ENGEREK
KULLANICI YÖNETİM SİSTEMİ

KURULUM, KONFİGÜRASYON, MİMARİ YAPI NOTLARI

İçindekiler

Başlangıç

Bu sayfalarda, Engerek uygulamasında atılacak ilk adımlar anlatılmaktadır. Ürünün kurulumu, gerekli ayarlar, bir hedef kaynağın ilk olarak tanımlanması ve temel provizyonlama adımlarından bahsedilmektedir. Temel olarak;

- İlk Yapılacaklar
- Kurulum Kılavuzu
- Bağlayıcı (konnektör) kılavuz
- Temel mimari ve veri yapısı bu bölümde anlatılmaktadır.

İlk Yapılacaklar

Engerek Kurulumu

Kurulum sonrasında, Engerek ürünü neredeyse boş bir veritabanı ile oluşmaktadır. Yazılımın kullanıcı ihtiyaçlarını karşılayacak tam işlevsellik için yapılandırılması gereklidir. Detaylı kurulum adımları için “Kurulum Kılavuzu” bölümünü inceleyiniz.

Ürüne İlk Giriş

Engerek ürünü, yönetici uygulamasına giriş için aşağıdakine benzer bir bağlantı kullanılabilir ;

http://hostname:8080/engerek/admin

Kurulum sonrası, Engerek ürünü önceden tanımlanmış bir yönetici hesabını barındıracak ve ürüne ilk giriş için bu hesap kullanılacaktır.

KullanıcıAdı	***administrator***
Parola	***5ecr3t***

Bu hesap, Engerek veritabanında normal bir kullanıcı hesabı olarak tanımlanmıştır. Bu nedenle, daha sonra değiştirebilir, başka yönetici hesapları oluşturabilirsiniz.

Konfigürasyon Dosyalarının Yönetimi

Engerek, XML konfigürasyon dosyaları kullanır ve bu dosyalar dahili ayrıştırıcılar (xml Parser) ile işlenir. Dışarıdan yüklediğiniz XML dosyaları, sistem tarafından doğrulanıp saklandığında, yine sistem tarafından bazı notasyonlar ile güncellenerek, okunması ve anlaşılması zor bir yapıya dönüşebilir. Bu nedenle, tekrar yüklenmesi gerekebilecek dosyaların değişmesi ihtimaline karşılık, bu dosyaların orjinal kopyasını bulundurmanız ve dahili XML editor yerine dışarıda orjinal dosya üzerinde çalışmanız daha kolay olacaktır.
Genel olarak, konfigürasyon dosyalarının ayrı bir konfigürasyon / kod versiyonlama aracı ile yönetilmesi tercih edilmelidir.

Temel Kullanıcı Yönetimi

Ürünün kurulumu sonrasında, hesap yönetimi adına yapılacak pek bir işlem yoktur. Kurumun personeli, müşterileri, danışmanları gibi fiziksel kullanıcılarının tanımlandığı ortam hazırlanmış olur, ve “`Kullanıcılar`” menü komutları ile bu kullanıcılar tanımlanabilir. Engerek ürünü, KYS ortamlarında genel olarak bulunması gereken temel özellik setini sağlar. Öntanımlı şema ile gelen bu özellikler ilgili sayfadaki bir pencerede gösterilmektedir. Tüm kullanıcı özellikleri listesi çok büyük olduğundan, sadece temel bir kısmı listelenmektedir. Tüm özellikleri listelemek için, pencerinin “Detaylar” kesiminin sağ üst köşesinde yer alan “`Boş alanları göster`” butonu kullanılabilir. Bu özellik listesinin yeterli olmadığı durumda, Engerek sisteminin özelleştirilmesi sürecinde şemada yeni özellikler tanımlanması mümkündür.

Menüden, “`Kullanıcılar`” -> “`Yeni Kullanıcı`” komutunu seçin. Kullanıcı oluşturmak için gerekli minimum alanları doldurun. Özellikle Kullanıcı kodunu oluşturacak olan “`Kodu`” alanına değer girin. Kırmızı * ile işaretli tüm zorunlu alanlara değer girilmesi gereklidir. Bu işlem ile Engerek kullanıcı deposunda yeni bir kullanıcı oluşturulacaktır. Bu kayıt, provizyonlama sistemindeki tüm kayıtlar için ana hesap kaydıdır. Bu aşamada, herhangi bir hedef kaynağın güncellenmesi söz konusu değildir.

Doldurulan bu kullanıcı formu gönderildiğinde, kullanıcı kaydının oluştuğu veritabanında görülmelidir. Bu listeyi görmek için “`Kullanıcılar`” -> “`Kullanıcı Listesi`” komutu kullanılabilir.

Listenin herhangi bir satırında yer alan kullanıcı adı seçilerek, ilgili kullanıcı için "`Kullanıcı Sayfası`" açılabilir. Kullanıcı verilerini birkaç ayrı panelde görüntülenecektir. Bu kullanıcıya herhangi bir kaynak atanamayacaktır, çünkü sistemde henüz hedef bir kaynak tanımı yapılmamıştır.

Kaynaklar ve Hesaplar

Yukarıda tanımlanan kullanıcılar, Engerek kullanıcı deposunda bulunmaktadır. Kimlik Yönetim Sisteminin temel bir hedefi, kullanıcının sistemlerdeki hesaplarını yönetmektir. Hesap, kullanıcıyı hedef sistemlerde tanımlayan veri yapısıdır. Örneğin bir kullanıcının Active Dierctory, LDAP, Solaris hesapları bulunabilir. Hesapların tanımlandığı bu hedef sistemler, Engerek sistemi tarafından **kaynak** olarak bilinir. Engerek, hesap yönetimi yaptığı bu kaynaklar hakkında biraz detay bilgiye sahip olmalıdır. Sahip olunması gereken bilgiler, kaynağın türü, sunucu adı, iletişim kurulacak port bilgisi, yönetici hesabı ve parolası, kullanıcı hesabı için şablon yapısı, özellik senkronizasyon tanımları sayılabilir. Bundan sonraki paragraflarda, bu tür kaynak tanımlama için örnekler verilmektedir.

	Anlamı	Tutulduğu depo	Örnek
Kullanıcı	***Fiziksel kullanıcı***	***5ecr3t***	***Ahmet Demir Fatma Keskin***
Hesap	***Hedef sisteme erişen veri kaydı***	***Kaynak***	***Ademir123 uid=fkeskin,ou=people,dc=ulakbim,dc=gov,dc=tr***
Kaynak	***Engerek uygulaması tarafından yönetilen hedef sistem***	***-***	***Ldap sunucu – tst.ulakbim.gov.tr, Bayi sistemi– Active Directory sunucusu***

OpenDJ Kaynak Konfigürasyonu

Bu bölümde, kaynak olarak OpenDJ Ldap sistemi örneklenmektedir. Kurulumu ve kullanımı kolay ve açık kaynak bir ürün olduğu için, Engerek sistemi ile hesap yönetimi testleri için, OpenDJ iyi bir örnektir.

Bir kaynakta hesap yönetimi yapılabilmesi için, kaynak tanımlama Engerek sisteminde yapılmalı ve bu kaynağın da kurulmuş olmalıdır. Bu nedenle, OpenDJ ortamı da entegrasyon öncesi hazırlanmalıdır.

Kaynak Tanımının Engerek Sistemine Aktarılması

Kaynak Tanımı, Engerek sistemindeki kaynak parametrelerini tanımlayan XML yapısının bir parçasıdır. Kaynağa erişim için kullanılacak sunucu adı, port numarası, yetkili hesap ve parolası gibi parametreleri tanımlayan bağlayıcı (connector) bilgisini içermektedir. Bu parametreler kaynak türüne göre farklılaşabilir. Engerek sisteminin bağlantı yapacağı herbir kaynak için tanımlar yapılmalıdır.

Örnek olarak düşünülen OpenDJ sistemine bağlantı için kullanılacak XML dosya, (***opendj-localhost-resource-sync-advanced.xml***) **<baseDirectory>/ornek/kaynaklar/opendj** klasöründe bulunabilir. Bu dosyanın Engerek sistemine aktarılması için doğrulanması gerekecektir.

Kaynak tanımının Engerek sistemine aktarılması için http://<engerek_url>/engerek bağlantısı ile sisteme giriş yapılarak, “`Konfigürasyon`” -> “`İçe nesne aktar`” menü komutu kullanılmalıdır. Örnek olarak kullanılacak (***opendj-localhost-resource-sync-advanced.xml vb***) dosya açılarak, tüm içerik kopyala-yapıştır yöntemi ile sayfadaki metin penceresine alınmalı ve “`Nesneyi içe Aktar`” butonu ile işlem başlatılmalıdır.

Herhangi bir problemden dolayı aynı kaynak dosyası tekrar içe aktarılmak istenirse, “`Mevcut nesne üzerine yaz`” kutucuğu işaretlenmiş olmalıdır.

Hesap Yönetimi

Kaynak tanımı yapıldıktan sonra, bu kaynakta hesap oluşturulabilirsiniz. Kaynaklarda doğrudan hesap tanımlanmayıp, Engerek sisteminde daha önce tanımlanmış olan kullanıcılar için kaynak atama yöntemi ile hesap oluşturmak mümkün olmaktadır. Bu yöntem, Kimlik Yönetim Sisteminin temel amacıdır. Kullanıcı seçilerek ilgili kaynakta yeni hesap oluşturulacaktır.

“Kullanıcılar” -> “Kullanıcı Listesi” komutu ile var olan kullanıcılar listelenir. KullanıcıKodu’na tıklanarak bir kullanıcı seçilir. “Kullanıcı Detayları” sayfasının sağ tarafındaki panel, bu kullanıcının sahip olduğu atamalar ve hesapları göstermektedir. İlk başta bu kısım boş olarak gelmelidir.

Bu panelin altında yer alan “Atamalar” alanının yanında yer alan düzenleme ikonuna basılarak açılan menüde “Kaynak Hesabı ata” komutu ile Kaynak seçim sayfası açılır, listedeki OpenDJ kaynağı işaretlenir ve sayfanın altındaki “Ata” butonu ile kaynakta hesap oluşturulur.

Yeni hesabın detaylarını gösteren form, Atamalar panelinde görüntülenebilir durumda olmalıdır. Bu formda bazı alanlar * (mavi yıldız) ile belirtilmiş olabilir. Bu alanların değerleri , kaynak tanımında belirtilen ifade (expression) ile hesaplanarak otomatik olarak doldurulmuş ve tekrar girilmesine gerek kalmamıştır. Diğer alanlar için istenen değerler manuel olarak girilmelidir. * (kırmızı yıldız) ile işaretli alanlar zorunludur ve boş geçilmemelidir.

Girilen değerler ile yeni hesap oluşturulması için, en alt satırda yer alan “Kaydet” butonu kullanılmadır. Kayıt gerçekleşince, yeşil bir panel içinde “*Değişiklikler kaydedildi*” gibi bir mesaj görülür. Bu durumda, ilgili hesap, Kullanıcı Detayları kısmında listelenecektir. Hesabın doğru değerler ile oluşturulduğu OpenDJ sisteminde de kontrol edilmelidir.

Son kullanıcı Arayüzü Kullanımı

Son kullanıcılar, yönetici rollerine sahip olmayan, yani kaynaklar ve diğer kullanıcılar üstünde işlem yapamayan, sadece kendileri için izin verilen ve kendi hesapları üzerinde işlem yapan kullanıcılardır. Engerek sistemine, aynı arayüzden erişerek işlem yapabilirler. İlgili sayfalar, sahip oldukları yetkilere göre düzenlenecektir. Kullanıcı, kullanıcı adı ve şifresiyle sisteme giriş yaptıktan sonar ulaştığı sayfada “Kullanıcı Profili” linkine tıklayarak kendi bilgilerini ve üzerindeki Projeksiyon, Organizayon ve Atamalarını görüntüleyebilir.

Atamalar

Bu alan, giriş yapan kullanıcının sahip olduğu kaynak hesapları, alanlarını ve Rollerini görüntülemek için kullanılır. Alt başlıklara tıklayarak ilgili bölüm açılıp kapatılabilir.

OrganizasyonlarBu bölümde, giriş yapan kullanıcının dahil olduğu organizayonlar gösterilmektedir.

Projeksiyonlar

Bu bölümde, giriş yapan kullanıcının provizyonlandığı kaynaklar gösterilmektedir.

Detaylar

Bu bölümde, giriş yapan kullanıcının en temel bilgileri görüntülenmektedir. “`Düzenle`” seçeneği ile, daha önceden girilmiş olan bazı alanların değiştirilmesi mümkün olabilir. Bu alanlardaki değişiklikler, kaynak hesaplarına otomatik olarak aktarılacaktır.

Ara

Bu seçenek, Engerek sisteminde kayıtlı olan kullanıcıların belli özelliklere göre aranıp listelenmesini sağlar.

Kurulum Kılavuzu

Giriş

Bu kılavuz, Engerek KYS ürününün derlenmiş dağıtımından nasıl kurulacağına dair adımları içerir. Engerek KYS, WAR şeklinde dağıtımı yapılan açık kaynak kodlu bir Java Web uygulamasıdır.

Uygulamanın çalıştırılması için tek gereken uygun bir uygulama sunucusudur. Bu kılavuz, Engerek KYS’nin Apache Tomcat üzerinde nasıl çalıştırılacağını

anlatmaktadır. Bunun yanında, burada anlatılanlar olabildiğince genel geçer olarak anlatılmış olup, diğer ortamlarda çalıştırılması için de yol gösterici olması amaçlanmıştır.

Başlamadan önce

Sürüm Notları

Lütfen yüklemesini yapmak istediğiniz sürümün bilinen sorunları veya ortam uyuşmazlıkları için ürünün “Sürüm Notları” kısmına göz atınız.

Gereksinimler

Java SE Development Kit 7

Uygulamanın sorunsuz çalıştırılabilmesi için, aşağıdaki linkten indirilebilecek JDK7 ürünün kurulu olması gerekmektedir.

http://www.oracle.com/technetwork/java/javase/downloads/index.html.

Sisteminizin JAVA_HOME çevre değişkeninin (environment variable) JDK’nın yüklenmiş olduğu yola işaret etmesi gerekir.

Lütfen Java 6 platformunun artık desteklenmediğini (Bazı durumlarda çalışır olsa da) unutmayınız.

Apache Tomcat

Engerek KYS çalışmak için bir uygulama sunucusuna ihtiyaç duymaktadır. Şu an için Apache Tomcat 6.x veya 7.x versiyonları desteklenmektedir. Sonraki sürümlerde daha fazla uygulama sunucusu desteğinin geleceği öngörülmektedir.

Apache Tomcat’i http://tomcat.apache.org/download-70.cgi adresinden indirebilirsiniz. Tomcat’in yüklendiği hedef dizin dökümanın ileriki bölümlerinde <tomcat> olarak gösterilmiştir.

Windows platformu tercihi durumunda, Apache Tomcat’in çalıştırılabilir dağıtımını indirmek yerine, zip olarak indirilmesi tavsiye edilmektedir. Tomcat sunucusunu,

Windows'ta <tomcat>\bin dizinindeki startup.bat dosyası aracılığı ile başlatabilirsiniz.

Sistem Gereksinimleri

Önerilen sistem gereksinimleri

CPU: Çift Çekirdekli CPU

RAM: 4GB

HDD: 2GB

Minimum sistem gereksinimleri

CPU: 1800Mhz Athlon XP 2500+

RAM: 1GB

HDD: 200MB

Kurulum

Engerek Uygulamasının İndirilmesi

Engerek uygulamasının derlenmiş paketi, Pardus Kurumsal depolarından, kaynak kodları ise,

Git.Pardus.org.tr/Engerek bağlantısından indirilebilir.

Engerek.WAR'ı kendi belirlediğiniz bir dizine çıkartınız. Kılavuzun kalanında bu dizine değinilecektir.

Engerek Ana Klasörü

Eğer özel konektörler ve/veya şema kullanacaksanız, Engerek'in kullanacağı, konfigürasyon dosyalarını da içeren bir klasör oluşturmanız gerekmektedir. Engerek, bu klasör içinde gerekli dosyaları ilk açılışta oluşturacaktır. Eğer gerekli dosyalar mevcut ise, uygulama her açılışta veya yeniden başlatmada aynı işlemi tekrar yapmayacaktır. Bu klasör, başka bir ayar yapılmadığı takdirde, repository datalarının tutulduğu bütünleşik veritabanı dosyaları için de kullanılacaktır. Eğer ürün üzerinde

özelliştirme yapmayarak sadece Engerek'i deneme amaçlı kuruyor iseniz bu adımı geçerek direk olarak Veritabanı Başlatma' bölümünden devam edebilirsiniz.

Engerek ana klasörünü bilgisayarınızdaki herhangi bir yerde oluşturabilirsiniz, sadece oluşturduğunuz klasöre uygulamanın yazma hakları olduğuna dikkat ediniz.

Engerek' için oluşturduğunuz bu klasörü tanımlamak için kullandığınız sunucuda, Tomcat başlatılırken gereken bir Java parametresi geçirilmesi gerekir.

Tomcat için ***<tomcat>/bin/catalina.sh*** " dosyasında "JAVA_OPTS" değişkeni olarak aşağıdaki gibi bir tanımlama yapabilirsiniz.

```
...
JAVA_OPTS="$JAVA_OPTS -Dmidpoint.home=/opt/engerek -XX:MaxPermSize=256m"
...
```

Windows platformlarında ise benzer şekilde bin klasöründe bulunan catalina.bat dosyasında alttaki gibi tanımlama yapılabilir.

```
...
set JAVA_OPTS=%JAVA_OPTS% -Dmidpoint.home=c:/engerek -XX:MaxPermSize=256m
...
```

Eğer uygulama çalıştırılırken özel bir -Dmidpoint.home parametresi bulamazsa, sistemde kullanıcı klasörü altında bir klasörü kendiliğinden oluşturacaktır.

Veritabanı Başlatma

Engerek KYS özellikle demo ve test amaçlı kullanmayı oldukça kolaylaştıran, bütünleşik bir veritabanı ile gelmektedir.(H2)

İlk açılışta Engerek otomatik olarak bu veritabanını kullanacağından ayrıca bir veritabanı konfigürasyonu yapılmasına gerek yoktur. Bu veritabanı bir önceki adımda bahsedilen Engerek Ana Klasöründe tutulacaktır.

Sistem aynı zamanda Mysql,PostgreSql,Mariadb vb. gibi tam ölçekli veritabanları ile de çalışabilmektedir. Bu konuda daha detaylı bilgi için “Veri Deposu (Repository)” Konfigürasyonu dökümanına bakınız.

Engerek KYS Yüklemesi

Engerek KYS, engerek.war dosyası aracılığı ile deploy edilir. Bunun için WAR dosyasını bir klasöre çıkarınız, ve sonrasında:

engerek.war dosyasını ***<tomcat>/webapps klasörüne kopyalayınız***.

Tomcat’i başlatınız.

Bu işlem sonrasında uygulama war’ınız otomatik olarak deploy edilecektir.

Yükleme Sonrası

http://localhost:8080/engerek/ Url’inden uygulamaya ulaşılır.
(Tomcat Server otomatik olarak 8080 portunu kullanmaktadır, eğer farklı bir ayar yapıldıysa ona göre bu port değerinin değişmesi gerektiğini unutmayınız.)
Sisteme ilk girişte aşağdaki kullanıcı bilgileri kullanılabilir.
Kullanıcı Adı:administrator
Şifre: 5ecr3t

Eğer bir problem var ise; <tomcat>/log/catalina.out ve <tomcat>/log/idm.log yollarında bulunan Tomcat’in loglarına bakınız.

Kaynak ve Bağlayıcı (Konnektör) Referansı

Bu bölümde Engerek uygulamasının, değişik uç sistemlere / kaynaklara nasıl bağlandığı anlatılmaktadır.

Bu bölümde, Engerek sistemi ile bağlantı yapılabilen ortam ve sistem türleri ele alınacaktır.

Kimlik Bağlayıcıları

Kimlik Bağlayıcıları Hakkında

Engerek uygulaması, Sun Microsystem tarafında geliştirilen “Identity Connector Framework (ICF)” temelli bağlayıcılarını kullanmaktadır. Bu yapı, KYS ile hedef ve kaynak sistemlerini ayıran bir katman sağlamaktadır ve Java, .NET tabanlı bağlayıcıları destekler.

Bağlayıcılar, Engerek ve entegre edilecek sistemler arasında iletişimi sağlayan katmandır. Entegre edilecek sistemler, Engerek sistemine kullanıcı verisini sağlayabilir veya Engerek sistemi, bu ortamlardaki kullanıcı hesaplarını yönetebilir veya kaynaklar arası veri senkronizasyonunu sağlayabilir. Her sistemin yapısı, iletişim protokolü farklı olacağından, Engerek sisteminde standard konfigürasyon ve operasyon için bağlayıcılar oldukça önem arzetmektedir.

Active Directory Bağlayıcısı

ADSI protokolü kullanılarak Active directory’ye bağlanılmasını ve hesap yönetimini sağlar. Sadece .NET bağlayıcı sunucusu üzerinde çalışmaktadır.

Protocol	**ADSI**
Versiyon	OpenICF 1.1.x

Adı	ActiveDirectory.Bağlayıcısı
Bağlayıcı adı	Org.IdentityConnectors.ActiveDirectory.ActiveDirectoryConnector
Ek gereksinimler	.NET Bağlayıcı sunucusu (connector server)

Yetenek ve özellikler

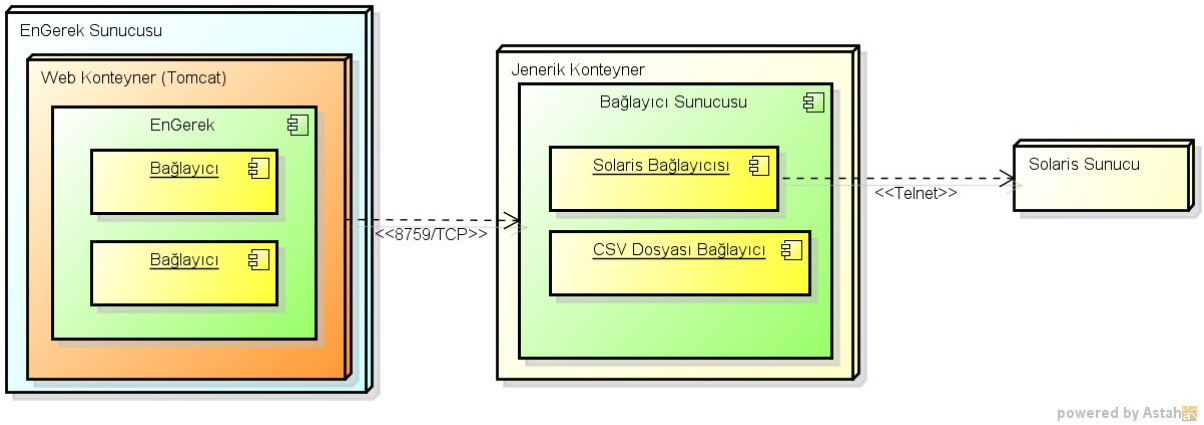
Provizyonlama	Var
Gerçek zamanlı senkronizasyon	Var
Şifre	Var
Aktivasyon	Var
Betik çalıştırma	Var

Bağlayıcı Sunucusu

Açıklama

Bağlayıcı sunucusu, bağlayıcıları barındıran bir konteynerdir. Hafif bir bileşen olup, Engerek'in çalıştırıldığı ortama da kurulabilir. Engerek, doğrudan bağlanamadığı sistemlere bu bağlayıcı sunucusunu (proxy, gateway gibi) kullanarak bağlanabilir.

Bağlayıcı sunucuları farklı şekillerde kullanılabilir. En sık kullanım nedenlerinden biri, Java ortamından doğrudan erişilemeyen sistemlerdir. Active Directory Bağlayıcısı buna en sık rastlanabilecek örnektir. Bağlayıcı sunucuları, aynı zamanda, ağ üzerinden doğrudan ulaşılamayan kaynaklara (ör: uç sunuculardaki dosyaların dışa aktarımları) bağlanmak için kullanılabilir. Bağlayıcı sunucusu, güvenlik amaçlı da kullanılabilir. Engerek ve ve bağlayıcı sunucusu arasındaki iletişimin güvenliği, ortak bir şifre ve/veya SSL üzerinden sağlanabilir. Böylelikle, hedef sistemin önüne kurulacak bir bağlayıcı sunucusu, doğrudan iletişimden daha güvenli olacaktır.



Bağlayıcı Sunucularının Kurulumu ve Konfigürasyonu:

İki tip ICF bağlayıcı sunucusu bulunmaktadır:

- Java Bağlayıcı Sunucusu bilinen Java tabanlı bağlayıcılar için kullanılır. Bağlayıcının, EnGerek'in çalıştırıldığı host makinanın dışında çalıştırılması gerektiği durumda tercih edilmektedir. CSV Dosyası için bağlayıcılar için kullanılabilir. Böylelikle, dosyanın EnGerek kurulu olduğu makineye kopyalanması ya da FTP ile gönderilmesine gerek kalmaz.
- .NET Bağlayıcı Sunucusu .NET tabanlı bağlayıcılar için kullanılır. Genellikle sadece Active Directory ya da benzer Microsoft tabanlı teknolojilerde tercih edilmektedir.

EnGerek Konfigürasyonu

Bağlayıcı sunucusu kurulduktan sonra, EnGerek içerisinde konfigürasyonu yapılmalıdır. EnGerek, yerel makinada bağlayıcıları otomatik olarak keşefedebilmektedir. Fakat uç sistemler için durum farklıdır. Öncelikle EnGerek, bir bağlayıcı sunucusunun varlığından haberdar olmalıdır. Ayrıca bağlantı ve güvenlik parametreleri de belirtilmeldir. Bağlayıcı sunucusu kullanacak bir bağlayıcı, *Connector Host* parametresini kullanarak yapılandırılmaktadır.

Bağlayıcı Host

Bağlayıcı Host nesnesi, bağlayıcı sunucusunun tanımlamalarını içerir. Bağlayıcı sunucusunun host adı, bağlantı için kullanılan port numarası, ortak şifre ve diğer erişim ve güvenlik bilgilerini kapsar. Tanımlarda, bir bağlayacıya özel tanımlama bulunmaz. Bağlayıcı ile ilgili tanımlamalar, her bağlayıcı için ayrı bağlayıcı nesnelerinde tanımlanır. Bunlar genellikle yarı otomatik olarak, uç bağlayıcı keşfi sürecinde oluşturulur. Bu bağlayıcı nesnesi, sonra tekrar Bağlayıcı Host nesnesine geridönüş yapar.

ConnectorHostType örneği

```
<connectorHost>
 <name> foobar.example.com:8759 üzerindeki ICF Bağlayıcı sunucusu</name>
 <hostname>foobar.example.com</hostname> <!—uç bağlayıcı sunucusunun kurulduğu
hostun adı/IPsi
 server installed -->
 <port>8759</port> <!-- uç bağlayıcı sunucusunun kurulduğu hostun portu-->
 <sharedSecret>
 <clearValue>secret</clearValue> <!—EnGerek ve uç bağlayıcı sunucusunun paylatığı
sifre-->
```

```
 </sharedSecret>
</connectorHost>
```

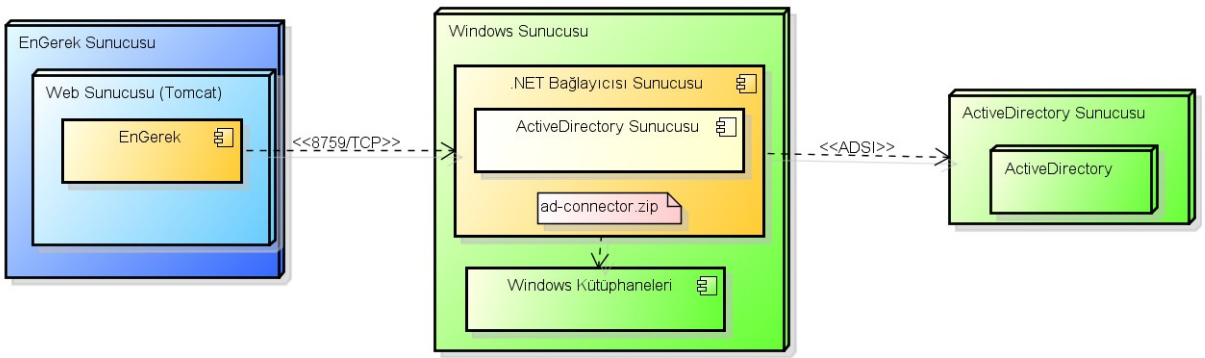
SSL Konfigürasyonu

EnGerek ve Bağlaycı sunucusu arasındaki iletişim, SSL ile güvenlik altına alınabilir. Bu durumda, Bağlayıcı sunucusu SSL sunucu, EnGerek ise SSL istemci olur. İki tarafta da gerekli SSL kurulumu yapılmalıdır.

	EnGerek	**Bağlayıcı Sunucusu**
Rol	SSL istemci	SSL Sunucusu
Kriptolanmış materyal	CA sertifikası	Anahtar çifti (özel anahtar ve sunucu sertifikası)
Amaç	Bağlaycısı sunucusu sertifkasyonun validasyonu	Bağlayıcı sunucusunun Engerek'e yetkilendirilmesi
Kriptolanmış materyalin kurulumu	Güvenli CA sertifkasının EnGerek anahtar deposuna eklenmesi	Bağlayıcı sunucusu SSL'inin anahtar çifti kullanılarak yapılandırılması (Java, .NET)
SSL'i etkinleştirme	ConnectorHostType içinde `connectorserver.ssl` opsiyonu `true` olarak değiştirilir.	Bağlayıcı sunucunun konfigürasyon dosyasında `connectorserver.ssl` opsiyonu `true` olarak değiştirilir.

Bağlayıcı sunucusu, herhangi bir tipte SSL sunucusu sertifikasına ihtiyaç duyar. Sertifikanın "public" olmasına gerek yoktur. Başka bir deyişle, sertifakının güvenilir sertifika sağlayıcı tarafında üretilmiş olması gerekmez. Özel sertifikalar kullanılabilir.

.NET Bağlayıcı Sunucusu



Bağlayıcı sunucusunun pratik anlamda kullanılma sebebi, Windows'a özel bağlayıcıların kullanılmasına olanak sağlamasıdır. Örneğin, ActiveDirectory

bağlaycısı Windows'a özel .NET kütüphaneleri kullanmaktadır ve Java ortamında doğrudan kullanılamazlar. Bu yüzden EnGerek, bu tip kütüphaneleri yerel bağlayıcılarda (ya da uç Java bağlayıcılarında) kullanamazlar.

Gereksinimler:

Windows 2003 / 2008

.NET Framework 3.5 veya ileri versiyonlar

Yaklaşık 20MB boş disk alanı

Yaklaşık 200MB kullanılabilir RAM

Kurulum:

`ServiceInstall-*.msi` dosyasını çalıştırın ve kurulum sihirbazını takip edin.
Bağlayıcı sunucusu Windows servisi olarak kurulacaktır.
Varsayılan kurulum alanı ***C:\Program Files\Identity Connectors\Connector Server***

Konfigürasyon:

Windows 'da "*Yerel Servisleri Görüntüle*" sayfasına gidin ve Bağlayıcı Sunucusunun başlatıldığını kontrol edin. (***Connector Server*** hizmetine bakın). Çalışıyorsa durdurun.

Komut satırı açın, bağlayıcı Sunucusunun anahtarını, ***C:\Program Files\Identity Connectors\Connector Server*** klasörüne gidip aşağıdaki komutu çalıştırarak kurun;

```
ConnectorServer /setkey <yeni_anahtar>
```

Buradaki yeni anahtar bağlayıcı sunucusun anahtarıdır.. Aynı anahtar Engerek'in deposundaki ConnectorHostType objesi içinde ayarlanmalıdır.
Bağlayıcı sunucusunun konfigürasyon dosyası ***ConnectorServer.exe.Config*** dir. Bu XML dosya, bağlayıcı sunucusu klasöründe bulunmaktadır. Genel anlamda en çok değişklik gerektiren değerler *appSettings* bölümündeki port, trace ve ssl ayarlarıdır.

SSL Konfigürasyonu

Bağlayıcı sunucusu bir SSL sunucusudur, bu yüzden bir anahtar çiftine ihtiyaç duyar. (özel anahtar ve sertifika) . NET Bağlayıcı sunucusu sertifika ve anahtarları Windows sertifikasında saklar. Konfigürasyon değişlikliği yapmadan önce sunucunun durdurulmuş olması gerekmektedir.

Anahtar çiftini bir sertifika deposuna aktarma

Anahtar çifti genellikle PKCS#12 formatında dağıtılır. (p12 ya da pfx uzantılı bir dosya). Bunun bir sertifika deposuna aktarılması gerekir. Bunun için ***certutil*** programı aşağıdaki gibi kullanılır:

```
certutil -v -p changeit -importPFX mykeycert.pfx
```

PKCS#12 dosyaları genellikle şifreyle korunur; bu yüzden şifre *-p* opsiyonu ile sağlanır. Eğer program hata verirse, mevcut bir sertifika var demektir. Bunun için *-f* opsiyonunu kullanarak mevcut sertifikanın üzerine tekrar yazdırabilirsiniz.

Bağlayıcı sunucusunun anahtar çiftini bulabilmesi, sertifikanın ayrı bir sertifika deposuna aktarılması gerekmektedir.

Bunun için yine ***certutil*** programını kullanarak aşağıdaki komut kullanılır:

```
certutil -addstore ConnectorServerSSLStore cert.pem
```

certutil programının kullanılabilmesi için, sertifikanın PEM ya da DER formatında olmalıdır.

Bağlayıcı sunucusunun kullandığı sertifika deposunun ismi *ConnectorServerSSLStore* dır.

Eğer ***ConnectorServerSSLStore*** oluşturulmamışsa, ***certutil*** programı ***-f*** opsiyonunun kullanılmasını isteyecektir.

Bağlayıcı Sunucusunda SSL ayarı yapılması

ConnectorServer.exe.Config dosyasında düzenleme yapmak gerekir. ***connectorserver.certificatestorename*** parametresi ***ConnectorServerSSLStore*** a çevirilir. Ayrıca ***connectorserver.usessl*** parametresi true yapılır. Dosya, aşağıdaki örnekteki gibi olmalıdır:

```
<configuration>
  ...
  <appSettings>
<add key="connectorserver.port" value="8759" />
<add key="connectorserver.usessl" value="true" />
<add key="connectorserver.certificatestorename" value="ConnectorServerSSLStore" />
<add key="connectorserver.ifaddress" value="0.0.0.0" />
<add key="logging.proxy" value="false" />
<add key="connectorserver.key" value="..." />
</appSettings>
...
</configuration>
```

Bu ayarı yaptıktan sonra sunucuyu başlatabilirsiniz. EnGerek tarafındaki ayarları yapmayı da unutmayınız.

Bağlayıcının kurulumu

1. Bağlayıcı sunucusu servisini durdurun.
2. Bağlaycının ZIP arşivini çıkartarak tüm bağlayıcı dosyalarını sunucu klasörüne kopyalayın. Varsayılan *klasör C:\Program Files\Identity Connectors\Connector Server*
3. Bağlayıcı sunucusunu servisini başlatın.

Log Kayıtları

Bağlayıcı sunucusu log kayıtları için standart .NET log mekanizmasını kullanır. Log konfigürasyonu *ConnectorServer.exe.Config* dosyası içerisinde *system.diagnostics* bölümünde yapılmaktadır.
Log konfigürasyonuna örnek bir XML bölümü aşağıda gösterilmiştir.

```
<system.diagnostics>
    <trace autoflush="true" indentsize="4">
      <listeners>
      <remove name="Default" />
      <add name="myListener" type="System.Diagnostics.TextWriterTraceListener"
initializeData="C:\Program Files (x86)\Identity Connectors\Connector
Server\connectorserver.log" traceOutputOptions="DateTime">
        <filter type="System.Diagnostics.EventTypeFilter" initializeData="All"/>
     </add>
     </listeners>
   </trace>
</system.diagnostics>
```

Yukarıdaki konfigürasyonda iki önemli parametre log dosya adı ve log seviyesidir. Bu iki ayar, ***initializedData*** parametresinde ayarlanır. Buradaki log seviyesi, ***"All"*** olarak en detay seviye olarak ayarlanmıştır. Seviye değerleri ***Error***, ***Warning***, ***Verbose*** ve ***All'dır.*** Bağlayıcı sunucusu, log lama ayarı yapıldıktan sonra tekrar başlatılmalıdır.

Notlar

.msi kurulumun için varsayılan klasör*: C:\Program Files\Identity Connectors\Connector Server*

64-bit sistemlerde, varsayılan klasör: *C:\Program Files (x86)\Identity Connectors\Connector Server*

Konfigürasyon dosyasının bulunduğu yer*: C:\Program Files\Identity Connectors\Connector Server\ConnectorServer.exe.Config*

Log dosyası oluşturulması için vasayılan klasör: *C:\ConnectorServer.log*

ConnectorServer 1.1.0.1 ve yukarısını kullanabilmek için .NET 4 framework gerekmektedir. Konfigürasyon dosyası aşağıdaki gibi güncellenmelidir, aksi halde bağlayıcı sunucusu başlatılamaz.

ConnectorServer.exe.config

```
<configuration>
  <runtime>
    <loadFromRemoteSources enabled="true"/>
  </runtime>
   ...
</configuration>
```

Hata ayıklarken loglamayı arttırmak içim konfigürasyonu aşağıdaki gibi değiştiriniz:

ConnectorServer.exe.config

```
<configuration>
...
  <filter type="System.Diagnostics.EventTypeFilter" initializeData="All"/>
...
```

Ayrıca Windows makinede 8759 nolu (ya da konfigürasyon dosyasında sizin belirlediğiniz port)

TCP portunun kullanılmasına izin veriniz.

Java Bağlayıcı Sunucusu

Java Bağlayıcı Sunucusu, EnGerek'in kullandığı bağlayıcılara benzer bağlayıcılar kullanmaktadır. *Java Uzak Bağlayacı Sunucusu* bağlanabileceği yerel kaynaklar olduğunda kullanılır. CSV Dosyası bağlayıcısı gibi yerel erişim gerektiren bağlayıcılar için tercih edilir. Bütünleşik bir şekilde yapılması zor ve riskli olan (dosya indirmenin yarıda kalması, hata idaresi vb.) FTP ya da benzer mekanizmalarla dosyaların kopyalanması operasyonu yerine daha güvenli bir yöntem sağlanmış olur. *Java Uzak Bağlayıcı Sunucusu*, aynı zamanda güvenlik duvarı geçişi veya güvensiz haberleşme protokollerinin güvenliğini sağlamak için de kullanılır.

Gereksinimler:

- Java SE 6 ya da sonrası

Bağlayıcı Sunucusunu, *git.pardus.org.tr/engerek* adresinden indirebilirsiniz.

SSL Yapılandırması

Bağlayıcı sunucusu bir SSL sunucusu olarak çalışacaktır. Bu nedenle, SSL sertifikasına ihtiyaç duyar. Java bağlaycısı sunucusu, gerekli anahtarı sertifika deposundan alır. Bunun için, standart Java JCE anahtar deposu kullanır. Anahtar deposu ilk kurulum aşamasında gelmeyebilir. Bu deponun oluşturulması ve bir anahtar çiftiyle doldurulması gerekir.

Anahtar Deposu Oluşturma ve Sertifika Ekleme

Sertifika genellikle PKCS#12 formatında (p12 ya da pfx uzantılı bir dosya) dağıtılır. Bu sertifikanın Java JCE anahtar deposunda kullanılabilmesi için dönüşüm yapılması gerekir. Bunun için Java platformunun içine bulunan ***keytool*** programı aşağıdaki gibi kullanılır:

PKCS#12 anahtarını dönüştürme ve java anahtar deposunda sertifika oluşturma

```
keytool -importkeystore -srckeystore mykeycert.p12 -srcstoretype pkcs12 -destkeystore
keystore.jks -deststoretype JKS
```

Yukarıdaki komut *keystore.jks* dosyasını oluşturur. Bu dosya aslında Java JCE anahtar deposudur. *keytool* komutu iki şifre sorar:

- Özel anahtar bulundurduğu için PCKS#12 dosyaları üzerindeki şifre
- Yeni oluşturulan anahtar deposu için gerekli şifre. ***Bu şifreyi unutmamalısınız!***

Burada dikkat edilmesi gereken bir husus vardır. Java JCE anahtar deposu tamamen şifre ile korunmaktadır. Aynı zamanda her tekil anahtar da bir şifreyle korunmaktadır. Genellikle bu şifreler aynıdır ve bağlayıcı sunucusunun da beklentisi budur. Öte yandan anahtar deposu PCKS#12 den dönüşümü yapıldığında, anahatar deposusunun şifresi son verilen şifredir, fakat anahtar şifresi PKCS#12 dosyasındaki gibi aynı kalır. Eğer şifre eşit değilse anahtar şifrelerinin değiştirilmesi için fazladan bir adım daha gereklidir:

Anahtar şifresini değiştirmek için:

```
keytool -keystore keystore.jks -storepass changeit -keypasswd -alias mykey
```

Anahtar deposu parametrelerinin Bağlayıcı Sunucusuna aktarılması

Yukarıda da belirtildiği üzere bağlayıcı sunucusu bir Java uygulaması olup ön tanımlı anahtar deposunu kullanır. Anahtar deposunun yeri, tipi ve şifresi bağlayıcı sunucusuna Java opsiyonları olarak geçirilir:

```
java ... -Djavax.net.ssl.keyStore=keystore.jks
-Djavax.net.ssl.keyStorePassword=changeit -Djavax.net.ssl.keyStoreType=JKS ...
```

Bu seçenekler bağlayıcı sunucusunu başlatan betiğe eklenir.

Bağlayıcı Sunucusunda SSL'i aktif hale getirme

connectorserver.usessl seçeneğinin değeri *connectorserver.properties* yapılandırma dosyasında "*true*" yapılır. Engerek tarafındaki konfigürasyonda da benzer şekilde ayarlanmalıdır. Bu işlemlerden sonra sunucu başlatılmalıdır.

CSV Dosya Bağlayıcısı

Açıklama:

Bu bağlayıcı, CSV formatındaki dosyalar için kullanılan genel geçer bağlayıcıdır. CSV formatlı dosyalardan içe dışa aktarma yapılan ortamlarda kullanılır. İnsan kaynakları ve benzeri sistemlerde tüm personel verilerinin dışa aktarımında kullanılabilir. Engerek kurulumlarında, genelllikle personel gibi kullanıcı verilerinin, Engerek sistemine aktarmak amacıyla kullanılmaktadır. Bu bağlayıcı, Engerek'in kurulduğu sunucu dışında bir ortama da kurulabilir, yani uzak bağlayıcı olarak çalıştırılabilir.

Protokol	YOK. Lokal CSV okuma / yazma
Versiyon	OpenICF 1.1.x
Ek gereksinimler	

Yetenek ve Özellikleri

Provizyonlama	Evet
Canlı Senkronizasyon	Evet
Şifre	Evet
Aktivasyon	Evet
Betik çalıştırma	Evet

Temel Veri Modeli

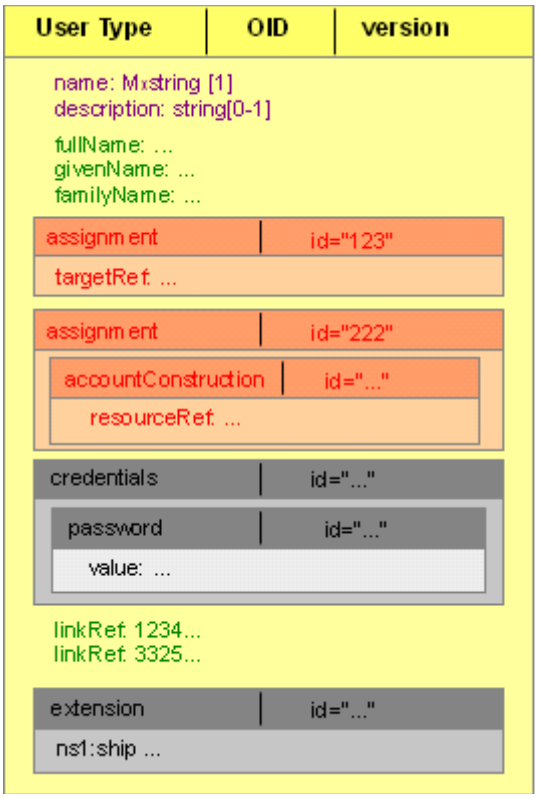
Engerek Veri Deposu

Engerek veri deposu, kullanıcılar, roller, kaynaklar, iş akışları, şifre politikaları ve diğer Engerek konfigürasyonları ile ilgili verileri tutar. Engerek veri deposu, *Engerek'in kendi veritabanıdır*, diğer sistemlerle, doğrudan paylaşılmaz. Veri senkronizasyonunun ne şekilde yapılacağı, eşlemelerin ve politikaların ne şekilde uygulanacağı, nelerin senkronize edileceği bilgisi Engerek veri deposunda tutulur. Bu bölümde, Engerek veri deposuna dair kavramlar ve veri deposunun yapısı anlatılmaktadır.

Hatırlatma: Bu bölümde verilen veri modeli bilgisi basitleştirilmiştir. Tam anlamı ile bir açıklama için XSD şema dosyalarına bakılmalıdır.

Nesneler

Engerek veri deposu nesne yönelimlidir. Depoda tutulan her bir bileşen, nesne biçimindedir. Aşağıdaki şekil, bir nesnenin 2 ayrık formdaki konumlanışını göstermektedir.




```
<user oid="abc123..." version="7">

 <name>foo</name>
 <description>Foo has home</description>

 <fullName>Foo Bar</fullName>
 <givenName>Foo</givenName>
 <familyName>Bar</familyName>
 <assignment id="123">
    <targetRef oid="..." type="c:RoleType"/>
 </assignment>

 <assignment id="222">
    <accountConstruction>
        <resourceRef oid="..." type="..."/>
    </accountConstruction>
 </assignment>

 <credentials>
    <password>
        <value>...</value>
    </password>
 </credentials>

 <linkRef oid="1234..."/>
 <linkRef oid="3325..."/>

 <extension>
    <ns1:ship>Black Pearl</ns1:ship>
 </extension>
</user>
```

Her nesne, kendine özgü bir OID nesne tanımlayıcısı ile tanımlanır. Bu OID, herhangi bir şekilde değiştirilemez. Bu özellik, nesneye, adı değiştirildiğinde ya da taşındığında, yok olmama özelliği kazandırır. OID'ler, nesneye, Engerek veritabanında oluşturuldukları anda atanır. Engerek sistemi, kullanıcıların OID'leri görmesine ihtiyaç duymayacak şekilde tasarlanmıştır. OID'ler gizlidir ve sistem yöneticilerinin dahi, OID ile herhangi bir operasyonu yoktur.

Nesneler, değerleri tutan *özelliklere* sahiptirler. Her bir *özellik* bir isimle temsil edilir. Yukarıdaki şekilde görülen nesnenin özellikleri, "name", "fullname", "familyName"dir.

- Özelliklerin değerleri, genel olarak temel (primitive) veri tipleri ile doldurulur (string, number vs).
- Bunların yanısıra daha karmaşık değerler de, örneğin zaman damgaları (tarih + zaman), isim ve url ler de temel veri biçimleri olarak düşünülebilir.
- Özellikler, karmaşık veri biçiminde de olabilir (XML veri biçimi).
- Özellik değerleri, tek ya da çoklu olabilir. Çoklu özellikler, "**sırasız**" olarak tasarlanmıştır, Engerek tarafından, yazım sıralamasının korunacağı garanti altına alınmamıştır.

Nesneler, başka nesnelere referanslar taşıyabilir. Taşınan referanslar, diğer nesnelerin OID'leri kullanılarak oluşturulur. Yukarıdaki şekildeki nesne, ***linkRef***ler sayesinde diğer iki nesneye referans taşımaktadır.

Özellikler ve referanslar ***"konteyner"*** adı verilen gruplarda tutulur. Yukarıdaki şekildeki nesnenin tuttuğu "konteyner"ler "assignment" ve "credentials" dir. Her konteyner bir "*id"* ile temsil edilir.

- *"id"* nesnenin tanımlandığı alanda tekildir.
- *"id"*, Engerek tarafından otomatik olarak atanır.
- Hiyerarşik bir veri yapısı oluşturmak üzere, konteynerler başka konteynerler de içerebiliir. Örneğin, yukarıdaki nesnenin "credentials" konteyneri, "password" konteynerini altında barındırmaktadır.

OID ve nesne tipinin yanısıra, "version" da bir nesne özelliği olarak tanımlanmış olmalıdır.

- Tekil olmak zorunda değildir.
- Otomatik olarak atanır.
- Görevi, gelişmiş şekilde cache leme ve kilitlemedir.

Yukarıdaki şekilde, "User Type" görülmektedir. Kullanıcı tipi, "assignment" tiplerini içermektedir. Bizim örneğimizde, ilk "assignment" başka bir nesneye referans olan "targetRef" özelliğini içermektedir. Bu durumda, nesne, "Role" tipindeki bir nesneye referans olmaktadır. İkinci atama ise, "accountConstruction" gömülü konteynerini içermektedir, resourceRef özelliğini taşımaktadır. Bu nitelik, bir kaynak tipi nesneye referans olmaktadır.

Aşağıda, fazladan bir "activation" konteyneri içeren "User Type" nesnesine dair örnek bulunmaktadır.

```
<user oid="8c048b23-...-001e8c717e5b">
<name>jack</name>
<fullName>Jack Sparrow</fullName>
<givenName>Jack</givenName>
<familyName>Sparrow</familyName>
<honorificPrefix>Cpt.</honorificPrefix>
<emailAddress>jack@blackpearl.com</emailAddress>
<employeeType>intern</employeeType>
```

```
<locality>Tortuga</locality>
<activation>
<enabled>true</enabled>
</activation>
<credentials>
<password>
<value>... encrypted data ...</value>
</password>
</credentials>
<linkRef oid="f792ad4e-..."/>
<linkRef oid="148f22be-..."/>
</user>
```

Nesne Tipleri

Temel nesne tipleri aşağıdaki tabloda özetlenmiştir.

Nesne Tipi	XML Şema İsmi	Amacı
Kullanıcı	UserType	Sistemdeki “gerçek, fiziksel” kullanıcıyı temsil eder.
Gölge	ShadowType	Bir hesap, grup, rol, yetki vb nesnenin, provizyonlama yapılan kaynaktaki yerel kopyasıdır.
Kaynak	ResourceType	Yönetilecek olan, Engerek dışında bir sistemi tanımlayan bir nesnedir.
Bağlayıcı	ConnectorType	Jenerik bir konektörün tanımıdır. Engerek sisteminde, yönetilen kaynağa bağlantı kuran her metoda konektör denir.
Rol	RoleType	Role Based Access Control (RBAC) ı uygulayan bir roldür.
Nesne Şablonu	UserTemplateType	Yeni kullanıcı tanımlamak

		ve varolan kullanıcıları düzenlemek üzere kullanılan şablondur. Kullanıcı nesneleri için bir “politika” formundadır.
Organizasyon	OrgType	Organizasyon birimi, takım, grup, bölümü gibi hiyerarşik veriyi tutar. Bu tipin nesne tipleri hiyerarşik yapıyı tutmak için karmaşık bir yapıya sahip olur.
Sistem Konfigürasyonu	SystemConfigurationType	Global sistem konfigürasyonunu tutan bir nesnedir. Genel olarak tek bir kopya olarak bulunur.

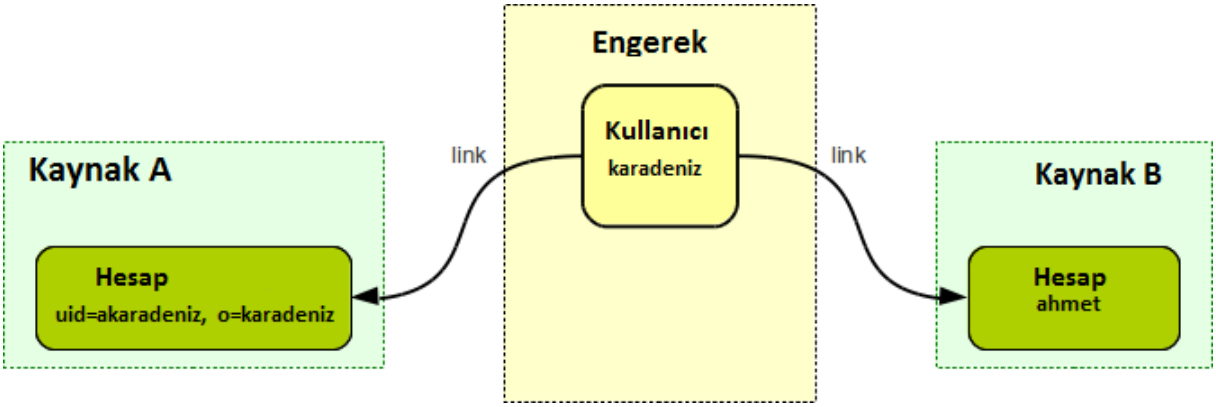
Kullanıcılar ve Hesaplar

Kullanıcılar ve hesaplar, kimlik yönetimi sistemlerinde iki temel kavramdır. Kullanıcı ve hesap arasında çok belirgin farklar vardır.

	Tanım	**Bulunduğu Yer**	**İçeriği**
Kullanıcı	Engerek veri deposunda bulunan kullanıcı, gerçek bir kişiyi temsil eder.	Engerek	Gerçek kişinin, çalışan, partner vb. Olarak özelliklerini tutar. Bu özellikler, adı, soyadı, ünvanı vb. özelliklerdir. Bu data kümesi çok geniştir.
Hesap	Bilgi sistemine erişim için gerekli erişim bilgilerini tutar. Hesap, genel olarak bir kaynağa ya da bir grup kaynağa uygulanır. Bazı durumlarda, örneğin dizin sunucuda tutulduğunda, değişik kaynaklar	Kaynak	Bir bilgi sistemine erişim için gerekli olan asgari erişim veri kümesidir. Bu küme genelde, kullanıcı adı, şifre, grup üyelikleri, yetkilerden oluşur.

	tarafından paylaşılabilir.		

Provizyonlama sisteminin en önemli görevlerinden biri, kullanıcı ile, sahip olduğu hesapların ilişkilendirilmesidir. Engerek, hesaplarla bu hesaplara sahip olan kullanıcı arasındaki ilişkiyi oluşturan bir link kurar. Bu linkler, neredeyse tüm Engerek fonksiyonlarında kullanılır. Örnek olarak, kullanıcı nesnesi üzerinde yapılacak değişiklikler, hesaplara da yansıtılır, ya da bir kullanıcı silindiğinde, hesapları da silinir.



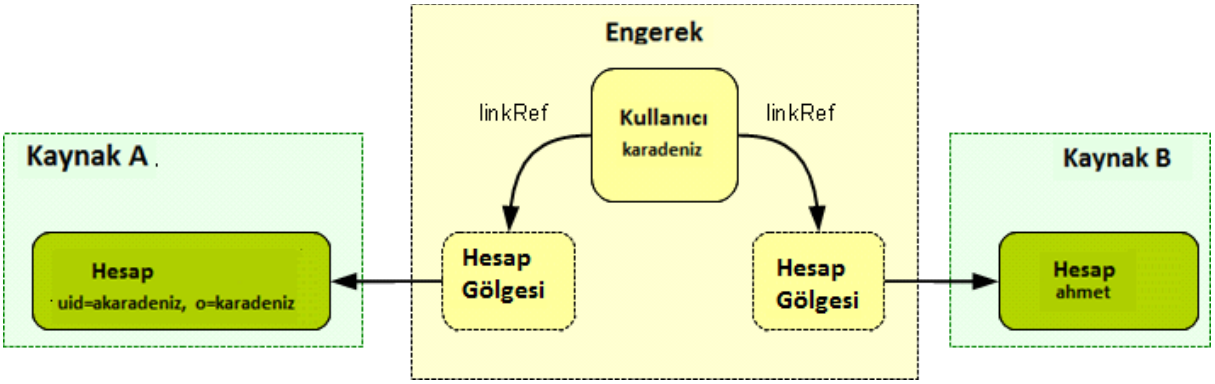
Hesaplar birçok şekil ve biçimde bulunabilir. Her hesap değişik özellikler, değişik özellik tipleri ve kısıtlar içerebilir. Örneğin kaynaklardan biri, kullanıcının tam adını tek bir string özelliği içinde *Unicode* biçiminde kullanılırken, diğer bir kaynakta, adı ve soyadı farklı *string* alanlarda, özel karakterler içermeden tutulmak zorunda olabilir. Bu farklılıkları çözümlemek ve birbirleri ile entegre etmek Engerek'in görevidir.

Hesap Gölgesi

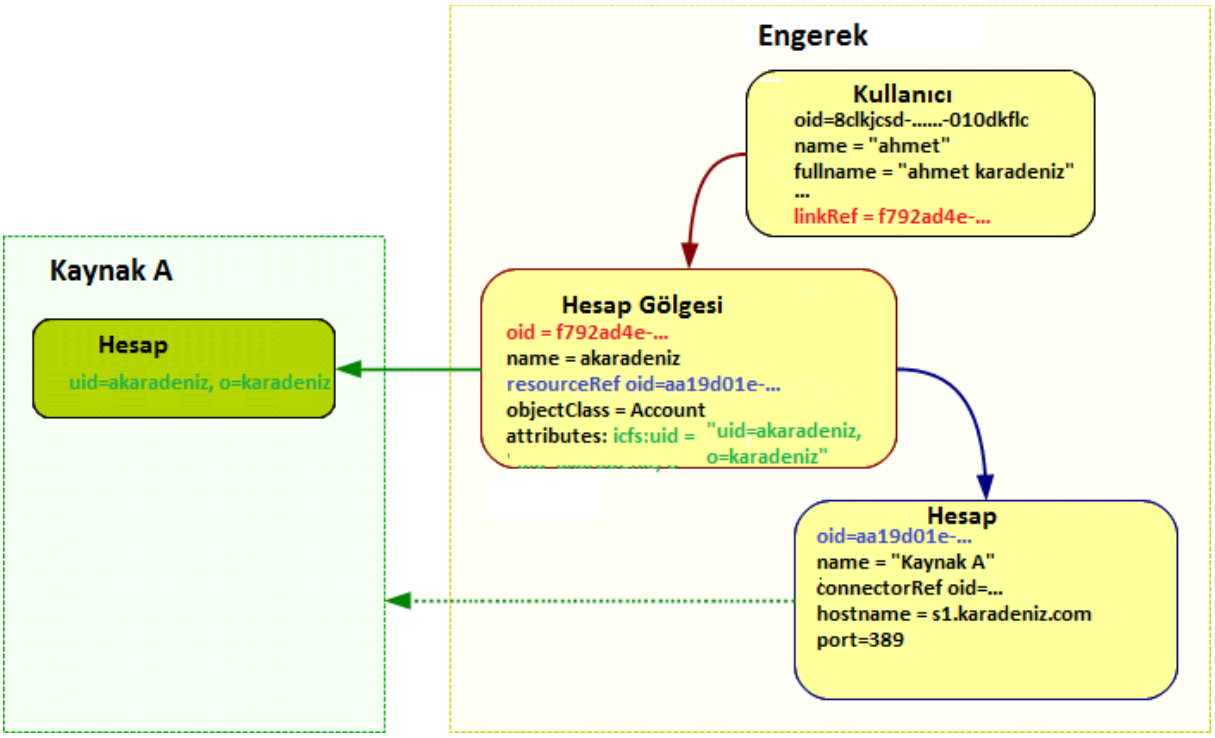
Diğer hesap özellikleri gibi, hesap tanımlayıcıları da sistemden sisteme değişiklik gösterebilir. Bazı hesaplar, bir hesabı tanımlamak üzere birden fazla değerin birleşmesi ile oluşan birleşik tanımlayıcılara dahi ihtiyaç duyabilir. Tanımlayıcılar, hesap-kullanıcı ilişkisinin kurulması için, Engerek tarafından ihtiyaç duyulan

bileşenlerdir. Hesaplar kaynakta tutulur, Engerek veri deposunda tutulmaz. Bu sebeple, kullanıcı (Engerek nesnesi) ile hesap (Engerek nesnesi değil) arasında link kurmak çok zordur.

Engerek, bu sorunu kolaylaştırmak için "*nesne gölgeleri*"ni, ya da kısaca "*gölgeleri*" kullanır. Gölgeler, Engerek veri deposunda, hesap karakteristiklerini temsil eden sıradan nesnelerdir. Gölge'de tutulan en önemli bilgi parçası, gölgenin temsil ettiği hesabın hesap tanımlayıcısıdır.



Tanımlayıcılar, gölge'de, çok esnek bir veri yapısı kullanılarak tutulurlar. Her kaynak için, tanımlayıcının tipi, biçimi ve numarası farklı olabilir. Gölgeler sıradan nesneler olduğu için, kullanıcı nesnesi, OID kullanarak gölge'lere kolaylıkla işaret edebilir. Aşağıdaki şekil, gölge nesneleri kullanılarak kullanıcı – hesap ilişkilendirmesinin ne şekilde yapıldığını göstermektedir.



Gölgelerin kullanılmasının bir çok faydası vardır. Öncelikle, hesap adları değiştiğinde, kullanıcı – hesap ilişkilendirmesini kolaylaştırmaktadır. Kaynak tarafında bir hesap silindiğinde bundan haberdar olamayabiliriz. Ancak gölge kullanımı ile, gölge'nin ilişkilendiği bir hesabın eksik olması durumu kolayca tespit edilebilir ve bu duruma uygun olan tepki Engerek tarafından verilir.

Gölge nesneleri sadece kullanıcı – hesap linklerini kurmak için kullanılmaz. Engerek'in birçok fonksiyonunda kullanılır. Bir hesabı temsil eden çok zengin veri nesneleri olarak da kullanılabilir. Aşağıdaki şekil, böyle bir gölgeyi anlatmaktadır.

Tam bir gölgenin şekli yukarıdaki gibi olsa dahi, bu nesnenin sadece bir bölümü veri deposunda tutulur. Aktivasyon verisi, credentials verileri, kaynaktan çekilir ve veri deposundaki veri ile birleştirilir.

Kaynak

Kaynak tipi, Engerek veri deposunda tutulur ve kaynağı tanımlar. Oldukça karmaşık bir nesnedir, zira tuttuğu özellikler fazladır. Kaynak tipi nesnesinin genelde içerdiği veriler:

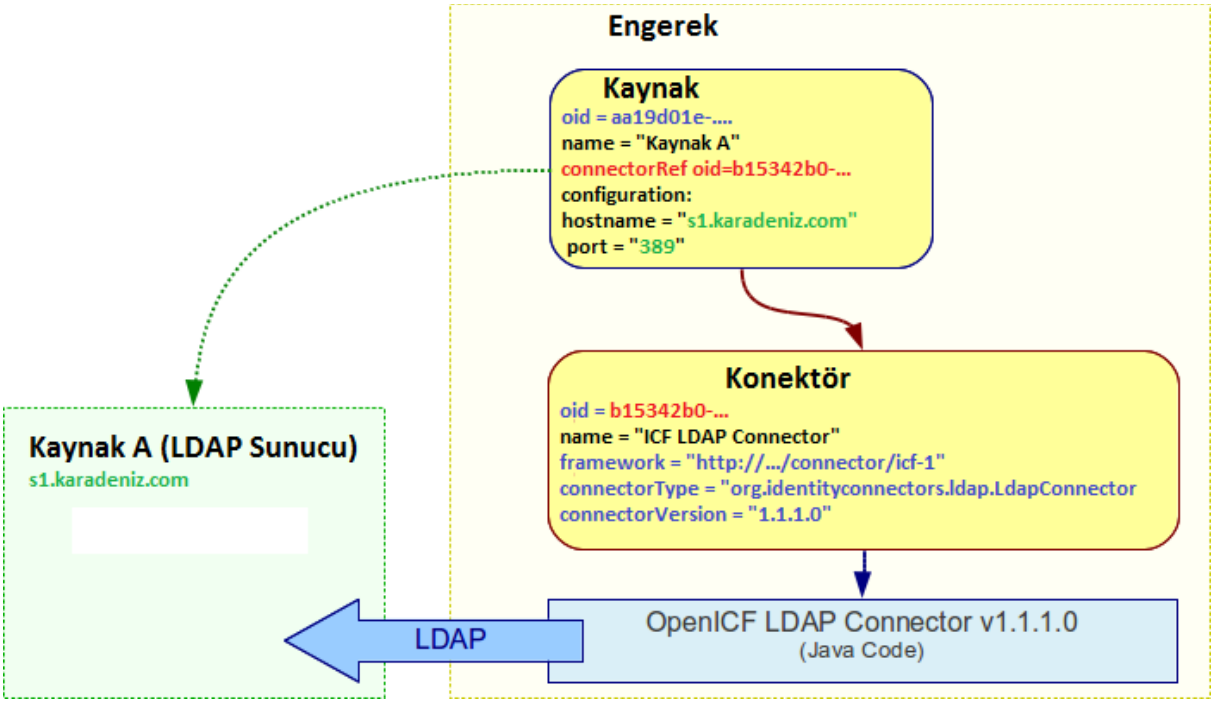
- Kaynağa erişmek için kullanılan bağlayıcıya (konektör) olan referans,
- Bağlayıcının eriştiği kaynağa dair port, hostname, yönetici hesabı ve şifresi, iletişim protokolü (SSH, telnet vbg) gibi bağlayıcı erişim ayarları,
- Kaynağın desteklediği nesne tiplerini (hesaplar, yetkiler vbg) tanımlayan kaynak şeması ve bu kaynak nesnelerinin hangi özelliklere sahip olduğu verisi,

- Kullanıcı özellikleri ile nesne özellikleri arasında eşlemeyi ve dönüştürmeyi yapmaya yönelik konfigürasyon verisi,
- Kaynağın senkronizasyon açısından güvenilir kaynak olup olmadığı bilgisi, hesabın kaynakta silinmesi durumunda ya da kaynakta yeni hesap oluşturulması durumunda hangi tepkilerin verileceğine dair senkronizasyon ayar verileri

olarak sıralanabilir. Kaynak tipi ya da kaynak tanımı, Engerek yönetimi ve konfigürasyonu açısından en önemli nesnedir. Engerek'in işlevlerinin büyük çoğunluğu, kaynak tanımının konfigürasyonu ile yapılır.

Bağlayıcı (Konektör)

Bağlayıcı, kaynağa erişim için kullanılan kod yapısıdır. Kaynak ile Engerek arasındaki iletişimi sağlamak, kaynağa özgü protokolü, Engerek tarafından anlaşılabilecek bir veri formuna dönüştürmekle görevli bir sürücü olarak düşünülebilir. Bağlayıcılar, Engerek uygulaması ile birlikte deploy edilen java jar dosyalarıdır. Bunun yanında, bağlayıcıya ilişkin nesne, jar dosyasından bağımsız olarak Engerek veri deposunda tutulur. Bu nesne ***ConnectorType*** olarak adlandırılmıştır. ConnectorType nesnesi, bağlayıcıyı tanımlar ve bağlayıcı için bir OID tanımlar.



Odak ve Yansıtma (Projeksiyon)

- Senkronizasyon
- Odak ve İzdüşüm

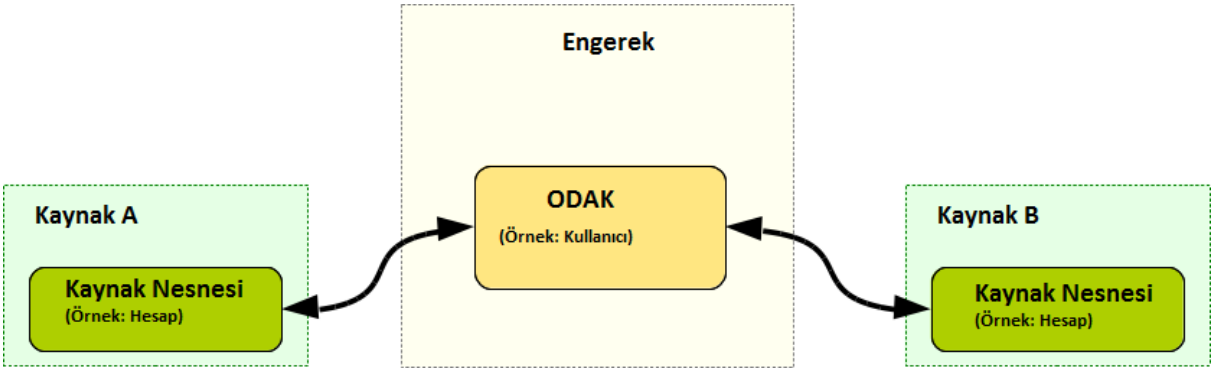
Senkronizasyon

Engerek'in birincil işlevlerinden biri, kaynaklarda tutulan nesneler arasında veri senkronizasyonunu yapmasıdır. Engerek, senkronizasyonu karmaşık hale getiren "herşeyi herşeyle senkronize etme" yönteminden uzak olarak tasarlanmıştır. Örneğin, Engerek, bir kaynaktaki hesabı, başka bir kaynaktaki hesapla doğrudan senkronize etmez. Engerek, bir hesabı, kendi üzerindeki kullanıcı nesnesi ile senkronize ettikten sonra, kullanıcı nesnesini, diğer hesap ile senkronize eder. Engerek her zaman ortadaki bileşendir ve "odak" işlevi görür.

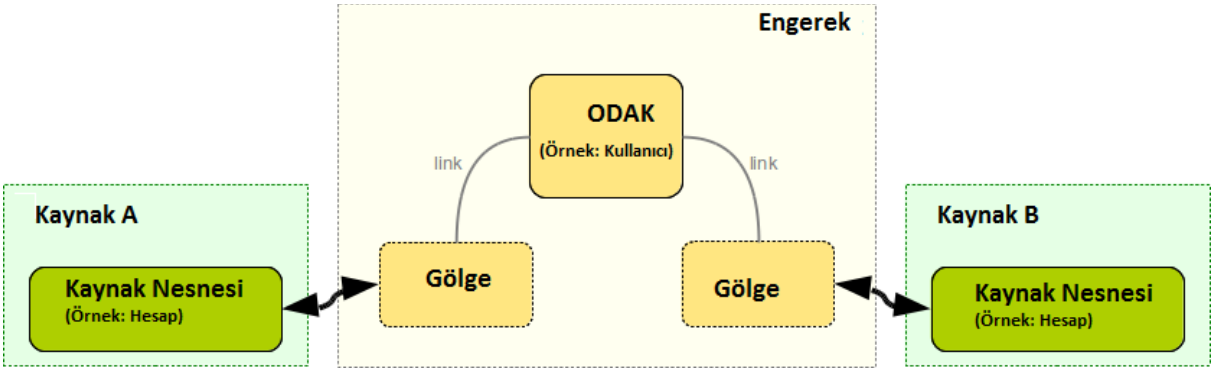
Odak ve İzdüşüm

Senkronizasyonun odağında Engerek vardır. Her bir veri parçası, "odak nesnesi"ne gider ve kaynaklara geri yansıtılır. "Kullanıcı" nesnesi tipik bir "odak nesnesi"dir.

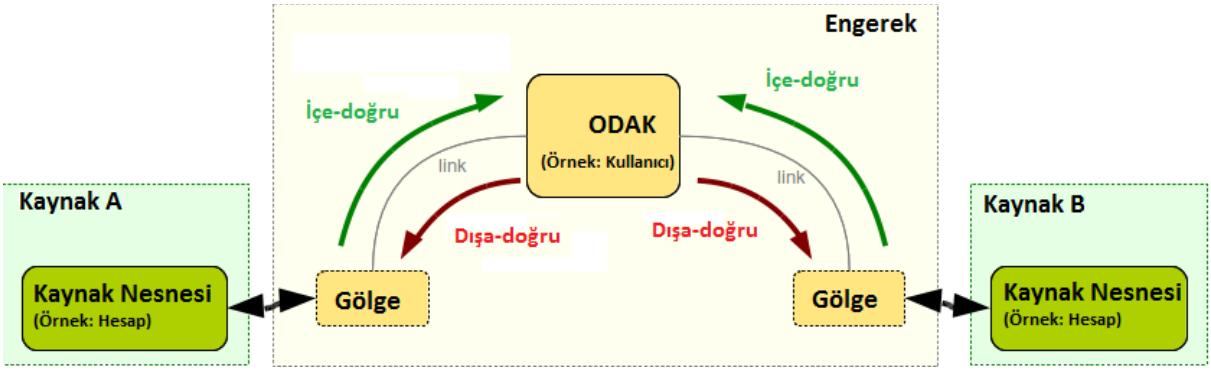
"Odak ve izdüşüm" yaklaşımını, akışlar üzerinden somutlamak gerekirse, aşağıdaki şekiller faydalı olacaktır.



Yukarıdaki şekil, Engerek-kaynak veri akışını şematik bir yaklaşımdır ve aslında akış daha karmaşıktır. İzdüşümler, kaynak nesneleri olduğu için, doğrudan onlarla çalışmak uygun olmayacaktır. Bu sebeple, tüm kaynak nesnelerinde olduğu gibi, *gölge*'lerle çalışmak gerekecektir.



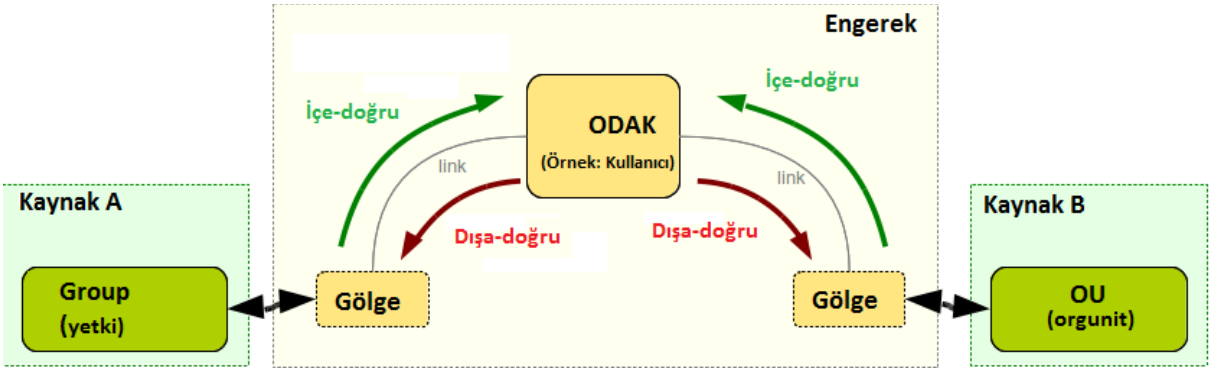
İzdüşüm gölge'leri, Engerek odakta olmak üzere, kaynaklarla senkronizasyon mekanizması ile senkronize edilir. Senkronizasyon yöntemi *içe-doğru* (inbound) eşlemelerle, *dışa-doğru* (outbound) eşlemelerden oluşur.



Odak nesnesi olarak

- Kullanıcı
- Rol
- Organizasyon

nesneleri kullanılabilir. Aşağıdaki şekilde, Organizasyon nesnesinin bir odak nesnesi olarak kullanıldığı durum temsil edilmiştir.



Kanallar

Engerek'te *kanal* kavramı, veri geçişi için bir yöntemi niteler. Engerek, veriyi kullanıcı – hesap ve hesap – kullanıcı arasında her dağıtışında, "kanallar"ı kullanır. Kanallar, değişikliğin *ne zaman* tespit edildiğini, *neden* bu değişikliğin yapıldığını tanımlar.

Kanallar, Engerek sisteminin optimum performans ile çalışması için kullanılabilir. Örneğin, kanallar sayesinde, belli durumlarda, belli eşlemelerin çalıştırılmasına kısıtlar getirilebilir. Bu durum için tipik bir örnek, Engerek'i içe-doğru eşlemelerin, sadece içe-aktarımlarda kullanılması için kısıtlamaktır. Bu işlem, eşlemeyi sadece "içe-aktarım kanalı"ndan yapmak, böylelikle, eşleme işlemini sadece hesaplar Engerek veri deposuna aktarılırken kullanmak, senkronizasyon sırasında kullanmamak yönünde yapılabilir.

Kanallar, url formatında tanımlanır. Engerek'te, senkronizasyon ve içe-aktarım gibi iç işlemlerde kullanılmak üzere, belli ön tanımlı url ler vardır. Ancak Engerek'e farklı durumlar için, yönetici tarafından yeni kanallar tanımlamak da mümkündür.

Ön tanımlı gelen kanalar aşağıdaki tabloda görülmektedir.

Kanal Adı	Tanım
Synchronization: liveSync	Canlı senkronizasyon tarafından tespit edilen değişikliklerde kullanılan kanaldır.
Synchronization: reconciliation	Reconciliation sırasında tespit edilen değişikliklerde kullanılan kanaldır.
Synchronization: discovery	Discovery sırasında tespit edilen değişikliklerde kullanılan kanaldır.
Synchronization: import	Engerek'te kullanıcı oluşturmak üzere hesapların içe aktarımı sırasında kullanılan kanaldır.
GUI: user action	Yönetim Arayüzünde, kullanıcının hareketleri için kullanılan kanaldır.
Web Service	Web servis çağrılarında kullanılan kanaldır.

Bildirimlerin Konfigürasyonu

Aktarım Konfigürasyonları

Şu an için, Engerek iki bildirim yöntemi içermektedir. Bunlar e-posta ve SMS'tir. E-posta ve SMS için konfigürasyon aşağıdaki xml dosyasından incelenebilir.

```
<mail>
<server>
<host>smtp.gmail.com</host>
<port>587</port> <!-- port: 25, 587 (MS Exchange, TLS vb.) veya diğer sunucular -->
<!-- port değerlerini silerek ön tanımlı değerlerin kullanımı sağlanabilir -->
<username>abc@gmail.com</username>
<password>
<clearValue>abcdef</clearValue>
</password>
<transportSecurity>ssl</transportSecurity> <!-- seçenekler: none, starttlsEnabled,
starttlsRequired -->
</server> <!—birden fazla sunucu belirtilebilir, yazılan sıra ile kullanılırlar -->
<defaultFrom>abc@gmail.com</defaultFrom>
<debug>true</debug> <!-- stardard javax.mail hata ayıklama dosyasıdır(idm.log) -->
</mail>
<sms name="default"> <!-- birden fazla SMS konfigürasyonu olabilir, farklı isim ile
belirtilirler -->
```

```
<gateway> <!—birden fazla gateway tanımı olabilir, birisi başarısız olursa diğeri
ile devam edilecektir -->
<url> <!—şu anda HTTP(S), GET methodu kullanılabilmektedir. Diğer mekanizmalar
geliştirilme aşamasındadır -->
<script>
<code>"https://my-sms-gateway.com/send?number=" + to + "&amp;text=" +
encodedMessageText</code>
</script>
</url>
</gateway>
</sms>
<sms name="test">
<redirectToFile>sms-notifications.log</redirectToFile> <!-- bu tanım yapıldığında
log kayıtları gateway yerine dosyaya gönderilir -->
</sms>
```

generalNotifier

generalNotifier, gelen her olayı kabul edip her türlü bildirimi oluşturabilir. Bu oluşturma işleminin tamamı, aşağıdaki tablodaki parametre ve ifadeler ile yapılır.

Parametre Adı	Tipi	Alacağı Değer	Anlamı
name (attribute)	String	0..1	Okumayı kolaylaştırmak için, bildiricinin adını tutar
description	String	0..1	Bilgilendirici tanımı
Herhangi bir handler adı	EventHandlerType	0..N	Bu bilgilendiriciye gelen olayları filtreler
recipientExpression	ExpressionType	0..1	Bilgilendirmeleri alacak olan alıcılar
subjectExpression	ExpressionType	0..1	Gönderilecek olan mesajın konusu ne şekilde oluşturulacak.
subjectPrefix	String	0..1	subjectExpression'a basit bir alternatiftir. Konu, bilgilendirici tarafından oluşturulur, ancak burada tanımlı kısaltma ile temsil edilir.
bodyExpression	ExpressionType	0..1	Mesaj textinin ne şekilde oluşturulacağını tarif eder.
watchAuxiliaryAttributes	Boolean	0..1	Yardımcı özellikler değiştiğinde (validityStatus, validityChangeTimestamp, effectiveStatus, disableTimestamp, modifyChannel, modifyTimestamp, modifierRef) bildirim gönderilip gönderilmeyeceğinin kararını verir. Varsayılan değeri : false (0) dır.
showModifiedValues	Boolean	0..1	Değiştirilen değerlerin gösterilip gösterilmeyeceğini

			tutar.
showTechnicalInformation	Boolean	0..1	Değişiklik sonrası teknik tanımın gösterilip gösterilmeyeceği bilgisini tutar.
transport	String	0..N	Hangi aktarım yöntemlerinin kullanılacağı bilgisini tutar.

simpleUserNotifier

Kullanıcı nesnesindeki bir değişiklik sonrası bildirim üretir. generalNotifier'ın parametreler dışında kendine özgü ek bir parametresi yoktur.

simpleAccountNotifier

Kullanıcı hesabında bir değişiklik olduğunda bildirim gönderir. Sadece bir parametresi vardır.

Parametre Adı	Tipi	Alacağı Değer	Anlamı
watchSynchronizationAttributes	Boolean	0..1	Senkronizasyonla ilgili yardımcı özellikler değiştiğinde (synchronizationSituationDescription, synchronizationSituation) bildirim gönderilip gönderilmeyeceğinin kararını verir. Varsayılan değeri : false (0) dır.

userPasswordNotifier, accountPasswordNotifier

Kullanıcı / hesap şifresi oluşturulduğunda ya da değiştirildiğinde bildirim gönderir. Özel bir parametresi yoktur.

workflowNotifier

Bir işlem ya da işlem kopyası tamamlandığında ya da başladığında bildirim gönderir. Özel bir parametresi yoktur.

Filtre Konfigürasyonu

categoryFilter

Belirtilen kategorilerdeki hareketleri geçirilmesi için kullanılır. Tanımlı kategoriler aşağıda listelenmiştir;

Kategori Adı	İfade	Tanım
userEvent	event.isUserRelated()	Veri deposunda kullanıcıya bağlı hareket.
accountEvent	event.isAccountRelated()	Kaynaktaki hesaba bağlı hareket
workItemEvent	event.isWorkItemRelated()	Bir bileşenin çalışmasının başlaması/bitmesi.
workflowProcessEvent	event.isWorkflowProcessRelated()	Bir iş akışı işleminin başlaması/bitmesi
workflowEvent	event.isWorkflowRelated()	workItemEvent ya da workflowProcessEvent

Örnek :

```
<categoryFilter>
<category>userEvent</category>
</categoryFilter>
```

statusFilter

Aşağıdaki durum tanımları ile tanımlanabilen hareketleri filtreler.

Adı	İfade	Kullanıcı olayları için Anlamı	Hesap Olayları için Anlamı	İş Akışı Olayları için Anlamı****
Success	event.isSuccess()	Tüm değişiklikler başarılı*.	Operasyon başarılı.	Talep onaylandı.
alsoSuccess	event.isAlsoSuccess()	En az bir değişiklik başarılı*.	Operasyon başarılı.	Talep onaylandı.
onlyFailure	event.isOnlyFailure()	Tüm değişiklikler başarısız**.	Operasyon başarısız.	Talep reddedildi.
failure	event.isFailure()	En az bir değişiklik başarısız**.	Operasyon başarısız.	Talep reddedildi.
inProgress	event.isInProgress()	En az bir değişiklik işlemde***.	Operasyon işlemde.	Talep sonucu bilinmiyor.

(*) Sonuç, SUCCESS, WARNING, ya da NOT_APPLICABLE.

(**) Sonuç, FATAL_ERROR, PARTIAL_ERROR ya da NOT_APPLICABLE.

(***) Sonuç, IN_PROGRESS.

operationFilter

Çalıştırılan ya da çalıştırılmaya teşebbüş edilmiş işlemleri temel alan olayları filtreler.

Adı	İfade	Kullanıcı ya da Hesap Olayları için Anlamı	İş Akışı Olayları için Anlamı
add	event.isAdd()	Kullanıcı/hesap oluşturuldu.	İşlem ya da iş bileşeni başlatıldı
modify	event.isModify()	Kullanıcı/hesap değiştirildi	-
delete	event.isDelete()	Kullanıcı/hesap silindi.	İşlem ya da iş bileşeni bitirildi

expressionFilter

ifade temelli olayları filtreler. Aşağıda bazı örnekler görülmektedir.

```
<expressionFilter> <!-- "security-admin" kullanıcısı tarafından yapılan 'yeni
hesap' taleplerini filtreler -->
<expression>
<script><code>event.isAccountRelated() &amp;&amp; event.isAdd() &amp;&amp;
"security-admin".equals(requester?.getName()?.getOrig())</code></script>
</expression>
</expressionFilter>
```

İfadeler

Engerek tarafından desteklenen tüm ifadeler kullanılabilir. Değişkenler aşağıdaki tablodaki gibidir.

Adı	Tipi	Tanımı
event	Event	İşlenmiş olan olay.
Requester	UserType	Operasyonu talep eden kullanıcı
Requestee	ObjectType	Operasyon sonucunda değişen nesne (genellikle kullanıcı) ya da operasyon sonucunda değişen hesabın sahibi
Assignee	UserType	İş bileşenine atanmış kullanıcı
transportName	String	Aktarım Adı.

Şemanın Özelleştirilmesi

Engerek, bir çok kurumda ortak olarak kullanılacağı varsayılan nitelik ve özelliklere sahip, zengin bir kullanıcı şeması sunar. Fakat bazı zamanlarda bu özellikler yeterli olmayabilir. Bu durumda şemanın, yeni tanımlanacak özelliklerle genişletilmesi gerekir. Bu işlem, uygun XSD dosyasının Engerek paketine eklenmesi ile yapılır.

Örnek :

Aşağıdaki örnek, *a:extension* XSD formatında tanımlanmış olan UserType şemasını, *officeNumber* ve *favoriteColor* özellikleri ile genişletmeyi göstermektedir.

```
<xsd:schema elementFormDefault="qualified"
targetNamespace="http://example.com/xml/ns/mySchema"
xmlns:tns="http://example.com/xml/ns/mySchema"
xmlns:a="http://prism.evolveum.com/xml/ns/public/annotation-2"
xmlns:c="http://midpoint.evolveum.com/xml/ns/public/common/common-2a"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:complexType name="UserExtensionType">
<xsd:annotation>
<xsd:appinfo>
<a:extension ref="c:UserType"/>
</xsd:appinfo>
</xsd:annotation>
<xsd:sequence>
<xsd:element name="officeNumber" type="xsd:string" minOccurs="0"
maxOccurs="1">
<xsd:annotation>
<xsd:appinfo>
<a:indexed>true</a:indexed>
<a:displayName>office number</a:displayName>
<a:displayOrder>120</a:displayOrder>
</xsd:appinfo>
</xsd:annotation>
</xsd:element>
<xsd:element name="favoriteColor" type="xsd:string" minOccurs="0"
maxOccurs="unbounded">
<xsd:annotation>
<xsd:appinfo>
<a:indexed>true</a:indexed>
<a:displayName>favorite color</a:displayName>
<a:displayOrder>130</a:displayOrder>
<a:help>The favorite color</a:help>
</xsd:appinfo>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Yukarıdaki dosyanın yaptığı yegane iş, şemayı iki özelleştirilmiş nitelik ile genişletmektir.

Adı	Görünür Adı	Tipi
officeNumber	*Office number*	*String*
favoriteColors	*Favorite Color*	*string*

Engerek, *kod* alanı sayesinde özelliği tanır. Eşleştirmelerde ve konfigürasyonda bu alandan yararlanır. *Görünür ad* ise formlarda ve raporlarda, Engerek'in gösterdiği *ad* dır. Özellik Tipi alanı, özellik değerinin bulunabileceği aralık ve veri tipini belirtir.

Özelleştirilmiş şema tamamlandıktan ve Engerek uygulaması yeniden paketlendikten sonra, bu özellikler Grafik arayüzde gösterilmeye başlanacaktır.