Optional Private Name Spaces for Domains in Keystone

This proposal extends the keystone v3 API to allow the option for administrators to create domains for which they have a private name space in terms of user and project names. It is intended for consideration as core for Grizzly.

Update: 13-Feb-2013. After review by other contributors it was agreed that we would make name spaces mandatory for domains - i.e. a domain, by definition, has a separate name spaces for its user, group and project names. The impact is that the functionality described below is maintained, but without the ability to chose whether the name spaces is private or not. I have left those details in the description below, so that people can see the original proposal. The formal identity api updates (https://review.openstack.org/#/c/21323/), reflect this new agreement, however.

Problem Statement

Cloud providers who are hosting individuals or small enterprises have been able to use the current v2 Keystone API successfully. For v3, with domains, the distinction between a container that represents (and has the same lifetime) as a hosted customer (a domain) and the transient projects in use by a customer is clarified. This will help cloud providers better host and administer larger enterprises, each with many hundreds of users and projects.

However, larger enterprises will expect to able to utilize any human readable identifier (such as user and project name) they chose (within the constraints of length of name supported by openstack). In the current v3 API, however, this is not the case - user and project names are required to be globally unique. A customer might not be able call a project “Test”, simply because another customer of the cloud provider got there first. More crucially, they probably have a set of user names already assigned to their users - and don’t want to have to create new ones just for access to their domain in a openstack hosted public cloud. They might even expect to be able to use their own corporate LDAP, for example, as authentication *just for their domain* (although this is not part of this specific proposal).

Given that a Domain name and ID are both, individually, globally unique, then enterprises will expect that there can be given free reign within their domain to create whatever user and project names they please.

Implementation

The following summarizes the proposed changes:

1) The use of domains remains optional (i.e. a cloud provider can just have everything created in one, common domain). For such installations, uniqueness within (the only) domain and global uniqueness means the same thing and this proposal has no impact. Such an installation can be accessed via the v2 API with current semantics.

2) For multi-domain installations, when a domain is created it can now be optionally specified as having a private name space (for project names, user names or both). If a cloud provider chooses to not allow private names spaces, then again this proposal has no affect - all user and project names remain globally unique

3) If, however, the cloud provider allows a domain to be created with a private name space, then the impact is as follows:

For domains with private projects: User authentication remains unchanged, but project names within such domains only need to be unique within that domain. Any accesses via project_id are unaffected, but if any external component intends to access a project by name, then a domain name or id is also required.

For domains with private users: Authentication to such a domain by user_name/password would require the additional specification of a domain name or ID (both of which are globally unique). Once a token is obtained, this token effectively defines the scope to a domain - so all subsequent calls would not require specification of a domain.

4) Even in the case when there are private name space domains, other (non-private name space) domains can operate as normal. Their user and project names remain part of a common name space across such domains - and authentication (without specifying a domain) is still supported. [This can work since if no domain is specified, keystone would know to look for a user name defined in the common domain space]

5) All **IDs** for resources remain globally unique, so any APIs that specify resources by ID are unaffected

6) Except for the case of 1), projects that use keystone cannot assume that user name and project name alone are globally unique if they are using this to tag resources of some type. The strings of “Domain Name”+”User Name” and “Domain Name”+”Project Name” <u>are</u> globally unique and can be safely used. As stated in 3) above user_ID and project_ID also are independently globally unique. This would involve, for instance, changing of the interface of swift to keystone. [Technically this would also not be required in the case of 2) above, but I think we would want to state that if you support using the v3 API, then you should NOT assume uniqueness of user or project name].

7) Many cloud providers may build their own UI for their customers that handles how a domain

is specified. A common method of doing this will be to provide each customer a unique URL to the “log in page” (with the domain perhaps encoded within the URL). It is proposed the Horizon is updated to support such an ability as an example of such an implementation

The formal proposal for additions/changes the v3 Identity API can be found at: https://review.openstack.org/#/c/18805/. A summary is provided below for reference.

Resources

Domains: `/v3/domains`

Domains represent collections of both projects and users. Each project or user is owned by exactly one domain. Users, however, can be associated with multiple projects by granting roles to the user on a project (including projects owned by other domains).
Additional required attributes:

- `name` (string)
    - Globally unique name.

Optional attributes:

- `description` (string)
- `enabled` (boolean)
    - Setting this value to `false` also disables all projects and users owned by the domain, and therefore implies the same effects of disabling all of those entities individually.
- `priavte_projects` (boolean)
    - Setting this to true means that project names only needs to be unique within this domain. The fefault is false.
- `priavte_users` (boolean)
    - Setting this to true means that user names only needs to be unique within this domain. The fefault is false.

Example entity:

```
{
    "domain": {
        "enabled": true,
        "id": "1789d1",
        "links": {
            "self": "http://identity:35357/v3/domains/1789d1"
        },
        "name": "example.com",
        "private_projects": true,
        "private_users": false
    }
}
```

Core API

Authenticate: `POST /tokens`

For the use case where we are providing a username and password, optionally with a project_name or project_id. If a project_name or project_id is NOT provided, the system will use the default project associated with the user, or return a 401 Not Authorized if a default project is not found or unable to be used.

Request:

```
{
    "auth": {
        "password_credentials": {
            "username": "--user-name--",
            "password": "--password--",
            "user_id": "--optional-user-id--",
            "domain_name": "--optional-domain-name--",
            "domain_id": "--optional-domain-id--"
        },
    "domain_name": "--optional-domain-name--",
    "domain_id": "--optional-domain-id--"
    "project_name": "--optional-project-name--",
    "project_id": "--optional-project-id--"
    }
}
```

Domains

Create domain: `POST /domains`

Request:

```
{
    "description": "",
    "enabled": "",
    "name": "",
    "private_projects": "",
    "private_users": ""
}
```

Response:

```
Status: 201 Created
Location: https://identity:35357/v3/domains/--domain-id--

{
    "description": "desc of domain",
    "enabled": true,
    "id": "--domain-id--",
    "link": {
        "href": "http://identity:35357/v3/domains/--domain-id--",
```

```
        "rel": "self"
    },
    "name": "my domain",
    "private_projects": true,
    "private_users": false
}
```

List domains: `GET /domains`

query_string: page (optional) query_string: per_page (optional, default 30) query filter for "name" and "enabled" (optional)

Response:

```
Status: 200 OK

[
    {
        "description": "desc of domain",
        "enabled": true,
        "id": "--domain-id--",
        "link": {
            "href": "http://identity:35357/v3/domains/--domain-id--",
            "rel": "self"
        },
        "name": "my domain",
        "private_projects": true,
        "private_users": false
    },
    {
        "description": "desc of another domain",
        "enabled": true,
        "id": "--domain-id--",
        "link": {
            "href": "http://identity:35357/v3/domains/--domain-id--",
            "rel": "self"
        },
        "name": "another domain",
        "private_projects": true,
        "private_users": true
    }
]
```

Get domain: `GET /domains/{domain_id}`

Response:

```
Status: 200 OK

{
    "description": "desc of domain",
    "enabled": true,
```

```
    "id": "--domain-id--",
    "link": {
        "href": "http://identity:35357/v3/domains/--domain-id--",
        "rel": "self"
    },
    "name": "my domain",
        "private_projects": true,
        "private_users": false
}
```

Update domain: `PATCH /domains/{domain_id}`

The patching of the private-projects and proivate_users flag is not supported [Reason: for example if you set it from private to shared, then you might create clashes of user or project name in the shared name space.

Response:

```
Status: 200 OK

{
    "description": "desc of domain",
    "enabled": true,
    "id": "--domain-id--",
    "link": {
        "href": "http://identity:35357/v3/domains/--domain-id--",
        "rel": "self"
    },
    "name": "my domain",
        "private_projects": true,
        "private_users": false
}
```

Delete domain: `DELETE /domains/{domain_id}`

Response:

```
Status: 204 No Content
```

Get domain projects: `GET /domains/{domain_id}/projects`

query_string: page (optional) query_string: per_page (optional, default 30) query filter for "name", "enabled", or "domain_id" (optional)

Response:

```
Status: 200 OK

[
    {
        "domain_id": "--domain-id--",
        "enabled": true,
        "id": "--project-id--",
```

```
        "link": {
            "href": "http://identity:35357/v3/projects/--project-id--",
            "rel": "self"
        },
        "name": "a project name"
    },
    {
        "domain_id": "--domain-id--",
        "enabled": true,
        "id": "--domain-id--",
        "link": {
            "href": "http://identity:35357/v3/projects/--project-id--",
            "rel": "self"
        },
        "name": "another domain"
    }
]
```

Get domain users: `GET /domains/{domain_id}/users`

query_string: page (optional) query_string: per_page (optional, default 30) query filter for "name", "enabled", "email" (optional)

Response:

```
Status: 200 OK

[
    {
        "description": "a user",
        "email": "...",
        "enabled": true,
        "id": "--user-id--",
        "link": {
            "href": "http://identity:35357/v3/users/--user-id--",
            "rel": "self"
        },
        "name": "admin",
        "project_id": "--default-project-id--"
    },
    {
        "description": "another user",
        "email": "...",
        "enabled": true,
        "id": "--user-id--",
        "link": {
            "href": "http://identity:35357/v3/users/--user-id--",
            "rel": "self"
        },
        "name": "someone",
        "project_id": "--default-project-id--"
```

```
    }
]
```