MANUAL D'INTEGRACIÓ ROLSAC

DESEMBRE DE 2014

ROLSAC

FULL DE CONTROL DE VERSIONS

DOCUMENT/ARXIU

Títol: **MANUAL D'INTEGRACIÓ API ROLSAC V2**	Nombre Arxiu/s**: ROLSAC-1.2-API_V2_MAN_INTG.doc**
Codi: **ROLSAC-1.2-API_V2_MAN_INTG**	Suport lògic**: Word**
Data: **15-12-2014**	Ubicació física:
Versió**: 1**	

REGISTRE DE CANVIS

Versió	Pàgines	Mou del canvi
1		Creació del document.

DISTRIBUCIÓ DEL DOCUMENT

Nom	Personal
Estela Pisano	DGTIC
Salvador Gelabert	DGTIC
Bartomeu Cerdà	Brújula
Bartomeu Bennassar	Brújula
Antoni Garcia	AT4

CONTROL DEL DOCUMENT

PREPARAT	REVISAT/ APROVAT	ACEPTAT	ACEPTADO
Bartomeu Cerdà	Salvador Gelabert		

Emplenar amb el nom, la firma i la data.

Només per clients

1 Introducció

ROLSAC, a partir de la branca 1.2, ofereix una API completa de consulta de tota la informació pública disponible.

Aquest document cobreix les instruccions bàsiques d'integració, en diversos escenaris:

- Per a integradors que emprin JAVA com a plataforma de desenvolupament.
- Per a integradors que emprin qualsevol altre tecnologia de client.

Per a una descripció exhaustiva dels serveis disponibles, paràmetres d'entrada i dades de retorn de cada un dels mètodes, es poden consultar els *apidocs* adjunts.

2 Disseny general

API de consulta

- A nivell del servidor, és una API composada per una jerarquia de serveis (EJB i serveis web SOAP) que recuperen DTOs. N'hi ha un per cada entitat (rolsac, procediment, formulari,..). Cal fer notar que les consultes d'entitats no retornen mai dades de relacions associades a aquesta entitat. Per exemple, si es volen consultar els tràmits d'un procediment, cal cridar explícitament al mètode *llistarTramits* del servei de procediments.
- A nivell de client Java, la API també està dividida en una jerarquia d'objectes, un per entitat. Des del punt de vista del desenvolupador del client JAVA, l'ús de EJBs o serveis web és transparent, i es limita a la configuració. L'API de client empra el patró "adapters" per encapsular la informació de les entitats juntament amb els mètodes d'obtenció d'informació relacionada.

3 Ús de l'API de ROLSAC per a clients JAVA

Els desenvolupadors JAVA, poden emprar el client java del API.

3.1 Dependències per l'API EJB.

Si l'aplicació client es troba desplegada en el mateix servidor d'aplicacions que l'aplicació ROLSAC, aleshores és possible emprar l'API EJB, que ofereix major rendiment en vers a l'API WS.

Les següents dependències han d'estar disponibles en el classloader client:

- API client de ROLSAC (sac-api-client.jar)
- SpringFramework 3.0.5.RELEASE
- Apache commons beanutils 1.8.3
- Apache commons lang 2.1
- Apache commons logging 1.0.4

3.2 Dependències per l'API WS.

Si es vol emprar la versió de serveis web SOAP, les següents dependències han d'estar disponibles en el classloader client:

- API client de ROLSAC (sac-api-client.jar)
- SpringFramework 3.0.5.RELEASE
- Apache commons beanutils 1.8.3
- Apache commons lang 2.1
- Apache commons logging 1.0.4
- Commons discovery 0.2
- Axis 1.4: axis.jar, jaxrpc.jar, saaj.jar, wsdl4j-1.5.1.jar

3.3 Configuració prèvia

El primer que s'ha de fer és crear una classe que ens faciliti la configuració, el més senzill és crear una classe APIUtils. A continuació es proporciona un exemple del codi Java que hauria de contenir aquesta classe.

Amb aquesta classe per una banda establim l'autentiticació contra l'API i per l'altre configuram l'estratègia de connexió (EJB/WS) mitjançant la constant API_STRATEGY. Els valors de configuració són STRATEGY.EJB i STRATEGY.WS.

Codi APIUtils.java

```
import es.caib.rolsac.api.v2.estadistica.EstadisticaInsertService;
import es.caib.rolsac.api.v2.general.BeanUtils;
import es.caib.rolsac.api.v2.general.BeanUtils.STRATEGY;
import es.caib.rolsac.api.v2.general.CertificadoUtil;
import es.caib.rolsac.api.v2.rolsac.RolsacQueryService;

public class APIUtil {

   public static final STRATEGY API_STRATEGY = STRATEGY.WS;

    public static enum Sexo {
        MASCULINO,
        FEMENINO;

        public static Sexo getSexo(Integer sexo) {
            return sexo == 2 ? FEMENINO : MASCULINO;
        }
    };

    private APIUtil() {}

    public static RolsacQueryService getRolsacQueryService() {

       String keyStoreName = System.getProperty("property.name");
       String keyStorePass = System.getProperty("property.pass");

       if (API_STRATEGY == STRATEGY.WS)
          CertificadoUtil.autentificar(keyStorePass, keyStoreName);

       return (RolsacQueryService) BeanUtils.getAdapter("rolsac", APIUtil.API_STRATEGY);

    }

    public static EstadisticaInsertService getEstadisticaInsertService() {

        return (EstadisticaInsertService) BeanUtils.getAdapter("estadistica",
APIUtil.API_STRATEGY);

   }

}
```

Si l'estratègia emprada és la del WS, aleshores és necessari utilitzar un KeyStore que guarda el certificat pel tema de seguretat. Dins el projecte ROLSAC es proporciona un KeyStore per començar a desenvolupar.

Aquest KeyStore el podeu trobar a la següent ruta del projecte:

/test/api/integracion/resources/storerolsac.jks

La forma correcta d'utilitzar el KeyStore és mitjançant el mètode estàtic "autentificar" que proporciona ROLSAC dintre la classe es.caib.rolsac.api.v2.general.CertificadoUtil. El mètode rep dos paràmetres, el primer és el nom del keystore i el segon la seva clau.

En l'exemple de codi, els valors d'aquests dos paràmetres es recuperen a partir de les propietats del sistema:

- property.name=storerolsac.jks
- property.pass=contrasena

Es recomana que s'utilitzin propietats de sistema en comptes de "hardcodear" el nom i la contrasenya.

3.4 Criteria i ordenació

L'API proporciona un conjunt de serveis per recuperar informació, els quals poden ser parametritzats a partir de filtres anomenats Criteria. Així per exemple, si s'utilitza un mètode *llistarSeccions* del servei *rolsac* podem filtrar o restringir la cerca per les diferents propietats del SeccioCriteria.

Exemple:

```
SeccioCriteria seccionCriteria = new SeccioCriteria();
seccionCriteria.setIdioma("ca");
seccionCriteria.setCodigoEstandard("seu");
```

L'API també permet recuperar la informació de forma ordenada mitjançant el mètode ".setOrdenar(...)" dels diferents Criterias.

El mètode rep com a paràmetre un array d'ordenacions predefinides per cada Criteria.

Així, cada Criteria ens ofereix un conjunt de camps disponibles pels quals podem ordenar els resultats. La prioritat de l'ordenació és la mateixa que l'ordre definir en l'array.

Exemple:

```
IniciacioOrdenacio[] ordenacions = new IniciacioOrdenacio[2];
ordenacions[0] = IniciacioOrdenacio.codigoEstandar_asc;
ordenacions[1] = IniciacioOrdenacio.id_desc;
iniciacioCriteria.setOrdenar(ordenacions);
```

3.5 Consulta

Per realitzar la consulta sols és necessari cridar al mètode desitjat amb el Criteria adient. Tot seguit hi ha un exemple d'un client fent ús de l'API.

Exemple complet:

```
import java.util.List;

import es.caib.rolsac.api.v2.exception.QueryServiceException;
import es.caib.rolsac.api.v2.normativa.NormativaCriteria;
import es.caib.rolsac.api.v2.normativa.NormativaQueryServiceAdapter;
import es.caib.rolsac.api.v2.procediment.ProcedimentCriteria;
import es.caib.rolsac.api.v2.procediment.ProcedimentQueryServiceAdapter;
import es.caib.rolsac.api.v2.rolsac.RolsacQueryService;

public class RolsacClient {

  /**
   * @param args
   * @throws QueryServiceException
   */
  public static void main(String[] args) throws QueryServiceException {

      String idUa = "3";

      RolsacQueryService rqs = APIUtil.getRolsacQueryService();
      ProcedimentCriteria procedimentCriteria = new ProcedimentCriteria();
      procedimentCriteria.setIdioma("es");
      procedimentCriteria.setUnidadAdministrativa(idUa);
      int numProcs = rqs.getNumProcediments(procedimentCriteria);

      System.out.println("Num de procediments per la UA " + idUa + ":" +
numProcs);

      //Per lo general els llistats necessiten que indiquem el tamany de
paginació
      procedimentCriteria.setTamany("20");
      //Per poder oredenar el llistat de procediments en funció de id
      procedimentCriteria.setOrdenar(new ProcedimentOrdenacio[] {
ProcedimentOrdenacio.id_desc });
```

```
      List<ProcedimentQueryServiceAdapter> procediments =
rqs.llistarProcediments( procedimentCriteria );

      for (ProcedimentQueryServiceAdapter procedimentDTO : procediments) {
            System.out.println(procedimentDTO.getNombre());

            //Les relacions fan una nova crida al API
            List<NormativaQueryServiceAdapter> normatives =
procedimentDTO.llistarNormatives(new NormativaCriteria());
            for (NormativaQueryServiceAdapter normativa : normatives) {
                  System.out.println("normativa:" + normativa.getTitulo());
            }
      }
  }
}
```

3.6 Tipus de retorn

En les seccions anteriors hem vist com configurar i emparar l'API mitjançant el client que proporciona ROLSAC (sac-api-client.jar). El tipus de retorn d'aquest client són els Adapters que encapsulen les dades de retorn i mètodes d'accés a informació relacionada.

No obstant això, un programador podria escriure el seu propi client a partir dels WSDL, en aquest cas, el tipus de retorn serien els DTOs proporcionats al WSDL com es veu a l'apartat següent de clients no Java.

4 Ús del API de ROLSAC per a clients no java

Els clients no java han d'emprar els serveis web SOAP per accedir al API. Els arxius *wsdl* del conjunt de serveis web disponibles es poden consultar als fonts del projecte, a la carpeta moduls/api/etc/wsdl. Els arxius clau són:

RolsacWS.wsdl.- Servei web general que ofereix accés de recerca a les diferents entitats.

<entitat>WS.wsdl.- Servei web de cada <entitat> que ofereix accés al contingut relacionat amb l'entitat.

4.1 Codi d'exemple php5.

```php
<?php

//Url base dels serveis web
$serverURL = 'http://caibrolsac.in.at4.net:8080/sacws-api/services/';

//Servei web general
$clientRolsac = new SoapClient($serverURL . 'RolsacWS?wsdl', array("trace" => 1,
"exception" => 0));

//Servei web de procediment
$clientProcediment = new SoapClient($serverURL . 'ProcedimentWS?wsdl',
array("trace" => 1, "exception" => 0));

$idUA = 3;

//Desde PHP, es poden enviar els objectes com vectors associatius
echo "Num de procediments per la UA $idUA:";
echo $clientRolsac->getNumProcediments(
    array(
        "idioma"=>"es",
        "unidadAdministrativa"=>$idUA
        )
    );
echo '<br/>';

//O emprar la forma elegant, replicant els objectes.
//Certs frameworks poden crear el conjunt de les classes a partir dels WSDL
//s'ha de crear la classe ProcedimentCriteria perquè aquest exemple funcioni.
require "./ProcedimentCriteria.class.php";
```

```php
$criteria = new ProcedimentCriteria();
$criteria->idioma = "es";
$criteria->unidadAdministrativa = $idUA;

echo "Num de procediments per la UA $idUA:";
echo $clientRolsac->getNumProcediments( $criteria );
echo '<br/>';

//Per lo general els llistats necessiten que indiquem el tamany de paginació
$criteria->tamany = "20";

echo "<pre>";
$procediments = $clientRolsac->llistarProcediments( $criteria );
$procedimentDTO =$procediments[0];
var_dump( $procedimentDTO );
echo "</pre>";

echo '<br>';

//Per llistar les relacions, ja hem d'emprar el webservice corresponent a cada
tipus d'entitat
//Llistar les normatives d'un procediment
var_dump($clientProcediment->llistarNormatives($procedimentDTO->id, array(
"idioma"=>"es") ));


?>
```

5 Serveis disponibles

Actualment l'API disposa de 34 serveis. Tots els serveis s'utilitzen de la mateixa forma. Entre ells n'hi ha dos (RolsacWS i EstadisticaWS) que per les seves característiques convé destacar.

RolsacWS és un servei general que permet recuperar informació de diferents entitats. Mitjançant aquest servei podem atacar a l'API i accedir de manera fàcil a la resta de serveis relacionats.

L'altre servei que cal mencionar és el de les estadístiques, EstadisticaWS. Cal destacar que és l'únic WS de l'API que pot fer operacions d'escriptura. Aquest WS ens permet anar registrant les estadístiques d'accés sobre les entitats de consulta (procediments, fitxes informatives, unitats administratives, etc).

A continuació es llisten els serveis disponibles:

- AgrupacioFetVitalWS
- AgrupacioMateriaWS
- ButlletiWS
- CatalegDocumentsWS
- DocumentTramitWS
- DocumentWS
- EdificiWS
- EnllacWS
- EspaiTerritorialWS
- EstadisticaWS
- ExcepcioDocumentalWS
- FamiliaWS
- FetVitalWS

- FitxaUAWS
- FitxaWS
- FormulariWS
- IconaFamiliaWS
- IconaMateriaWS
- IdiomaWS
- MateriaAgrupacioWS
- MateriaWS
- NormativaWS
- PerfilWS
- PersonalWS
- ProcedimentWS
- PublicObjectiuWS
- RolsacWS
- SeccioWS
- TaxaWS
- TipusWS
- TramitWS
- UnitatAdministrativaWS
- UnitatMateriaWS
- UsuariWS