datapoints?rdf_type=puti_TrainInfo&puti_TrainInfoLine=<RAILWAY_NAME>

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。

[表12-15] 運行情報の取得のリクエストパラメータ

パラメータ名	値の型	説明
rdf_type	xsd:anyURI	[必須] 運行情報のクラス。 puti_TrainInfo に固定。
puti_TrainInfoLine	xsd:string	[必須] 対象路線名。
stream	xsd:integer	0を指定した場合、ストリーム配信を開始

<レスポンス>

以下のプロパティをもつ RDF/JSONデータである。

[表12-16] 運行情報の取得のレスポンスパラメータ

プロパティ名	値域	意味 [例]
rdf_type	—	puti:TrainInfo に固定
dc:time	xsd:datetime	データ生成時刻 [2013-01-13T15:10:00+9]
puti:ofRailway	xsd:anyURI	発生路線ucode [urn:ucode:_00001C0000 0000000001000000012345]
puti:trainInfoTime	xsd:string	発生時刻 [15時03分頃]
puti:trainInfoLine	xsd:string	発生路線名 [中央線快速電車]
puti:trainInfoFrom	xsd:string	発生場所起点 [中野]
puti:trainInfoTo	xsd:string	発生場所終点 [高円寺]
puti:trainInfoRange	xsd:string	発生区間 [中野-高円寺]
puti:trainInfoDirection	xsd:string	行き先
puti:trainInfoCause	xsd:string	発生理由 [信号トラブル]
puti:trainInfoStatus	xsd:string	現状 [運転見合わせ]
puti:trainInfoText	xsd:string	運行情報テキスト [中央線快速電車は15 時03分頃中野～高円寺駅間での信号トラ ブルの影響で上下線で運転を見合わせて います]

——[12-9: リアルタイムな公共交通施設情報の取得]——

<機能概要>

リアルタイムに変化する公共交通施設情報を取得する。

<メソッド>

GET

<URLパス>

/api/v1/places?rdf_type=puti_Sensor&lat=<lat>&lon=<lon>&radius=<radius>&...

<リクエストパラメータ>

リクエストパラメータは、GETメソッドのURL部に格納する。

[表12-17] リアルタイムな公共交通施設情報の取得のリクエストパラメータ

パラメータ名	値の型	説明
lat	xsd:double	[必須] WGS84での緯度
lon	xsd:double	[必須] WGS84での経度
radius	xsd:double	[必須] 検索半径[m]
floor	xsd:double[]	階数、上下限値をカンマ区切り、地上(屋外)は0。
alt	xsd:double[]	高度、上下限値をカンマ区切り。
offset	xsd:integer	lat、lon で指定した地点からの距離が近い順に並べた場合にoffset番目からlimit個分のデータを要求する。
limit	xsd:integer	(前項参照)
rdf_type	xsd:anyURI	[必須] puti_Sensor 固定

<レスポンス>

以下のプロパティをもつ RDF/JSONデータである。

[表12-18] リアルタイムな公共交通施設情報の取得のレスポンスパラメータ

プロパティ名	値域	意味
rdf_type	xsd:anyURI	puti:Sensor 固定
dc:time	xsd:datetime	センサからのデータ取得時刻
uc:temperature	xsd:float	気温(セルシウス度)
uc:humidity	xsd:float	湿度(パーセント)
puti:pollen	xsd:integer	花粉飛散量(個/m3)
w3cgeo:lat	xsd:double	代表点の緯度。10進数表記
w3cgeo:long	xsd:double	代表点の経度。10進数表記