SPARROW

A new kind of Directory

kayyagari@keydap.com

What is project Sparrow?

- A Directory built on top of SCIM protocol
- An Identity Server supporting OpenIDConnect and OAuth2.0 and a read-only LDAP interface

Goals

- Simplified schema and access control
- Unified service for storage, authentication and SSO
- Better audit log analysis and event management
- Support both HTTP and TCP out of the box (no additional frontends)

SCIM

System for Cross-domain Identity Management

Two IETF standards:

1. Schema rfc7643
2. Protocol rfc7644

SCIM

Schema

- Resources
- Resource schemas and Resource types
- Simple and Complex attributes

SCIM-Schema

Resource

```
{
    "schemas":[
        "urn:ietf:params:scim:schemas:core:2.0:User"
    ],
    "userName":"elecharny",
    "externalId":"elecharny",
    "name":{
        “formatted":"Emmanuel Lecharny",
        "familyName":"Lecharny",
        "givenName":"Emmanuel"
    }
    …
}
```

SCIM-Schema

Resource Schema

```
{
    "id":"urn:ietf:params:scim:schemas:core:2.0:User",
    "name":"User",
    "description":"User Account",
    "attributes":[
        {
            "name":"userName",
            "type":"string",
            "multiValued":false,
            "description":"Unique identifier for the User",
            "required":true,
            "caseExact":false,
            "mutability":"readWrite",
            "returned":"default",
            "uniqueness":"server"
        },
        {
            "name":"name",
            "type":"complex",
            "multiValued":false,
            "description":"The components of the user's real name",
            "required":false,
            "subAttributes":[
                {
                    "name":"formatted",
                    "type":"string",
                    "multiValued":false,
                    "description":"The full name",
                    "required":false,
                    "caseExact":false,
                    "mutability":"readWrite",
                    "returned":"default",
                    "uniqueness":"none"
                },
```

SCIM-Schema

Resource Type Schema

```
{
    "schemas":[
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
    ],
    "id":"User",
    "name":"User",
    "endpoint":"/Users",
    "description":"User Account",
    "schema":"urn:ietf:params:scim:schemas:core:2.0:User",
    "schemaExtensions":[
     {
  "schema":"urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
  "required":false
     }
    ],
    "meta":{
        "location":"v2/ResourceTypes/User",
        "resourceType":"ResourceType"
    }
}
```

SCIM-Schema

Resource with Extension

```
{
    "schemas":[
        "urn:ietf:params:scim:schemas:core:2.0:User"
    ],
    "userName":"elecharny",
    "externalId":"elecharny",
    "name":{
        “formatted":"Emmanuel Lecharny",
        "familyName":"Lecharny",
        "givenName":"Emmanuel"
    },
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User":{
        "employeeNumber":"701982",
        "costCenter":"4030",
        "organization":"Apache",
        "division":"Software",
        "manager":{
            "value":"26118915-6090-4610-87e4-49d8ca9f808d",
            "$ref":"../Users/20118915-6090-4610-87e4-49d8ca9f808e",
            “displayName":"John Legend"
        }
    }
}
```

SCIM

Protocol

- Based on REST
- Search Filters

SCIM-Protocol

REST

- All CRUD operations are performed over HTTP(S)
- URI follows the format */<resource-type-endpoint>*

e.g. POST https://localhost:7090/Users

```
{
    "schemas":[
        "urn:ietf:params:scim:schemas:core:2.0:User"
    ],
    "userName":"elecharny",
    "externalId":"elecharny",
    "name":{
        "formatted":"Emmanuel Lecharny",
        "familyName":"Lecharny",
        "givenName":"Emmanuel"
    }
}
```

SCIM-Protocol

Search Filters

- Supports 10 operators excluding AND, OR and NOT
- Operators - EQ, NE, CO, SW, EW, GT, LT, GE, LE, PR
- Search can be performed using GET or POST

Examples:

```
GET https://localhost:7090/Users?filter=

1. userName eq “elecharny”

2. userType eq “Contractor” and emails[type eq “work" and
value co “example.com"]
```

Access Control

- SCIM doesn’t have a model for access control
- Sparrow adds custom attributes to Group resource
- Supports “who can do what on a resource”

```
{
    "displayName": "Administrator",
    "id": "00000000-1000-0000-0000-000000000000",
    "permissions": [
        {
            "value": "READ"
        },
        {
            "value": "CREATE"
        },
        {
            "value": "UPDATE"
        },
        {
            "value": "DELETE"
        }
    ],

    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:Group"
    ]
}
```

Sparrow Java Client

- Simplifies working with SCIM resources
- Creates Java types from SCIM schemas (JSON)
- Extendable authenticator

Sparrow Client API

Examples

Generated User resource class

```
@Resource(schemaId="urn:ietf:params:scim:schemas:core:2.0:User", endpoint="/
Users", desc="User Account")
public  class User
{
	private String id;
	private String externalId;
	private Meta meta;
	private String userName;
	private Name name;
	private boolean active;
	private String password;
	private List<Email> emails;
	private List<Group> groups;
	private List<X509Certificate> x509Certificates;

	@Extension("urn:ietf:params:scim:schemas:extension:enterprise:2.0:User")
	private EnterpriseUser enterpriseUser;
…
```

Sparrow Client API

Examples

Creating a User resource

```java
String apiUrl = "http://localhost:7090/v2";
String oauthUrl = "http://localhost:7090/oauth2";

SparrowAuthenticator authenticator = new SparrowAuthenticator("admin", "example.com", "secret");

SparrowClient client = new SparrowClient(apiUrl, oauthUrl, authenticator);
client.register(User.class, Group.class, Device.class);

User u = new User();
u.setUserName("elecharny");
Name name = new Name();
name.setFamilyName("Lecharny");
name.setFormatted("Emmanuel Lecharny");
name.setGivenName("Emmanuel");
u.setName(name);

Email e = new Email();
e.setValue("elecharny@example.com");
e.setPrimary(true);
e.setType("home");

Response<User> resp = client.addResource(u);
```

Sparrow Client API

Examples

Searching for Users

```
SearchRequest req = new SearchRequest();
req.setFilter("emails.type eq \"work\"");
req.setAttributes("username");

SearchResponse<User> resp = client.searchResource(req, User.class);
```

LDAP Interface

- LDAP interface is read-only
- Allows only bind, unbind and search operations
- Supports StartTLS and Password Modify (rfc3062) extended operations
- Creates entries using configurable templates

LDAP Templates

Mapping User resource to User entry

```
{
    "type": "User",
    "objectClasses": [
        "top",
        "person",
        "organizationalPerson",
        "inetOrgPerson",
        "extensibleObject"
    ],

    "dnPrefix": "uid={userName},ou=Users",
    "attributes": [
        {
            "scimAttrPath": "id",
            "ldapAttrName": "entryUUID"
        },
        {
            "scimAttrPath": "userName",
            "ldapAttrName": "uid"
        },
        {
            "scimAttrPath": "name.familyName",
            "ldapAttrName": "sn"
        },
        {
            "scimAttrPath": "addresses",
            "ldapAttrName": "postalAddress",
            "format": "{streetAddress}${locality}${region}${country}${postalCode}"
        },
        {
            "scimAttrPath": "groups",
            "ldapAttrName": "member",
            "format": "cn={display},ou=Groups,{dn}"
        }
    ]
```

Sourcecode Repositories

- Sparrow server (Go) - https://bitbucket.org/keydap/sparrow
- Sparrow Client (Java) - https://github.com/keydap/sparrow-client

Questions?

Thank You