GOCDB Grid Topology Information System

David.meredith@stfc.ac.uk (corresponding author)
John.Casson@stfc.ac.uk
George.Ryall@stfc.ac.uk
James.McCarthy@stfc.ac.uk

Contents

Executive Summary

GOCDB is a Configuration Management Database (CMDB[1]) for recording and managing assets in e-infrastructure projects. It defines a number of topology objects including projects, admin-domains, sites, services, service-groups, service-endpoints, service-downtimes, users and roles. GOCDB is the central CMDB for the EGI and EUDAT projects. The tool provides a web portal for editing information and a REST style programmatic interface (PI) for querying data in XML. Relationships between different objects are defined using a well constrained relational schema that closely resembles a sub-set of the GLUE 2[2] information model. A comprehensive role-based permissions model controls user permissions. Importantly, projects can self-manage their own users; users make requests for roles over target objects and users that already hold the necessary role(s) can accept or reject those role requests.

A flexible tag-cloud mechanism allows objects to be tagged with one or more 'scope-tags'. This allows resources to be tagged and grouped into multiple categories without duplication of information – this is essential to maintain the integrity of topology information across different infrastructures and projects. Different scope tags can be defined when necessary, for example, tags can be used to reflect different projects, infrastructure groupings and sub-projects. Resources can be flexibly 'filtered-by-tag' when querying for data via the programmatic interface (PI).

Core objects can also be extended using a powerful extensibility mechanism that allows custom key-value pairs to be added to those objects. These objects can then be flexibly 'filtered-by-custom-property' when selecting/ querying data.

An authentication abstraction layer has been integrated to allow different authentication mechanisms to be supported using a pluggable 'AuthenticationProvider' interface. Requests are authenticated using extensible 'Authentication' tokens. Implementations are provided for x509, SAML2 and username/password.

GOCDB supports multiple databases out-of-the-box through the use of the Doctrine[3] Object Relational Mapping library (ORM). A comprehensive DBUnit test suite ensures out of the box compatibility with Oracle and MySQL. Other databases such as Postgres should also be supported via Doctrine with a change to the DB connection settings. GOCDB provides an administration interface for common admin tasks. The codebase uses standards and established design patterns including MVC and ORM to provide a stable, easily customisable product.

[1] http://en.wikipedia.org/wiki/Configuration_management_database
[2] http://www.ogf.org/documents/GFD.147.pdf
[3] http://www.doctrine-project.org/projects/orm.html

Domain Model Overview

The GOCDB domain closely resembles a sub-set of the GLUE 2 Grid entity model. The GOCDB core entities are described below and their main relationships are summarised in Figure 1. The full entity relationship diagram (ERD) is provided in Appendix1. Since the structure of the GOCDB data model closely resembles GLUE 2, especially in terms of the entities and their relationships, we expect the structure of the data model to remain largely static. However, it is expected that new attributes will be added to the existing entities to populate more of the GLUE 2 attributes.

- **OwnedEntity**: An abstract super class that allows user Role objects to be linked to objects that extend OwnedEntity. This currently includes Project, NGI, Site and ServiceGroup.

- **Project** (extends OwnedEntity): A Project is a generic entity and can be defined multiple times in GOCDB. A Project can aggregate zero or more 'NGI' objects and is used to cascade Project level roles over its child NGIs.

- **NGI** (extends OwnedEntity): An 'NGI' is an EGI specific term but it simply reflects an administrative domain, corresponding to a GLUE 2 AdminDomain. An NGI aggregates zero or more Sites and can belong to one or more Projects. Users with roles over an NGI can have different permissions cascading over its child Sites. The GOCDB team have experience removing references to "NGI"s and other EGI specifics while collaborating with EUDAT on their GOCDB instance.

- **Site** (extends OwnedEntity): A Site represents a physical site with a location. A Site hosts zero or more Services. A Site also corresponds to a GLUE 2 AdminDomain. Users with roles over the Site have various permissions over a Site's services.

- **ServiceGroup** (extends OwnedEntity): A ServiceGroup is also known as a 'Virtual-Site'. It is used to group existing services that are physically distributed across multiple hosting sites into a virtual service grouping. Users with roles over a ServiceGroup do not have permissions over the aggregated services. Rather, a user must apply for a role over the service's hosting Site.

- **Service:** Represents an instance of a specific service and has a defined service-type enum value (e.g. 'org.service.type.X' or 'org.service.type.Y'). GOCDB does not currently distinguish between ComputingService and StorageService like GLUE 2. A Service defines zero or more Endpoint.

- **ServiceEndpoint**: (EndpointLocation). A ServiceEndpoint defines a network location/address for a service. An endpoint can be linked to zero or more Downtimes.

- **Downtime:** A Downtime object can be linked to one or more service endpoints. A key difference between GOCDB and GLUE 2 is that in GOCDB, a service endpoint can be linked to zero or many downtimes. This is required so that a history of past and pending downtimes can be recorded. In contrast, a GLUE 2 service can only publish a single set of downtime information for a particular service instance (usually the current or future downtime).

- **User**: Represents a user account. A User object owns one or more Role objects.
- **Role**: A Role joins and User and an OwnedEntity. It is used to define user permissions over the joined OwnedEntity. A user can own many Roles.
- **Scope**: A Scope entity defines a tag/label that can be associated to any entity that defines the 'IScopedEntity' interface. Entities implementing this interface include Site, NGI, Service (and by extension Downtime), ServiceGroup and Project.

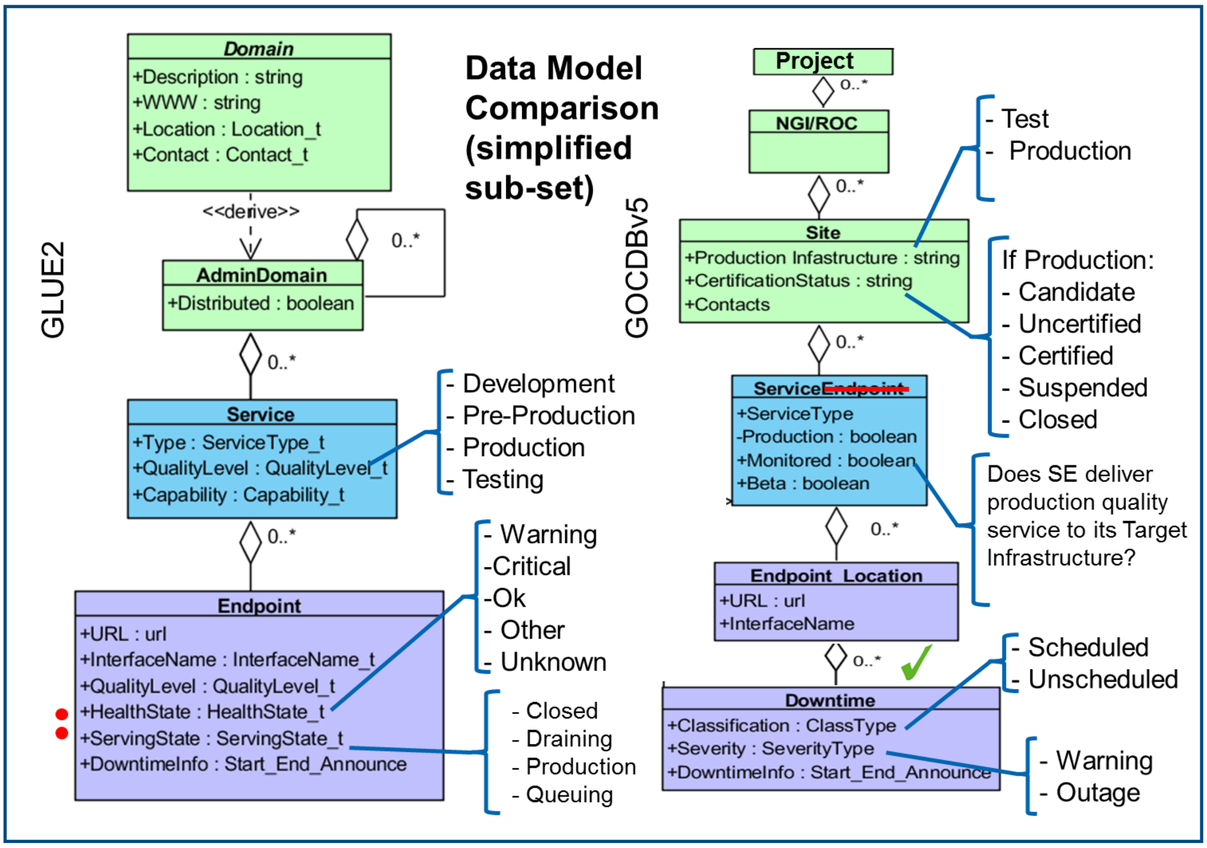


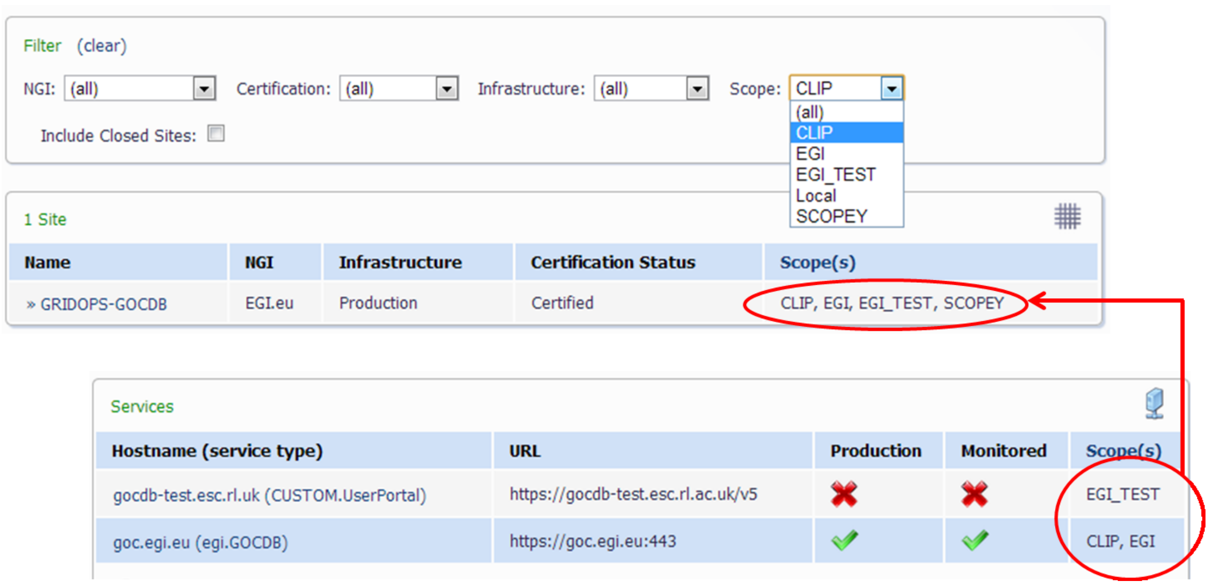
Figure 1. The GOCDB domain model (showing simplified sub-set) closely resembles a sub-set of the GLUE 2 Grid model, especially in terms of the entities and their corresponding relationships. To support GLUE 2, new attributes will need to be added to the existing GOCDB entities as/when required.

Multiple Projects and Scope Tags

Scope tags help organise resources into different categories and groupings. New tags can be added on request allowing users to tag their own resources with one or more scope-tags as necessary. The GOCDB admins control which scope tags are made available to avoid proliferation of tags (user defined tags are reserved for the extensibility mechanism). As shown in Figure 2, a site's scope list could aggregate all of the scopes defined by its child services. In doing this, the site scope list becomes a union of its service scopes plus any other site specific tags defined by the site. By defining scope tags, resources can be 'filtered-by-scope-tag' when querying for data in the PI using the 'scope' and 'scope_match' parameters (see the section on PI for details).

Clear Separation of Concerns

- It is important to understand that scopes and projects are distinct:
    - Projects are used to cascade roles and permissions over child objects
    - Scope-tags are used to filter resources into flexible categories/groupings
- Scope tags can be created to mirror the projects. For example, assuming two projects (e.g. EGI.eu and EU-DAT), two corresponding tags may be defined ('EGI' 'EUDAT').
- In addition, it is also possible define additional scopes for finer grained resource filtering e.g. 'CLIP' and 'EGI_TEST'.
- The key benefit: A clear separation of concerns between cascading permissions and resource filtering.



Name	NGI	Infrastructure	Certification Status	Scope(s)
» GRIDOPS-GOCDB	EGI.eu	Production	Certified	CLIP, EGI, EGI_TEST, SCOPEY

Hostname (service type)	URL	Production	Monitored	Scope(s)
gocdb-test.esc.rl.uk (CUSTOM.UserPortal)	https://gocdb-test.esc.rl.ac.uk/v5	✗	✗	EGI_TEST
goc.egi.eu (egi.GOCDB)	https://goc.egi.eu:443	✓	✓	CLIP, EGI

Figure 2. Sites and Services can be associated with one or more scope tags (other entities can also be tagged as required). In this example, the hosting site aggregates all of the scope tags defined by its child services. In doing this, the site scope list effectively becomes a union of the service scope tags + additional site specific scopes.

The Role Model and Permissions

- A user can request different Roles over different OwnedEntities.
- A Role object has a status of GRANTED or PENDING and links a User object to an OwnedEntity (see Figure 3).
- OwnedEntity is an abstract super class. Implementations currently include Project, Site, NGI and ServiceGroup.
- Role requests can be granted or revoked by users who already own the necessary roles over the specified OwnedEntity, or by users who have higher level roles over parent objects. The GOCDB administrators bootstrap this process by granting the initial role requests. Projects and users then subsequently manage their own role requests.
- A Role has a defined type. Different role types are used to define varying permissions, each enabling different 'Actions' over a target OwnedEntity. Role types include 'Site Administrator,' 'Site Ops Manager,' 'NGI Security Officer,' 'Chief Operations Officer' etc (not all listed here).
- Current Actions include EDIT_OBJECT, DELETE_OBJECT, GRANT_ROLE, REVOKE_ROLE, NGI_ADD_SITE (not all Actions are listed here).
- The role model is flexible and can be customised by adding new Role types and defining new Actions.
- The role model is simple to use by invoking the 'authoriseAction()' method and passing the requesting user and the required action that affects the target OwnedEntity, e.g. 'authoriseAction(Action::EDIT_OBJECT, $site, $user)'.

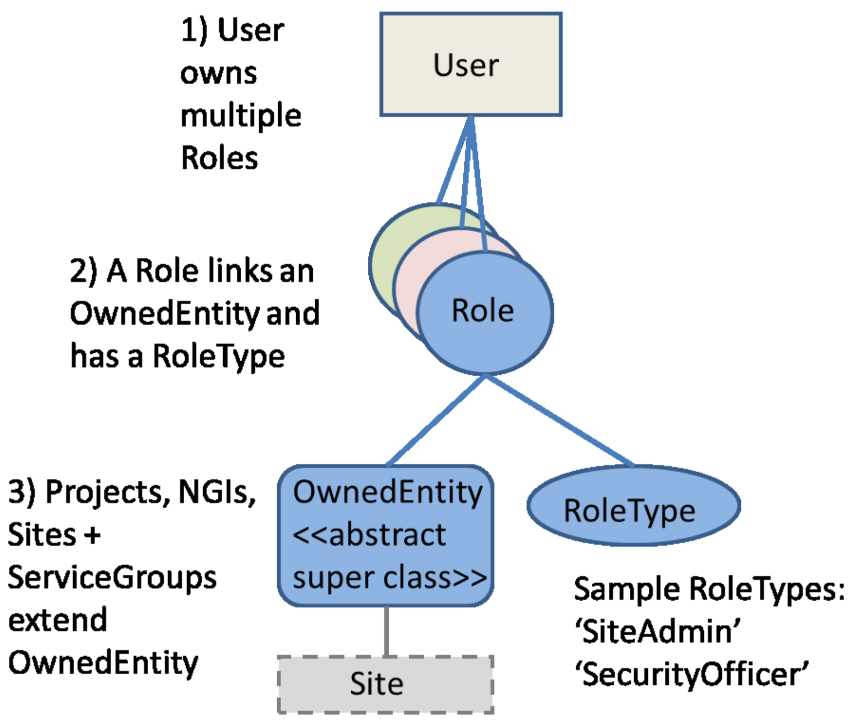
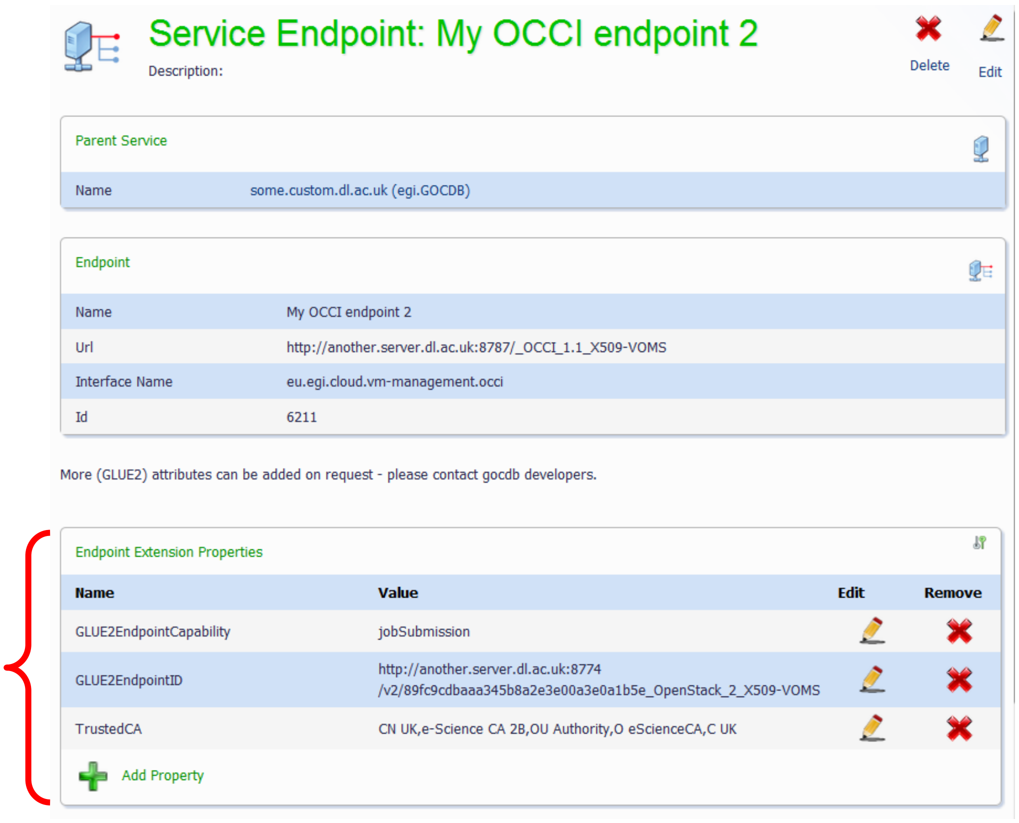

1) User owns multiple Roles
User
2) A Role links an OwnedEntity and has a RoleType
Role
3) Projects, NGIs, Sites + ServiceGroups extend OwnedEntity
OwnedEntity <<abstract super class>>
RoleType
Site
Sample RoleTypes: 'SiteAdmin' 'SecurityOfficer'

Figure 3. GOCDB Role model (simplified)

Custom Properties Extensibility Mechanism

Core objects can be extended using an extensibility mechanism that allows custom key-value pairs to be added to those objects. This allows flexible 'filter-by-custom-property' when selecting resources. For example, in the example below you could query for all service endpoints having a GLUE2EndpointCapability of 'jobSubmission' using:

- `get_service_endpoint&extensions=(GLUE2EndpointCapability=jobSubmission)`



Service Endpoint: My OCCI endpoint 2

Description:

Delete Edit

Parent Service

Name	some.custom.dl.ac.uk (egi.GOCDB)

Endpoint

Name	My OCCI endpoint 2
Url	http://another.server.dl.ac.uk:8787/_OCCI_1.1_X509-VOMS
Interface Name	eu.egi.cloud.vm-management.occi
Id	6211

More (GLUE2) attributes can be added on request - please contact gocdb developers.

Endpoint Extension Properties

Name	Value	Edit	Remove
GLUE2EndpointCapability	jobSubmission		
GLUE2EndpointID	http://another.server.dl.ac.uk:8774 /v2/89fc9cdbaaa345b8a2e3e00a3e0a1b5e_OpenStack_2_X509-VOMS		
TrustedCA	CN UK,e-Science CA 2B,OU Authority,O eScienceCA,C UK		

Add Property

Figure 4. Custom extension properties can be defined on core entities including Sites, Services and ServiceEndpoints. In this example, three custom properties are defined.

Within EGI Inspire, the custom extension properties have been used to advertise the pricing of resources using pre-agreed key-names such as 'P4U_Pilot_Grid_CPU,' 'P4U_Pilot_Cloud_Wall' and 'P4U_Pilot_VAT'. Custom properties allow folksonomy building and rapid prototyping for future extensions to the domain model.

Programmatic Interface (PI)

- GOCDB provides a comprehensive REST style programmatic interface (PI[4]) for querying the data in XML. Methods include; get_downtime, get_site, get_ngi, get_service_endpoint, get_site_contacts, get_service_types (not all methods are listed here).
- Queries can be refined by passing different URL parameters to narrow results, e.g. 'get_site&sitename=RAL_LCG2' to return just the specified site or 'get_site&country=UK' to return all UK sites.
- A number of core methods support the 'scope' and 'scope_match' parameters. The 'scope' parameter is used to specify a comma separated list of scope tags, and the 'scope_match' parameter is used to specify the value 'any' or 'all' ('any' means match all resources that define any of the specified scope tags, 'all' means only match those resources that define all of the specified scope tags). For example:
    - 'get_service_endpoint&scope=EGI,CLIP,PROJX&scope_match=any' (return all services that define either EGI, CLIP and/or PROJX scope tags)
    - 'get_site&scope=EGI,EUDAT&scope_match=all' (return all sites that define both EGI and EUDAT scope tags)
- A number of core methods support the 'extensions' property to allow results to be filtered by custom extension properties. The value part of a (k=v) pair can be omitted if filtering by value is not required (i.e. '(somekey=)' means select all resources that define the 'somekey' property with any value. (k=v) pairs can be optionally prefixed with one of following operators: AND, OR, NOT. Example where CPU_HS01_HOUR and CPU_HS02_HOUR are custom properties:
  ```
  get_service_endpoint&extensions=(CPU_HS01_HOUR=1)OR(CPU_HS02_H
  OUR=2)
  ```

Authentication Abstractions

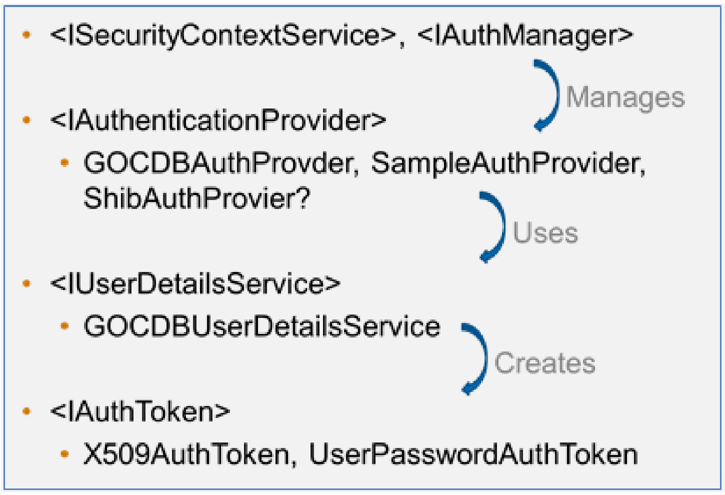
GOCDB includes core authentication abstractions to facilitate the plug-in of different user authentication mechanisms such as x509 certificates and SAML2 assertions. These core abstractions have been inspired by the Spring Security 3 framework (disclaimer: in no way is the GOCDB security module a full implementation of Spring Security 3[5], rather, it is a simplification).

In brief (and summarised in Figures 5 and 6):

- Client code access the FirewallComponentManager and selects a FirewallComponent for the current page request.
- The ISecurityContextService is used to store the users IAuthToken in HTTP session so that re-authentications are not necessary across different page requests (this requires cookies are enabled in the browser).
- The IAuthenticationProvider interface authenticates the user if their IAuthToken is null.
- The IUserDetailsService abstracts the local credential store (e.g. a local database that stores user accounts identified by certificate DN or username).

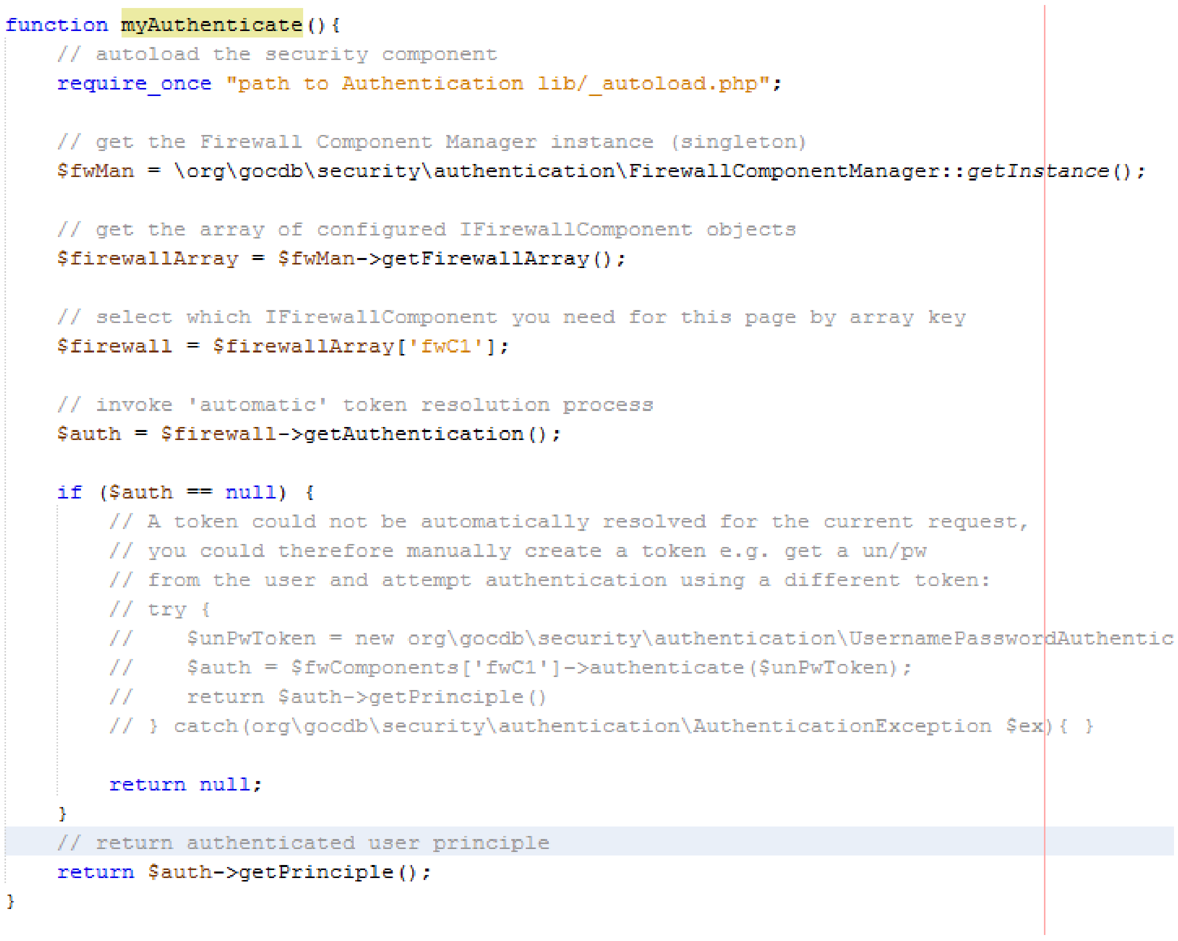
[4] https://wiki.egi.eu/wiki/GOCDB/PI/Technical_Documentation
[5] http://docs.spring.io/spring-security/site/index.html

The authentication module does not perform authorisation, this is delegated to the role model. The EGI implementation uses x509 and SAML2 Authentication tokens. There is plenty of scope to further refine and improve the authentication abstractions for use in different projects.



- <ISecurityContextService>, <IAuthManager>
- <IAuthenticationProvider>
  - GOCDBAuthProvder, SampleAuthProvider, ShibAuthProvier?
- <IUserDetailsService>
  - GOCDBUserDetailsService
- <IAuthToken>
  - X509AuthToken, UserPasswordAuthToken

Figure 5. Core abstractions in the authentication module



```
function myAuthenticate(){
    // autoload the security component
    require_once "path to Authentication lib/_autoload.php";

    // get the Firewall Component Manager instance (singleton)
    $fwMan = \org\gocdb\security\authentication\FirewallComponentManager::getInstance();

    // get the array of configured IFirewallComponent objects
    $firewallArray = $fwMan->getFirewallArray();

    // select which IFirewallComponent you need for this page by array key
    $firewall = $firewallArray['fwC1'];

    // invoke 'automatic' token resolution process
    $auth = $firewall->getAuthentication();

    if ($auth == null) {
        // A token could not be automatically resolved for the current request,
        // you could therefore manually create a token e.g. get a un/pw
        // from the user and attempt authentication using a different token:
        // try {
        //     $unPwToken = new org\gocdb\security\authentication\UsernamePasswordAuthentic
        //     $auth = $fwComponents['fwC1']->authenticate($unPwToken);
        //     return $auth->getPrinciple()
        // } catch(org\gocdb\security\authentication\AuthenticationException $ex){ }

        return null;
    }
    // return authenticated user principle
    return $auth->getPrinciple();
}
```

Figure 6. Sample usage of the authentication abstractions

Useful Links

SRC: https://github.com/GOCDB/gocdb

GOCDB Production EGI release: https://goc.egi.eu

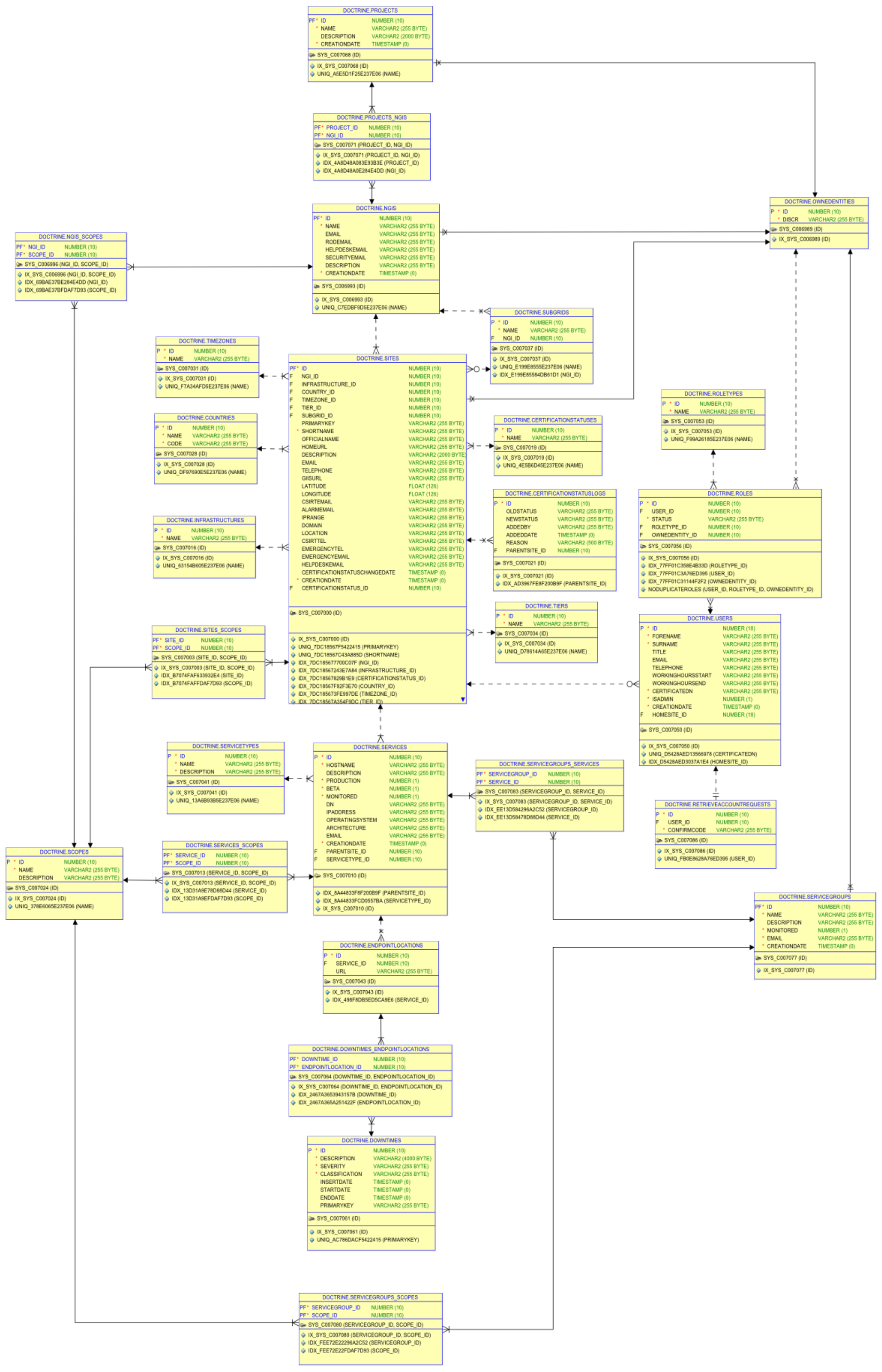
GOCDB Primary failover: https://goc.dl.ac.uk

GOCDB Wiki and user documentation: https://wiki.egi.eu/wiki/GOCDB_Documentation_Index

Future Developments: https://wiki.egi.eu/wiki/GOCDB/Release4/Development

Programmatic Interface doc: https://wiki.egi.eu/wiki/GOCDB/PI/Technical_Documentation

Appendix 1 – GOCDB Entity Relationship Diagram (ERD)


DOCTRINE.PROJECTS
PF* ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
DESCRIPTION VARCHAR2 (2000 BYTE)
* CREATIONDATE TIMESTAMP (0)
SYS_C007068 (ID)
IX_SYS_C007068 (ID)
UNIQ_A5E5D1F25E237E06 (NAME)
DOCTRINE.PROJECTS_NGIS
PF* PROJECT_ID NUMBER (10)
PF* NGI_ID NUMBER (10)
SYS_C007071 (PROJECT_ID, NGI_ID)
IX_SYS_C007071 (PROJECT_ID, NGI_ID)
IDX_4A8D48A083E93B3E (PROJECT_ID)
IDX_4A8D48A0E284E4DD (NGI_ID)
DOCTRINE.NGIS
PF* ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
EMAIL VARCHAR2 (255 BYTE)
RODEMAIL VARCHAR2 (255 BYTE)
HELPDESKEMAIL VARCHAR2 (255 BYTE)
SECURITYEMAIL VARCHAR2 (255 BYTE)
DESCRIPTION VARCHAR2 (255 BYTE)
* CREATIONDATE TIMESTAMP (0)
SYS_C006993 (ID)
IX_SYS_C006993 (ID)
UNIQ_C7EDBF9D5E237E06 (NAME)
DOCTRINE.NGIS_SCOPES
PF* NGI_ID NUMBER (10)
PF* SCOPE_ID NUMBER (10)
SYS_C006996 (NGI_ID, SCOPE_ID)
IX_SYS_C006996 (NGI_ID, SCOPE_ID)
IDX_69BAE37BE284E4DD (NGI_ID)
IDX_69BAE37BFDAF7D93 (SCOPE_ID)
DOCTRINE.OWNEDENTITIES
P * ID NUMBER (10)
* DISCR VARCHAR2 (255 BYTE)
SYS_C006989 (ID)
IX_SYS_C006989 (ID)
DOCTRINE.SUBGRIDS
P * ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
F NGI_ID NUMBER (10)
SYS_C007037 (ID)
IX_SYS_C007037 (ID)
UNIQ_E199E8555E237E06 (NAME)
IDX_E199E85584DB61D1 (NGI_ID)
DOCTRINE.TIMEZONES
P * ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
SYS_C007031 (ID)
IX_SYS_C007031 (ID)
UNIQ_F7A34AFD5E237E06 (NAME)
DOCTRINE.SITES
PF* ID NUMBER (10)
F NGI_ID NUMBER (10)
F INFRASTRUCTURE_ID NUMBER (10)
F COUNTRY_ID NUMBER (10)
F TIMEZONE_ID NUMBER (10)
F TIER_ID NUMBER (10)
F SUBGRID_ID NUMBER (10)
PRIMARYKEY VARCHAR2 (255 BYTE)
* SHORTNAME VARCHAR2 (255 BYTE)
OFFICIALNAME VARCHAR2 (255 BYTE)
HOMEURL VARCHAR2 (255 BYTE)
DESCRIPTION VARCHAR2 (2000 BYTE)
EMAIL VARCHAR2 (255 BYTE)
TELEPHONE VARCHAR2 (255 BYTE)
GIISURL VARCHAR2 (255 BYTE)
LATITUDE FLOAT (126)
LONGITUDE FLOAT (126)
CSIRTEMAIL VARCHAR2 (255 BYTE)
ALARMEMAIL VARCHAR2 (255 BYTE)
IPRANGE VARCHAR2 (255 BYTE)
DOMAIN VARCHAR2 (255 BYTE)
LOCATION VARCHAR2 (255 BYTE)
CSIRTTEL VARCHAR2 (255 BYTE)
EMERGENCYTEL VARCHAR2 (255 BYTE)
EMERGENCYEMAIL VARCHAR2 (255 BYTE)
HELPDESKEMAIL VARCHAR2 (255 BYTE)
CERTIFICATIONSTATUSCHANGEDATE TIMESTAMP (0)
* CREATIONDATE TIMESTAMP (0)
F CERTIFICATIONSTATUS_ID NUMBER (10)
SYS_C007000 (ID)
IX_SYS_C007000 (ID)
UNIQ_7DC18567F5422415 (PRIMARYKEY)
UNIQ_7DC18567C43A885D (SHORTNAME)
IDX_7DC185677700C07F (NGI_ID)
IDX_7DC18567243E7A84 (INFRASTRUCTURE_ID)
IDX_7DC18567829B1E9 (CERTIFICATIONSTATUS_ID)
IDX_7DC18567F92F3E70 (COUNTRY_ID)
IDX_7DC185673FE997DE (TIMEZONE_ID)
IDX_7DC18567A354F9DC (TIER_ID)
DOCTRINE.COUNTRIES
P * ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
* CODE VARCHAR2 (255 BYTE)
SYS_C007028 (ID)
IX_SYS_C007028 (ID)
UNIQ_DF97690E5E237E06 (NAME)
DOCTRINE.INFRASTRUCTURES
P * ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
SYS_C007016 (ID)
IX_SYS_C007016 (ID)
UNIQ_63154B605E237E06 (NAME)
DOCTRINE.ROLETYPES
P * ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
SYS_C007053 (ID)
IX_SYS_C007053 (ID)
UNIQ_F99A26185E237E06 (NAME)
DOCTRINE.CERTIFICATIONSTATUSES
P * ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
SYS_C007019 (ID)
IX_SYS_C007019 (ID)
UNIQ_4E5B6D45E237E06 (NAME)
DOCTRINE.CERTIFICATIONSTATUSLOGS
P * ID NUMBER (10)
OLDSTATUS VARCHAR2 (255 BYTE)
NEWSTATUS VARCHAR2 (255 BYTE)
ADDEDBY VARCHAR2 (255 BYTE)
ADDEDDATE TIMESTAMP (0)
REASON VARCHAR2 (500 BYTE)
F PARENTSITE_ID NUMBER (10)
SYS_C007021 (ID)
IX_SYS_C007021 (ID)
IDX_AD3967FE8F200B9F (PARENTSITE_ID)
DOCTRINE.ROLES
P * ID NUMBER (10)
F USER_ID NUMBER (10)
* STATUS VARCHAR2 (255 BYTE)
F ROLETYPE_ID NUMBER (10)
F OWNEDENTITY_ID NUMBER (10)
SYS_C007056 (ID)
IX_SYS_C007056 (ID)
IDX_77FF01C358E4B33D (ROLETYPE_ID)
IDX_77FF01C3A76ED395 (USER_ID)
IDX_77FF01C31144F2F2 (OWNEDENTITY_ID)
NODUPLICATEROLES (USER_ID, ROLETYPE_ID, OWNEDENTITY_ID)
DOCTRINE.TIERS
P * ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
SYS_C007034 (ID)
IX_SYS_C007034 (ID)
UNIQ_D78614A65E237E06 (NAME)
DOCTRINE.USERS
P * ID NUMBER (10)
* FORENAME VARCHAR2 (255 BYTE)
* SURNAME VARCHAR2 (255 BYTE)
TITLE VARCHAR2 (255 BYTE)
EMAIL VARCHAR2 (255 BYTE)
TELEPHONE VARCHAR2 (255 BYTE)
WORKINGHOURSSTART VARCHAR2 (255 BYTE)
WORKINGHOURSEND VARCHAR2 (255 BYTE)
* CERTIFICATEDN VARCHAR2 (255 BYTE)
* ISADMIN NUMBER (1)
* CREATIONDATE TIMESTAMP (0)
F HOMESITE_ID NUMBER (10)
SYS_C007050 (ID)
IX_SYS_C007050 (ID)
UNIQ_D5428AED13566978 (CERTIFICATEDN)
IDX_D5428AED3037A1E4 (HOMESITE_ID)
DOCTRINE.SITES_SCOPES
PF* SITE_ID NUMBER (10)
PF* SCOPE_ID NUMBER (10)
SYS_C007003 (SITE_ID, SCOPE_ID)
IX_SYS_C007003 (SITE_ID, SCOPE_ID)
IDX_B7074FAF633932E4 (SITE_ID)
IDX_B7074FAFFDAF7D93 (SCOPE_ID)
DOCTRINE.SERVICETYPES
P * ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
* DESCRIPTION VARCHAR2 (255 BYTE)
SYS_C007041 (ID)
IX_SYS_C007041 (ID)
UNIQ_13A6B93B5E237E06 (NAME)
DOCTRINE.SERVICES
P * ID NUMBER (10)
* HOSTNAME VARCHAR2 (255 BYTE)
DESCRIPTION VARCHAR2 (255 BYTE)
* PRODUCTION NUMBER (1)
* BETA NUMBER (1)
* MONITORED NUMBER (1)
DN VARCHAR2 (255 BYTE)
IPADDRESS VARCHAR2 (255 BYTE)
OPERATINGSYSTEM VARCHAR2 (255 BYTE)
ARCHITECTURE VARCHAR2 (255 BYTE)
EMAIL VARCHAR2 (255 BYTE)
* CREATIONDATE TIMESTAMP (0)
F PARENTSITE_ID NUMBER (10)
F SERVICETYPE_ID NUMBER (10)
SYS_C007010 (ID)
IDX_8A44833F8F200B9F (PARENTSITE_ID)
IDX_8A44833FCD0557BA (SERVICETYPE_ID)
IX_SYS_C007010 (ID)
DOCTRINE.SERVICEGROUPS_SERVICES
PF* SERVICEGROUP_ID NUMBER (10)
PF* SERVICE_ID NUMBER (10)
SYS_C007083 (SERVICEGROUP_ID, SERVICE_ID)
IX_SYS_C007083 (SERVICEGROUP_ID, SERVICE_ID)
IDX_EE13D584296A2C52 (SERVICEGROUP_ID)
IDX_EE13D58478D88D44 (SERVICE_ID)
DOCTRINE.RETRIEVEACCOUNTREQUESTS
P * ID NUMBER (10)
F USER_ID NUMBER (10)
* CONFIRMCODE VARCHAR2 (255 BYTE)
SYS_C007086 (ID)
IX_SYS_C007086 (ID)
UNIQ_FB0E8628A76ED395 (USER_ID)
DOCTRINE.SCOPES
P * ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
DESCRIPTION VARCHAR2 (255 BYTE)
SYS_C007024 (ID)
IX_SYS_C007024 (ID)
UNIQ_378E6065E237E06 (NAME)
DOCTRINE.SERVICES_SCOPES
PF* SERVICE_ID NUMBER (10)
PF* SCOPE_ID NUMBER (10)
SYS_C007013 (SERVICE_ID, SCOPE_ID)
IX_SYS_C007013 (SERVICE_ID, SCOPE_ID)
IDX_13D31A9E78D88D44 (SERVICE_ID)
IDX_13D31A9EFDAF7D93 (SCOPE_ID)
DOCTRINE.SERVICEGROUPS
PF* ID NUMBER (10)
* NAME VARCHAR2 (255 BYTE)
DESCRIPTION VARCHAR2 (255 BYTE)
* MONITORED NUMBER (1)
* EMAIL VARCHAR2 (255 BYTE)
* CREATIONDATE TIMESTAMP (0)
SYS_C007077 (ID)
IX_SYS_C007077 (ID)
DOCTRINE.ENDPOINTLOCATIONS
P * ID NUMBER (10)
F SERVICE_ID NUMBER (10)
URL VARCHAR2 (255 BYTE)
SYS_C007043 (ID)
IX_SYS_C007043 (ID)
IDX_498F8DB5ED5CA9E6 (SERVICE_ID)
DOCTRINE.DOWNTIMES_ENDPOINTLOCATIONS
PF* DOWNTIME_ID NUMBER (10)
PF* ENDPOINTLOCATION_ID NUMBER (10)
SYS_C007064 (DOWNTIME_ID, ENDPOINTLOCATION_ID)
IX_SYS_C007064 (DOWNTIME_ID, ENDPOINTLOCATION_ID)
IDX_2467A3653943157B (DOWNTIME_ID)
IDX_2467A365A251422F (ENDPOINTLOCATION_ID)
DOCTRINE.DOWNTIMES
P * ID NUMBER (10)
* DESCRIPTION VARCHAR2 (4000 BYTE)
* SEVERITY VARCHAR2 (255 BYTE)
* CLASSIFICATION VARCHAR2 (255 BYTE)
INSERTDATE TIMESTAMP (0)
STARTDATE TIMESTAMP (0)
ENDDATE TIMESTAMP (0)
PRIMARYKEY VARCHAR2 (255 BYTE)
SYS_C007061 (ID)
IX_SYS_C007061 (ID)
UNIQ_AC786DACF5422415 (PRIMARYKEY)
DOCTRINE.SERVICEGROUPS_SCOPES
PF* SERVICEGROUP_ID NUMBER (10)
PF* SCOPE_ID NUMBER (10)
SYS_C007080 (SERVICEGROUP_ID, SCOPE_ID)
IX_SYS_C007080 (SERVICEGROUP_ID, SCOPE_ID)
IDX_FEE72E22296A2C52 (SERVICEGROUP_ID)
IDX_FEE72E22FDAF7D93 (SCOPE_ID)