CREST API Descriptor

CREST is ForgeRock's standard library for exposing RESTful web services.

This document specifies an API descriptor that describes CREST APIs in a standard format that developers can read and tools can consume. API descriptors enable clients such as static documentation generators, live documentation websites, and other tooling.

When consuming a CREST API descriptor, the fundamental assumption is that the consumer knows CREST. Therefore, an API descriptor specifies which CRUDPAQ operations a resource supports, whether a resource supports MVCC, whether a create operation calls for a server-assigned or client-assigned ID, and what patch operations can be used with a resource that supports patch. An API descriptor does not describe CREST itself, however, nor does it describe how CREST binds to a transport protocol such as HTTP.

Status

This document is version `1.0.0`.

Revision History

Version	Description
1.0.0	Initial release.
0.2.1	Clarify explanations.
0.2	Adds: ability to specify common resources in `services`, collection semantics using `items`, parent-child relations using `subresources`, complex read states, complex field validation policies, and binary field support.
0.1	Initial version.

Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Types

In this document the following type names are used:

- frURI - A ForgeRock API Descriptor URI. These start with the `frapi:` scheme, and are followed by colon-separated heirarchical component IDs to make a unique identifier. For example: `frapi:openam:identities`, which could be used in a JSON Reference: `frapi:openam:/:identities#/definitions/user`

Specification

ApiDescription (Top-Level Object)

At the top level, the API is described as a collection of schema definitions, service definitions, errors, and paths (endpoints) available in the application.

The API documented by an ApiDescription is simply a set of CREST web services exposed to consumers. An ApiDescription MAY cover a single endpoint or an entire product. An ApiDescription does not necessarily represent a single, coherent library of services for a specific purpose.

The top-level MUST contain at least one of *definitions*, *errors*, *paths*, or *services*.

Properties

Key	Type	Required?	Description
`id`	String		The frURI identifier of the API Descriptor
`version`	String		The version of the API
`description`	String		Human-readable description of the API for documentation purposes
`definitions`	Definitions		Locally defined, extended JSON schema for resources
`services`	Services		Locally defined CREST services that can be exposed at a path
`errors`	Errors		Locally defined errors
`paths`	Paths		Paths (endpoints) exposed by the API

The top-level `version` and `description` properties apply to the *entire API*.

Individual request handlers MAY be versioned separately, as described in VersionedPath.

Definitions

Map of locally defined extended JSON schema, that can be referred to via JSON References.

Properties

Key	Type	Required?	Description
*	Schema		The schema definitions

Services

Map of locally defined service definitions, that can be referred to via JSON References.

A service definition describes resources and operations they support, independently of the path (endpoint) where the resource is exposed. Service definitions are useful when the path depends on configuration, and when multiple paths can expose the same web service.

Properties

Key	Type	Required?	Description
*	Resource		The service definitions

Errors

Map of locally defined errors, that can be referred to via JSON References.

Properties

Key	Type	Required?	Description
*	ApiError		The error definitions

Reference

JSON Reference syntax referring to schemas, service definitions, and errors that are defined locally or externally.

Properties

Key	Type	Required?	Description
`$ref`	String		A JSON Reference (`$ref`) to the required object. The URI should be an frURI type, or a URL.

Paths

Map of paths (endpoints) supported by the API being described.

Paths MAY include path parameters contained in curly braces, for example, `/users/{userId}/devices/{deviceId}`.

Paths MUST contain at least one *VersionedPath*.

Properties

Key	Type	Required?	Description
*	VersionedPath		A mapping of path strings to VersionedPath definitions.

VersionedPath

Optional version of the request handler at the top of a particular path.

When all request handlers share the same version defined at the top-level, API descriptors MAY omit this level of the Path hierarchy.

The resource MAY be specified using a Reference.

Properties

Key	Type	Required?	Description
*	Resource		The supported versions of the resources at this path. Format: `[1-9][0-9]*(\.[1-9][0-9]*)*`

Only *N* and *N.N* formats are supported for a version number key.

The reserved value `0.0` means "unversioned". If `0.0` is used it MUST be the only VersionedPath entry for a path.

Resource

The resource accessible at a given path and also the CREST operations it supports.

Collection resources have `items` that are all of the same type. For example, each item in a collection of users is a user. In a collection that supports create, the `items` are the resources to create. In a collection that supports queries, the `items` are the items in the `result` array of the response resource.

Resources can have either `items` or `subresources` that support their own CREST operations, and that are versioned with the resource. For example, a subscriber has subscriptions. A resource cannot have both `items` *and* `subresources` - if a resource has an `items` then the `subresources` are declared on that node.

A resource MUST define at least one CRUDPAQ Operation.

Properties

Key	Type	Required?	Description
`title`	String		Human-readable string used as a title in documentation.
`description`	String		Human-readable description for documentation purposes.
`resourceSchema`	Schema		The schema of the resource for this path. Required when any of create, read, update, delete, or patch are supported.
`create`	Create		Specifies the create operation that the resource supports.
`read`	Read		Specifies the read operation that the resource supports.
`update`	Update		Specifies the update operation that the resource supports.
`delete`	Delete		Specifies the delete operation that the resource supports.
`patch`	Patch		Specifies the patch operation that the resource supports.
`actions`	Action[]		Specifies the action operations that the resource supports.
`queries`	Query[]		Specifies the query operations that the resource supports. Resource queries arrays can include up to one query filter operation, one query expression operation, and multiple queries by ID.
`subresources`	SubResources		Sub-resources of this resource. Sub-resources use the same version as their parent resource, so are not separately versioned. This field should not be used when the `items` field is used - sub-resources should be added to `items/subresources` instead.
`items`	Items		Descriptor for the items in a collection. Defined only when the resource is a collection.
`mvccSupported`	boolean		Whether this resource supports MVCC operations.
`parameters`	Parameter[]		Extra parameters supported by the resource.

Items

The resource type and operations support on the items of a collection.

An item MUST define at least one CRUDPA Operation.

Properties

Key	Type	Required?	Description
`create`	Create		Specifies the create operation that the resource supports.
`read`	Read		Specifies the read operation that the resource supports.
`update`	Update		Specifies the update operation that the resource supports.
`delete`	Delete		Specifies the delete operation that the resource supports.
`patch`	Patch		Specifies the patch operation that the resource supports.
`actions`	Action[]		Specifies the action operations that the resource supports.
`pathParameter`	Parameter		The path parameter for the item instances.
`subresources`	SubResources		Sub-resources of this collection resource. Sub-resources use the same version as their parent resource, so are not separately versioned.

SubResources

Sub-resources are resources that are a component part of their parent. As such, they share the version of the parent from the parent's path binding. If a sub-path is separately versioned from a parent path, it MUST be listed as a separate path in the Paths object, rather than as a sub-resource of the resource at the parent path.

Properties

Key	Type	Required?	Description
*	Resource		A mapping of sub-resource paths to resource definitions.

Operation

A basic operation type for operations without any special features.

This is the supertype for all CRUDPAQ operations: All operations inherit these properties.

Properties

Key	Type	Required?	Description
`description`	String		Human-readable description for documentation purposes.
`supportedLocales`	String[]		Locale codes supported by the operation.
`errors`	ApiError[]		Errors known to be returned by this operation.
`parameters`	Parameter[]		Extra parameters supported by this operation.
`stability`	String		Interface stability for this operation. Supported values are: `internal`, `stable` (default), `evolving`, `deprecated`, and `removed`.

ApiError

Defines one of the possible error responses that are known to be returned by a given Operation. All standard CREST errors are defined under `frapi:common`, which is an API Descriptor that will always be available by default. Endpoints MAY overload any error *code* and unique *description* to define custom errors. CREST API clients should be prepared to handle undocumented/unexpected errors. It is a best practice to define a minimum ApiError array definition, with 500 Internal Server Error, as follows,

```
"errors" : [
    {"$ref" : "frapi:common#/errors/internalServerError"}
]
```

Properties

Key	Type	Required?	Description
`code`	Integer		Three-digit error code, corresponding to an HTTP status code.
`description`	String		Description of what may cause an error to occur.
`schema`	Schema		Schema for the error detail.

Parameter

Defines either an additional parameter for an operation that is not part of the request payload, or a path parameter expressed as a value surrounded by curly braces in a path.

Properties

Key	Type	Required?	Description
`name`	String		The name of the parameter.
`type`	String		The semantics/format of the parameter: `string`, `number`, `boolean`, and array variants.
`source`	String		Where the parameter comes from. Supported values are: `ADDITIONAL` or `PATH`.
`defaultValue`	String		The default value, if applicable.
`description`	String		The description of the parameter.
`required`	boolean		Whether the parameter is required. (default: `false`)
`enumValues`	String[]		One or more values that must match.
`enumTitles`	String[]		`options/enum_titles` - string descriptions in the same order as the enum values.

Other appropriate fields as described in the JSON Schema Validation spec may also be used.

Create

Indicates that creating a new resource is supported.

Extends Operation.

Properties

Key	Type	Required?	Description
`mode`	String		Supported values are: `ID_FROM_CLIENT`, `ID_FROM_SERVER`.
`singleton`	boolean		Specifies that create operates on a singleton as opposed to a collection.

Read

Indicates that reading the contents of an existing resource is supported.

Extends Operation.

Properties

No additional properties.

Update

Indicates that replacing the contents of an existing resource is supported.

Extends Operation.

Properties

No additional properties.

Delete

Indicates that deleting a resource is supported.

Extends Operation.

Properties

No additional properties.

Patch

Indicates that partially modifying the contents of an existing resource is supported.

Extends Operation.

Note that CREST has its own definition for the patch request payload. The content of the request payload is a JSON array whose elements are the modification operations to apply to the resource.

The following example adds the value `admin` to the `roles` field, and removes `valueToBeRemoved` from `field/subfield`:

```
[
  {
    "operation" : "add",
    "field" : "roles",
    "value" : "admin"
  },
  {
    "operation" : "remove",
    "field" : "field/subfield",
    "value" : "valueToBeRemoved"
  }
]
```

The request payload items' properties depend on the patch operation. The request payload values depend on the resource schema. Given these dependencies, a patch request payload has the following schema:

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "operation": {
        "type": "string",
        "enum": [
          "add",
          "remove",
          "replace",
          "increment",
          "move",
          "copy",
          "transform"
        ],
        "required": true
      },
      "field": {
        "type": "string"
      },
      "from": {
        "type": "string"
      },
      "value": {
        "type": "string"
      }
    }
  }
}
```

Properties

Key	Type	Required?	Description
`operations`	String[]		Set of supported patch operations. Supported values are: • `ADD` • `REMOVE` • `REPLACE` • `INCREMENT` • `MOVE` • `COPY` • `TRANSFORM`

Action

Indicates that one or more additional action operations on the resource are supported.

Extends Operation.

Properties

Key	Type	Required?	Description
`name`	String		The action name.
`response`	Schema		The schema of the response payload for this action.
`request`	Schema		The schema of the request payload for this action.

Query

Indicates that searching or listing the resources in this resource container is supported.

Extends Operation.

Resource queries arrays can include up to one query filter operation, one query expression operation, and multiple queries by ID.

Properties

Key	Type	Required?	Description
`type`	String		Supported values are:`ID`, `FILTER`, `EXPRESSION`.
`pagingModes`	String[]		Supported values are:`COOKIE`, `OFFSET`. Paging is not supported if omitted.
`countPolicies`	String[]		Supported values are:`ESTIMATE`, `EXACT`, `NONE`. Counts are not provided if omitted.
`queryId`	String	`type:ID`	Required if `type` is `ID`.

`queryableFields`	String[]	`type:FILTER`	Required if `type` is `FILTER`. Lists the fields in the `resourceSchema` that can be queried. * means all fields can be queried.
`supportedSortKeys`	String[]		The keys that may be used to sort the filter results. * means all keys are supported.

The following example shows a resource that supports all three types of query:

```
"paths": {
  "/openidm/managed/user": {
    "queries": [{
      "type": "EXPRESSION",
      "description": "Return the results of an SQL query such as `_queryExpression=select+%2A+from+managed_user`"
    }, {
      "type": "FILTER",
      "description": "Return resources matching the filter.",
      "queryableFields": ["*"]
    }, {
      "type": "ID",
      "queryId": "query-all-ids"
      "description": "Return all resources.",
    }]
  }
}
```

Schema

API descriptors use schemas to represent request payloads, response resources, and error responses.

API descriptors support using either a Reference to a defined schema, or OpenAPI-extended JSON schema with additional extensions for the following use cases:

Field Order

To specify the order of fields displayed in a UI.

Properties

Key	Type	Description
`propertyOrder`	number	The number that is specified can be used to compare the order of different properties. See also https://github.com/jdorn/json-editor/issues/110

Complex Read/Write States

To clarify read/write states more complex than `readOnly`.

Properties

Key	Type	Description
`readPolicy`	String	Supported values are: • `USER`: visible in the user-interface. (default) • `CLIENT`: hidden from user-interface, but readable via client APIs. • `SERVER`: available internally, but not exposed to client APIs.
`writePolicy`	String	Relevant only for "properties" definitions where `readOnly` is false. Supported values are: • `WRITE_ON_CREATE`: the property MAY be set in the create request, but not thereafter. • `WRITE_ONCE`: the property MAY be set only if the current value is NULL. • `WRITABLE`: the property can be set at any time. (default)
`errorOnWritePolicyFailure`	boolean	Whether the application will return an error (or ignore) when a `WRITE_ON_CREATE` or `WRITE_ONCE` field is attempted to be updated erroneously (default: `false`).
`returnOnDemand`	boolean	`true` when a field is available, but must be explicitly requested. `false` (default) when always returned.

Enumeration Value Descriptions

To assign titles/descriptions to enumeration values.

Enum titles are not supported officially for the JSON enum type.

This representation is already used in existing projects, and comes from JSON Editor.

Properties

Key Path	Type	Description
`options/enum_titles`	String[]	String descriptions in the same order as the enum values.

Example

```
{
  "type": "string",
  "enum": ["value1","value2"],
  "options": {
    "enum_titles": ["title1","title2"]
  }
}
```

Example values

Similar to the OpenAPI specification, we support example values for JSON Schemas, but unlike OpenAPI, we also support these example values on sub properties too.

Properties

Key	Type	Description
`schema`	*Match containing schema*	An example value. The type should match whatever the containing schema or property type is.

References

1. OpenAPI Specification, v2.0
2. JSON Reference, draft-03
3. JSON Schema: core definitions and terminology, draft-00
4. JSON Schema: interactive and non interactive validation, draft-00
