Report on the exportability of the Spanish e-APP system

Ministry of Justice of Spain

May 2011

TABLE OF CONTENTS

1. INTRODUCTION AND OBJECTIVES

The e-APP system (hereinafter, e-APP) of the Ministry of Justice of Spain has been developed for use by Competent Authorities under the jurisdiction of the Ministry as well as by other agencies that are competent to issue Apostilles in Spain. It has therefore been designed in a modular way, using the "dependency injection" ("connectors") and "service orientation" paradigms that allow certain components to be replaced with others in order to fit the needs and technological requirements of a given agency that may implement the system. It also offers an open and highly interoperable architecture.

The purpose of this document is to report on the potential to export the e-APP system developed by the Ministry of Justice of Spain to Competent Authorities in other jurisdictions both nationally and internationally.

2. DESCRIPTION OF THE E-APP SYSTEM

2.1. General description of the system

- The Spanish e-APP system is a J2EE web application architecture that resides on centralised servers, and which end users access via an Internet browser. The system consists of two subsystems:
  - **e-Apostille processing subsystem** . This subsystem is for internal use by users in charge of handling Apostilles. Users access this subsystem through a simple Internet browser. This subsystem will be used by users in Competent Authorities involved in the procedure of processing and issuing e-Apostilles
  - **Internet publication subsystem**. This subsystem allows the following actions over the Internet and through an Internet browser:
    - Downloading an e-Apostille issued (accessible for applicants, authentication required)
    - Consulting the e-Register of Apostilles (for authorities in other Contracting States / applicants, no authentication required)
- The system relies on services that provide core functions, such as the Digital Signature Platform or the Secure Validation Code Generation service, and in turn exposes services to be integrated with other applications through the so-called "integration layer " (SOA architecture)
- The products and technologies used in the solution implemented by the Ministry of Justice of Spain are:
  - Web applications developed in Java running on WAS 6.1
  - Database management system (RDBMS): Oracle 10g
  - Digital signature platform: ASF from TB Solutions
  - Standard document format: Adobe PDF
  - Digital signature standards:
    - PAdES for signing e-apostilles
    - XAdES, CAdES & PAdES are accepted as standards for digitally signing public documents
- Technological requirements for the points from which the Apostille system will be used are:
  - **e-Apostille processing subsystem** : This subsystem will be used by users in Competent Authorities involved in the procedure of processing and issuing e-Apostilles. In these cases, software requirements of client stations are:
    - OS: Windows 2000, XP, Vista or 7
    - Internet browsers: Microsoft Internet Explorer 7 or 8
    - PDF viewer: Adobe Reader X
    - Software for cryptographic cards reading devices: X509 v3

- **Internet publication subsystem**.

This subsystem will be used by applicants for the download of e-Apostilles or by foreign authorities in other Contracting States for their verification online. In those cases, software requirements of client stations are:

  - OS: Windows 2000, XP, Vista, 7 or Linux
  - Internet browsers: Microsoft Internet Explorer 7 or 8, Mozilla Firefox 2.0 or higher, Google Chrome 10
  - PDF viewer: Adobe Reader X

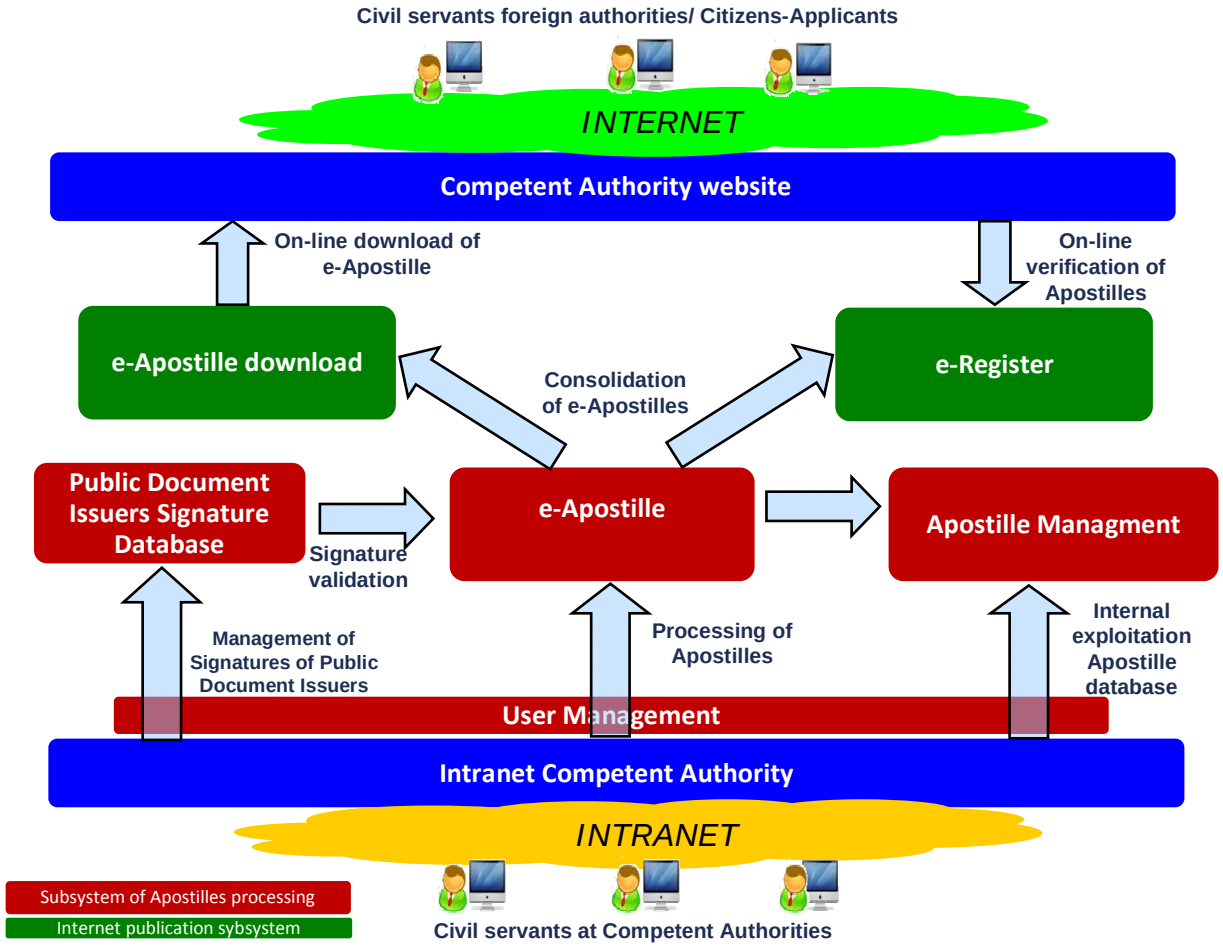
2.2. Description of functions

The two subsystems of the e-APP system (Subsystem of processing e-Apostilles and Internet publication subsystem) may be broken down into several modules on the basis of their functions:

- **e-Apostille processing subsystem** :
  - Module for issuing e-Apostilles (e-Apostille)
    - Generation and digital signature of the Apostille
  - Public Document Issuers Signatures Database Module
    - Creating, updating or deleting digitalised holographic signatures and/or digital signatures of officials and authorities whose signatures are to be recognised and verified by the Competent Authority in order to issue the Apostille.
  - Apostille Management Module
    - Allows for internal consultations and recordkeeping in relation to applications for Apostilles.
  - User Management Module
    - Creating, updating or deleting user accounts and allocation of user profiles within the various modules of the Subsystem of processing e-Apostilles.

- **Internet publication subsystem**:
  - e-Register of Apostilles Module
    - Electronic register where all Apostilles issued in both paper and electronic format are recorded and classified. Accessible via the Internet.
  - e-Apostille Download Module
    - Allows the applicant to download the e-Apostille via the Internet.

The abovementioned modules and how they interact can be represented by the following diagram:



The e-APP system also has an integration layer in the form of Web Services, which provides access to e-APP services from other systems. Within the integration layer two Web Services can be identified:

- Web service for the integration with e-Apostille (creating flows for processing Apostilles): Intended for integration with external applications generating electronic public documents.
- Web service for integration with e-Register: Allows Apostilles issued to be included into the e-Register. It also allows the e-Register module to function independently of the e-Apostille module.

2.3. Functional description

This section describes the basic functions covered by the e-APP system.

e-Apostille processing subsystem

e-Apostille Module

Procedure for issuing an Apostille for a paper public document

- An applicant submits a paper public document.
- A user at the Competent Authority (with permission to "register applications") creates a new request for an Apostille and fills in the required fields.
- Alternatively, the document can be digitalised (scanned) using a scanner. In this case, the scanned image of the paper public document will be added to the e-Apostille as an attachment.
- The user registers the application. The application goes then to the stage of "DP signature validation". A receipt confirming the application is produced, which the user at the Competent Authority prints out and hands to the applicant.
- A user with permission to "validate Apostilles" (the user may be the same as the user who registered the application) validates the holographic signature and / or seal on the public document. To this end, the public document can be viewed within the application (if scanned) as well as images of its signature and seal. If everything is correct, the user marks the request as "valid". The status of the request becomes “Validated”.
- A user with permission to "sign Apostilles" (the user may be the same as the user who validated the application for Apostille) issues and signs the Apostille. This document is a PDF file of the trilingual model Apostille certificate developed by the Hague Conference (Spanish - English - French). If the public document was scanned, the PDF file will contain an image of the scanned public document as an embedded attachment. The Apostille will be signed electronically with PAdES signature. Once signed, the Apostille application proceeds to “Issued” status.
- A user (with permission to "notify applications") retrieves the Apostille and notifies the applicant through one of the following chosen means:
    - In person: the user contacts (by e-mail or telephone) the applicant to notify him/her that the Apostille may be collected from the Competent Authority. The Apostille may be delivered in electronic format using a digital storing device such as a USB flash drive or similar) or in paper format. Once the applicant has collected the Apostille, the user marks the application in the system as “Notified”.
    - By mail: this only applies if the Apostille is to be delivered in paper form, physically attached to a paper public document. Once mailed, the user marks the application in the system as “Notified”.
    - By downloading the e-Apostille online: this only applies if explicitly requested by the applicant. In this case the applicant only needs to press the "Notify" button and the system will immediately make the e-Apostille available in the e-Apostille Download Module of the Internet publication subsystem.

Procedure for issuing an e-Apostille for an electronic public document

- An applicant submits an electronic public document on a digital storing device such as a USB flash drive or similar.
- A user at the Competent Authority (with permission to "register applications") creates a new request for an e-Apostille and fills in the required fields, uploading the electronic public document file.
- The user registers the application. The application goes then to the stage of "public document signature validation". A receipt of application is produced from the application, which the user at the Competent Authority prints out and hands to the applicant
- The system automatically and in an unattended way, validates the electronic signature on the public document and, if it is correct, looks for the matching certificate in the Public Document Issuers Signatures Database. If a public document issuer is found with a valid signature, the system will validate the application. If no valid signature is found in the Public Document Issuers Signatures Database, the status of the application for an Apostille is "Rejected".
- A user with permission to "sign Apostilles" issues and signs the e-Apostille. This document is a PDF file of the trilingual model Apostille certificate developed by the Hague Conference (English - Spanish - French), which will have the electronic public document file embedded as an attachment. The e-Apostille will be signed electronically with PAdES signature. Once signed, the Apostille application proceeds to "Issued" status.
- A user (with permission to "notify applications") retrieves the Apostille and notifies the applicant through on of the following chosen means:
    - In person: the user contacts (by e-mail or telephone) the applicant to notify him/her that the Apostille may be collected from the Competent Authority. The e-Apostille may only be delivered in electronic format (to a digital storing device such as a USB flash drive or similar). Once the applicant has collected the e-Apostille, the user marks the application in the system as "Notified".
    - By downloading it on-line: this only applies if explicitly requested by the applicant. In this case the user only needs to press the "Notify" button and the system will immediately make the e-Apostille available in the e-Apostille Download Module of the Internet publication subsystem.

Public Document Issuers Signatures Database Module

This module allows maintenance of the Public Document Issuers Signatures Database, including the following:

- Creating a new signature of an authority entitled to execute public documents, including data on name, position, institution, signature validity period. The type of signature (holographic or digital) should be indicated. Depending on the type of signature, one or two files are added:
    - For holographic signatures, a file containing the scanned holographic signature is added. The file must be in JPEG format.
    - For digital signatures, a file in DER format with the digital credentials of the public authority (exporting in DER format the public key of the certificate used to sign a public document by a given authority) is added.
- Search and retrieval of signatures of public document issuers, with the option of various searching criteria.

- Updating the signature of a public document issuer. As a preliminary step, the signature to be changed is selected, then the operation is similar to creating a new signature.
- Deleting the signature of a public document issuer. As a preliminary step, the signature to be deleted is selected; it may only be deleted if it is not being referred to by any pending application for an Apostille.

Apostille Management Module:

This module allows management of the database of Apostille applications, including the following:

- Search and retrieval of applications, with the option of various searching criteria.
- Modifying an Apostille application. As a preliminary step, the application to be updated is selected; only applications that have not reached "Validated" status may be modified.
- Deleting an Apostille application. As a preliminary step, application to be deleted is selected; only applications that have not reached "Issued" status may be deleted.
- Issuing a list of Apostille applications that can replace the traditional paper register of Apostilles. As a preliminary step, a selection of applications to include in the list is made.

User Management Module:

This module allows maintenance of system users, including the following:

- Creating a new user, specifying personal information, initial password and permissions.
- Search and retrieval of users, with the option of various search criteria
- Modifying a user profile. As a preliminary step, the user profile to be modified is selected; then the operation is similar to creating a new user.
- Deleting a user profile. As a preliminary step, the user profile to be deleted is selected.

Internet publication subsystem:

e-Register Module

This module allows foreign authorities or other interested person that receive an Apostille to validate it through several verification tasks. In order to access the e-Register and to perform any of these tasks, it is necessary to enter three items of information that identify each Apostille (these are indicated on the Apostille itself):

- Apostille number
- Date of issuance
- Secure Validation Code

The "Secure Validation Code" is an alphanumeric code that allows the e-Apostille document to be clearly identified. This code links the Apostille with the issuing Competent Authority and allows the integrity and authenticity of the Apostille to be verified by accessing the e-Register online. This is a concept recognised under Spanish law, whereby a printout of the e-Apostille is conferred the same legal status as a digitally signed document.

In the event that the e-APP system developed in Spain is implemented in other States which may not recognise this legal concept, it would be enough to assign an alphanumeric code that uniquely identifies each Apostille issued on a given date. That is, a code should not be reused for a given date of issuance.

Once these three items of information have been entered, the foreign authority or other interested person may:

- **Validate the Apostille**. This option simply confirms that there is indeed an Apostille in the system that matches the information entered, and displays a "printable version" (unsigned version that does not contain the public document as an attachment) for the purpose of visual comparison.

- **Validate the Public Document**. If an electronic public document together with an e-Apostille, this option allows the foreign authority/other interested person to verify that the electronic public document matches exactly the one that was apostillised. The applicant uploads the electronic public document file. The e-Register module then calculates the "digital print" (*hash code*) of the document and compares it against the digital print of the electronic public document connected to that Apostille (which was stored in the system). If both codes are the same, a message will appear confirming that the electronic public document uploaded is the same as the one that was apostillised.

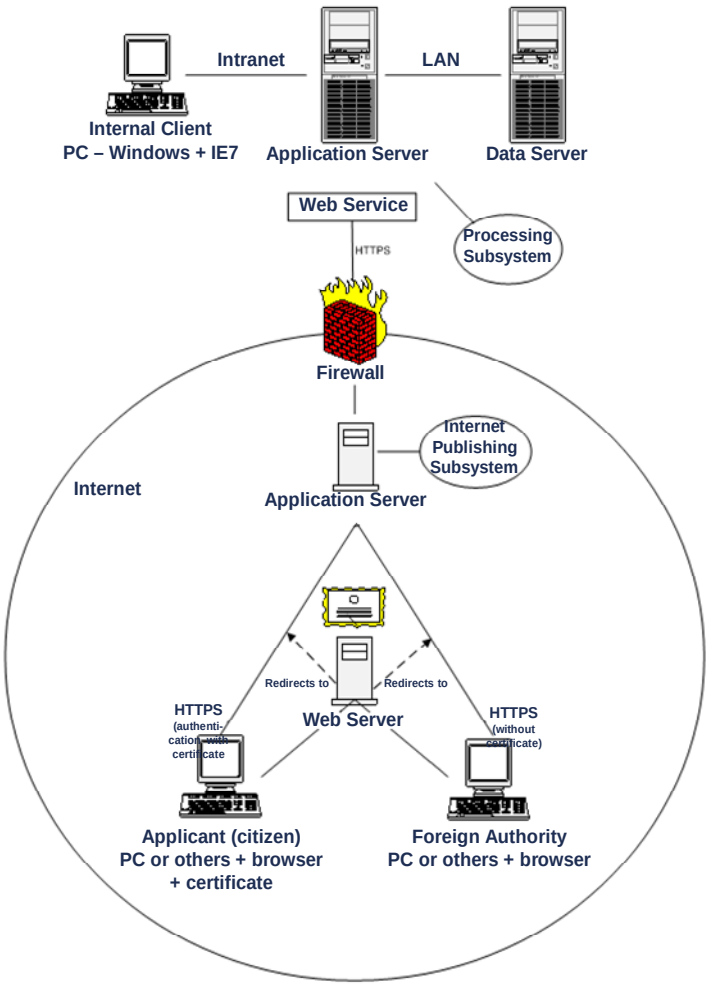
- **e-Apostille signature validation**. If the foreign authority/other interested person does not have a tool capable of opening and verifying the signature of PAdES documents (i.*e*. Adobe Reader X), or is otherwise unable to verify the signature of the Apostille with its own software, this option allows the signature of the Apostille to be verified (including the possible revocation of the certificate used to sign the e-Apostille) by uploading the e-Apostille to the server of the Ministry of Justice.

e-Apostille Download Module:

This module allows an applicant to download the e-Apostille from the Internet once it has been issued. To this end, the applicant must authenticate himself/herself using a username and password. This information appears on the receipt handed to the applicant by the Competent Authority. Once authenticated, the applicant may proceed to download the e-Apostille file.

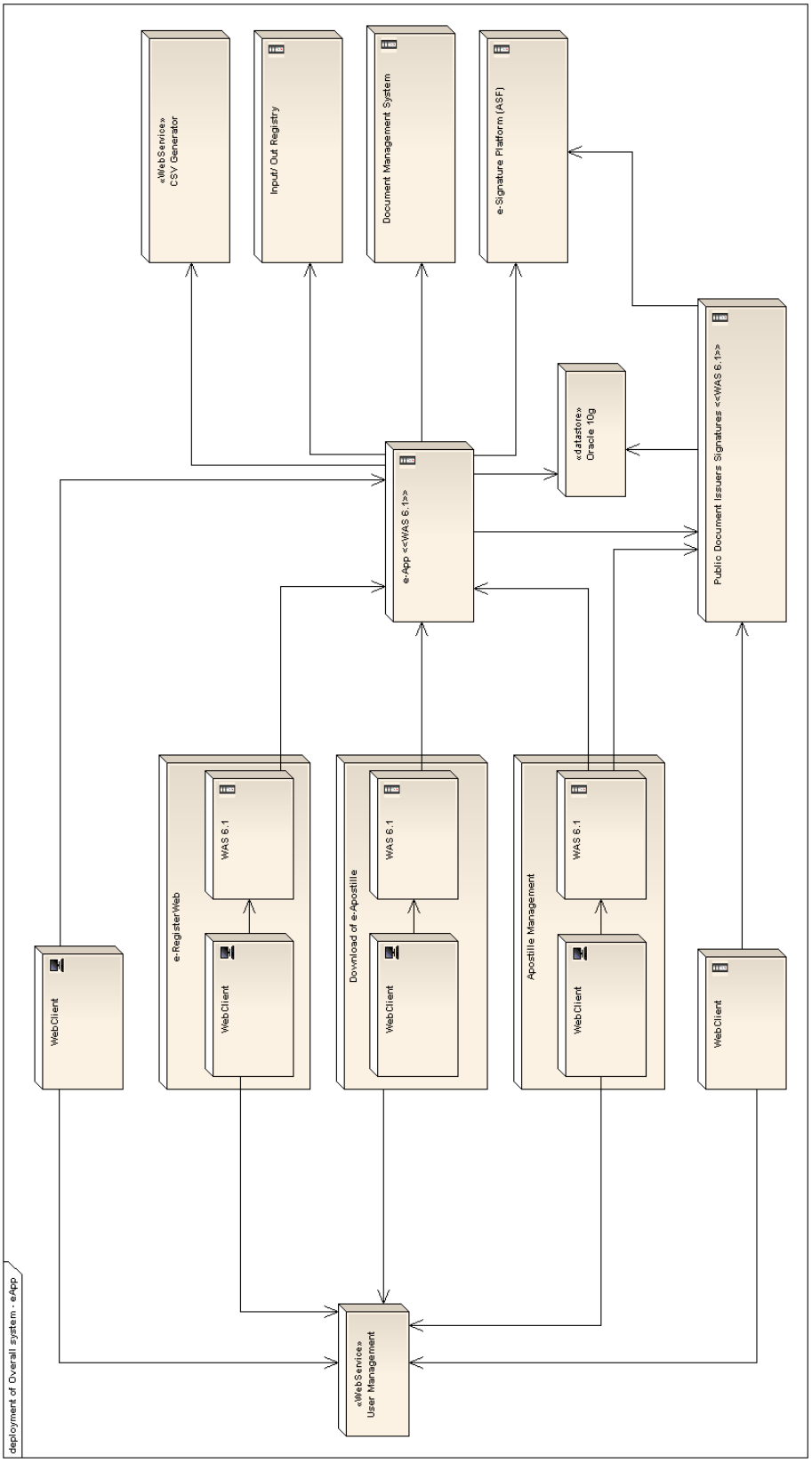
2.4. System deployment architecture

The system deployment architecture of the e-APP system can be illustrated by the following diagram:



2.4. Low level components

Going into greater detail, we find the following design components in the e-APP system:



3. EXPORTING THE E-APP SYSTEM

3.1. Replacement of connectors

Thanks to its modular design based on "connectors", the e-Apostille system allows some of the building modules that are used to implement the communication between the e-APP system and other corporate information systems to be replaced with other modules that better suit the needs of other authorities that may use the system.

A "connector" is basically a class that implements an interface that works as an "agreement" between the application using the connector and the connector itself, thus establishing a standardised communication gateway between both parts. This in turn allows a new implementation of the connector to be written without changing the application source code, as long as the connector complies with the "agreed" interface.

The connectors that may be replaced are:

- **Connector to the service generating Secure Validation Code (SVC).** The e-Apostille application invokes and loads this connector when it needs to generate a validation code that identifies the Apostille. The implementation used by the Spanish Ministry of Justice invokes a horizontal web service of the Ministry that is in charge of generating this code. This connector can be replaced by any other connector that generates a validation code with whatever procedure; the only restriction being that the validation code must be an alphanumeric chain of no more than 50 characters so as to guarantee the unique identifying triplet consisting of "Apostille number/ issuing date / validation code".
  The connector has to present the following interface:

```
public interface IConectorCSV extends IConector {
       public String obtenerCSV(String aplicacion, String tipoServicio)
                      throws GenericException;
}
```

  In other words, the connector must be a Java class that implements the method **obtenerCSV** (obtaincsv) that receives two arguments: **aplicación (aplication)** and **tipoServicio** (servicetype). The e-Apostille system will invoke this method passing the string "APOSTILLA" as the value for both arguments. These arguments are a requirement of the service used by the Ministry of Justice, but they can be ignored in any other implementation of the connector. The method only has to generate an alphanumeric code of up to 50 characters that guarantees that it is not repeated on the same date and for the same number of Apostille. For instance, a code could be generated that includes time-date of the system (year-month-day-hour-minute-second-millisecond) + a random number, as shown in the following example:

```
public String obtenerCSV(String aplicacion, String tipoServicio) {
   String result=null;
   DateFormat dateFormat = new SimpleDateFormat("yyyyMMddHHmmssSSS");
   result = dateFormat.format(Calendar.getInstance().getTime()) +
Integer.toString((int)(Math.random()*9999));
   return result;
   }
}
```

- **Document Management System Connector**. The e-Apostille application invokes and loads this connector when documents have to be stored in or retrieved from the system. It is used both for short term storage of documents (both electronic public documents and scanned paper public documents that have to be apostillised, e-Apostille) as well as long term storage (printable version or "cover sheet" of the Apostille). This connector is also used by the e-Register application to recover and display the printable version. The current implementation of the application stores the documents in a BLOB column of the Oracle table, although the model does allow an implementation that stores the documents in the repository of any other document management system. The only requirement is that the document management system assigns each document an identifier or unique locator that can be stored in a string of up to 50 characters; the document management system must return this same locator in the document archiving operation and must allow the subsequent retrieval of the same document using this identifier as the only parameter.
  The connector has to present the following interface:

```
public interface IConectorGestorDocumental extends IConector {
       public String altaDocumento(InputStream inFile, String descripcion,
                      String extension) throws GenericException;
       public boolean eliminarDocumento(String uid) throws GenericException;
       public DatosDocumento leerDocumento(String uid) throws GenericException;
}
```

In other words, the connector must be a Java class that implements the methods **altaDocumento** (translation: registerdocument), **eliminarDocumento** (translation deletedocument) and **leerDocumento** ((translation readdocument) that will receive as argument:

inFile: stream with the binary content of the file to be stored
descripcion (translation description): textual description of the file
extension: file name extension (pdf, doc, txt, …)
uid: unique identifier of the file in the document repository

The method **altaDocumento** (translation registerdocument) must return a unique identifier of the document once this is filed in the document repository.
The method **eliminarDocumento** (translation deletedocument) must return "**true**" if it managed to completely delete the document or otherwise raise an exception.

The method **leerDocumento** (translation readdocument) must return an object of the class **DatosDocumento** (translation documentdata) that contains all the information regarding the document, including its binary content. The definition of this class is shown in Annex I to this document.

The Ministry of Justice of Spain can provide interested agencies with the Document Management System connector that stores the documents in the database BLOB fields. In the specific case of the Ministry of Justice, the Oracle Database is used for this purpose, but this same connector could work for other database engines such as MySQL. The adaptations to perform within the connector are the same as those described later in section 3.3 "Changing the Database manager".

- **Electronic Signature Platform Connector**. The e-Apostille and the e-Register applications invoke and load this connector whenever an electronic signature is to be verified. The e-Apostille application uses this connector to verify both the signature on electronic public documents and on the e-Apostille itself once this has been issued. In the case of the e-Register application, this connector is used for the option "e-Apostille signature validation". It is also used by the Public Document Issuers Signatures Database Module to obtain the credentials (issuer and serial number) of the electronic certificates in DER format that are uploaded and associated to every signatory. As such, this connector may be replaced by any other connector that uses a different signature platform service , as long as the platform provides the following services:

  - Electronic signature verification of a document
  - DER format certificate data extraction (namely, issuer and serial number)

In future versions, it is foreseen that the e-Apostille system will allow Apostilles to be signed using "public institution certificates". These are a type of server side signatures, therefore, when this type of signature is implemented, the digital signature platform connector will also be used to sign Apostilles. The connector would then need to be adjusted to enable this feature. The interface to be provided by this connector is currently:

```
public interface IConectorPlataformaFirma extends IConector {
       public ValidacionResponse validarCertificado(InputStream certificadoDER,
                      String datosAdicionales) throws GenericException;

       public DatosValidarDocumentoFirmado[] validarDocumentoFirmado(
                      InputStream contenido, String datosAdicionales)
                      throws GenericException;
}
```

That is, the connector must be a Java Class that implements the methods **validarCertificado** (validatecertificate) and **validardocumentofirmado** (validatesigneddocument) that will receive as arguments:

certificadoDER (translation certificateDER): Exported binary content of the public part of a digital certificate in DER format

datosAdicionales (translation additionaldata): String with possible values "XADES", "CADES" or "PADES" that shows the kind of document to be validated. The method **validarCertificado** (translation validatecertificate) does not use this argument.
Contenido (translation content): binary content of the signed document to be validated.

The **validarCertificado** (translation validatecertificate) method returns a **ValidacionResponse** (translation validateresponse) class instance containing information concerning the certificate supplied as argument. The method **validarDocumentoFirmado** (translation validatesigneddocument) returns an array of class objects **DatosValidarDocumentoFirmado** (translation validatesigneddocumentdata) containing information regarding the signature(s) in the document. The definition of both classes is shown in Annex I to this document.

3.2. Replacing the client signature mechanism

The e-Apostille application currently uses a Java *applet* called *WebSigner*, from TB Solutions, to electronically sign the Apostilles on the client (browser). A *javascript* interface is used for the interaction with this *applet*. It is possible to replace this *applet* with another client signing solution, although in this case replacing the connector would not be enough and it would be necessary to modify the *javascript* code used to access the *applet*. As long as the chosen solution has a *javascript* programming interface similar to the one offered by WebSigner it would be possible to replace this component even though this change would be more "intrusive" as it implies modifying the javascript code of the e-Apostille web application.

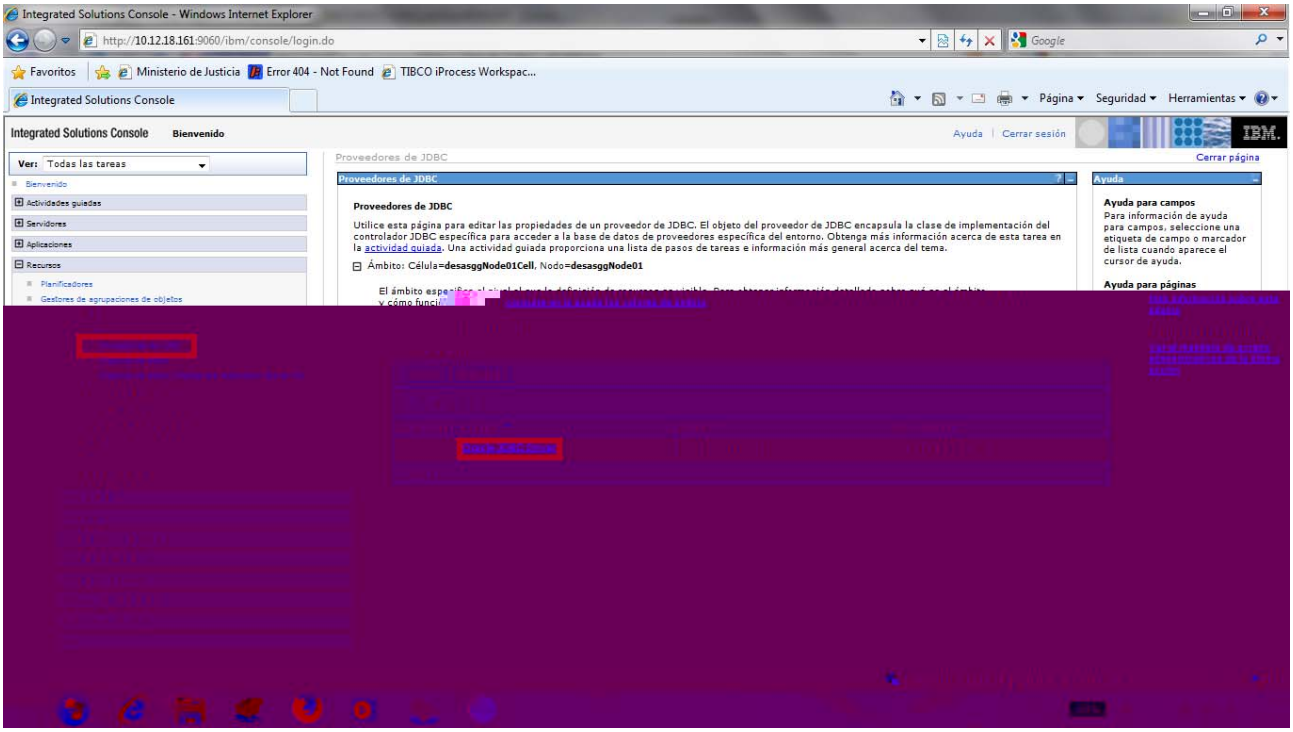
When, in future versions, the possibility of server-side electronic signature using the signing services platform is supported, the signing operation could be performed using a connector with such a platform. In this case, the replacement of the signing platform connector will allow the use of different signing platforms for server-side electronic signature.

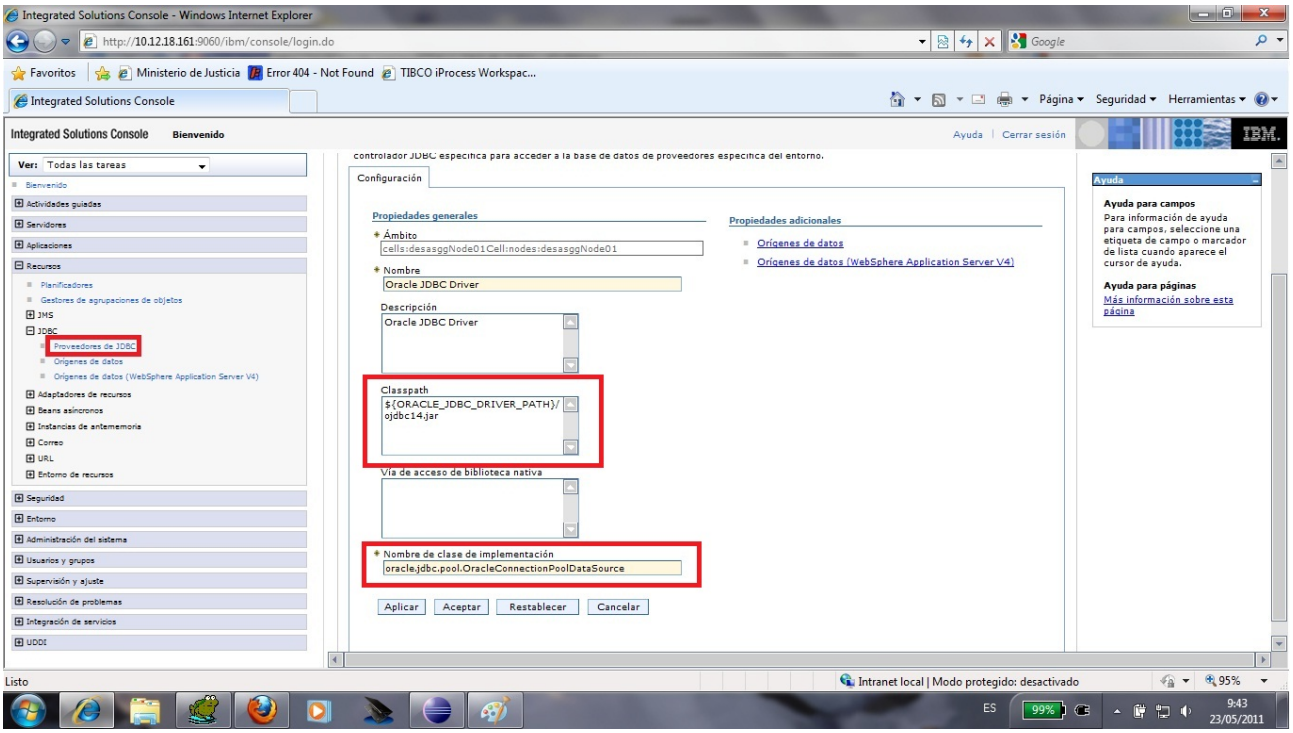
3.3. Changing the Database manager

All the modules that are integrated in the e-APP system use Hibernate object-relational mapping Framework. This software layer isolates applications from the specific relational database engine used. Therefore, any other database manager supported by Hibernate may be used; the only changes to perform would be:

- Set up a JDBC *driver* and create a *datasource* that points out to the database to be used
- Establish the *name JNDI* of the *datasource* in the properties files
- Modify in the properties files the SQL "dialect" corresponding to the database manager to be used

The configuration of the JDBC driver is done in the configuration application of the WebSphere Application Server:





For instance, in the case of **MySQL**, it would be necessary to introduce in *Classpath* the route to the jar module that contains the MySQL libraries: **`.../mysql.jar`** and in *Name of the implementation class*: **`com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource`**

Then, a *datasource* for this *JDBC driver* must be created and a *JNDI name* assigned to it.

Both the JNDI name and the SQL "dialect" SQL must be set up in the **applicationContext-configuration.xml** files that are within the relative path: **services/src/main/resources/common/hibernate** inside each of the projects that compose the e-Apostille system. For instance, for the e-Apostille application, the file to be modified should be: **/eApp/eApp-services/src/main/resources/common/hibernate/applicationContext-configuration.xml**

The following entries highlighted in yellow are the ones to be modified:

```
<bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
   <property name="dataSource" ref="dataSource" />
   <property name="mappingResources">
      <ref bean="mappingResources" />
   </property>
   <property name="hibernateProperties">
      <props>
         <prop key="hibernate.hbm2ddl.auto">none</prop>
         <prop key="hibernate.dialect">org.hibernate.dialect.Oracle10gDialect</prop>
         <prop key="hibernate.show_sql">false</prop>
      </props>
   </property>
</bean>
<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
   <property name="resourceRef"><value>false</value></property>
   <property name="jndiName"><value>jdbc/eapp</value></property>
</bean>
```

The *name JNDI* of the example (`jdbc/eapp`) should be replaced by the one assigned by the JDBC *datasource that is going to be used*.

Regarding the dialect, in order to use for example **MySQL 5.x**, the following entry has to be used:

```
<prop key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
```

The list of dialects supported by the Hibernate documentation can be found at:

http://www.163jsp.com/help/hibernate32api/org/hibernate/dialect/Dialect.html

3.4. Integration through Web Services

The integration layer within the e-APP system allows integration of external applications with complete modules of the system. Specifically, integration with the e-Apostille and e-Register modules is possible, following the steps detailed below.

3.4.1. Integration with e-Apostille

The e-Apostille integration web service publishes methods that allow an external application to:

- Create a new e-Apostille request, starting its processing flow (**`solicitarApostilla`** method) translation: requestApostille method.
- Know the status of an Apostille being processed, and specifically knowing if it has already been issued and signed (**`consultarApostilla`** method) translation: consultApostille
- Retrieve an Apostille already signed (**`obtenerApostilla`** method) translation: obtainApostille
- Cancel an Apostille request (previously performed through the web Service), if it has not been processed yet (**`anularApostilla`** method) translation: cancelApostille

Through this service, an external application that generates electronic public documents could provide an option to "mark", at the user interface, if they wish to obtain the requested public document already with an Apostille. In this case, the application, after generating and signing the public document would create an e-Apostille request for that electronic public document. This request would be processed through the regular flow of the e-Apostille application. The external application could ask the e-Apostille system periodically if the request has been fully processed, and if so, retrieve the e-Apostille and forward it to the final user.

Annex II to this document includes the WSDL definition of this service.

3.4.2. Integration with e-Register

The web service for integration with the e-Apostille application publishes a method that allows an external application to generate e-Apostilles and include them in the e-Register. To that end, the external application should provide at least:

- e-Apostille identification data: Number, date of issuance, validation code
- Other minimum data required by the Hague Apostille Convention for registering Apostilles:
    - Name of the person who has signed the public document
    - Capacity in which he/she has acted
    - Name of the authority which has affixed the seal or stamp
- A PDF file with a "printable version" of the e-Apostille

If the "public document hash value verification function" is to be made available for users, the following integration method must also be provided:

- The public document hash, generated by any of the supported algorithms
- The identifying name of the algorithm used

The supported *hash* algorithms are:

- SHA-1
- MD5

It should be noted that in order for the e-Register function "e-Apostille signature validation" to be available, a service platform and its corresponding connector needs to be available.

The definition of this service in WDSL is not yet available, but could be provided to interested parties once available.

3.5. Adaptation of Master Tables

In addition to the appropriate software adjustments to export the e-APP system of the Spanish Ministry of Justice to other agencies, it would be necessary to adjust the content of some of the master data tables that contain information regarding system configuration. The tables to be adjusted are the following:

- Locations Table (TC_SEDE). This must be loaded with data on the different locations of the Competent Authorities whose competence relates to the agency that is going to implement the solution. The definition can be found in Annex III.

3.6. Localisation

If the system is implemented in a non Spanish speaking country, both the static text on the forms and the contents of the "master" tables should be translated to the language of the implementation country (localisation process).

To adjust static text in this regard, properties files (text files), where the texts are stored and which are deployed together with the application on the WAS server, need to be modified. These files can be found in the path **src/main/resources/locale** of each module that is integrated in the e-APP system. For example, regarding the e-Apostille module, the properties file of each language can be found following the path: **eApp/eApp-webapp/src/main/resources/locale**

The master tables to be adjusted are:

- Country Tables (TC_PAIS)
- Status Tables (TC_ESTADO)

NOTE: The tables containing "Provinces" (TC_PROVINCIA) and "Municipalities" (TC_MUNICIPIO) are only active when the country selected as place of residence is "Spain"; therefore when implementing the system in other countries, these tables do not have to be adjusted nor fed with the specific territory information relating to the country where the solution is to be implemented. In that case, the municipality and/or province or similar to which the address belongs can be entered by typing it in manually.

Alternately, if the territory model of the country where the solution is to be implemented is similar to Spain's (division into provinces and municipalities or similar items), it is possible to adjust slightly the application code so that it takes into account those fields when choosing the country in question. In this case, the tables containing provinces and municipalities have to be fed with those corresponding to the country.
Annex III to this document shows the definition of all these tables.

ANNEX I. CLASS DEFINITIONS

Class DatosDocumento

```
public class DatosDocumento implements java.io.Serializable {
	private static final long serialVersionUID = -604263192269590408L;
	private String doUidDocumento;
	private String doDescripcion;
	private OutputStream doArchivo;
	private String doExtension;

	public DatosDocumento() {
	}

	public String getDoUidDocumento() {
		return this.doUidDocumento;
	}

	public void setDoUidDocumento(String doUidDocumento) {
		this.doUidDocumento = doUidDocumento;
	}

	public String getDoDescripcion() {
		return this.doDescripcion;
	}

	public void setDoDescripcion(String doDescripcion) {
		this.doDescripcion = doDescripcion;
	}

	public String getDoExtension() {
		return this.doExtension;
	}

	public void setDoExtension(String doExtension) {
		this.doExtension = doExtension;
	}

	public OutputStream getDoArchivo() {
		return doArchivo;
	}

	public void setDoArchivo(OutputStream doArchivo) {
		this.doArchivo = doArchivo;
	}
}
```

Class ValidacionResponse

```
public class ValidacionResponse implements IDescribeError {
	private static final long serialVersionUID = 3567193432574376046L;
	private String codError;
	private String descError;
	private String issuer;
	private String subject;
	private String serialNumber;
	private Calendar validFrom;
	private Calendar validUntil;
	private int certState;

	public String getIssuer() {
		return issuer;
	}

	public void setIssuer(String issuer) {
		this.issuer = issuer;
	}

	public String getSerialNumber() {
		return serialNumber;
	}

	public void setSerialNumber(String serialNumber) {
		this.serialNumber = serialNumber;
	}

	public Calendar getValidFrom() {
		return validFrom;
	}

	public void setValidFrom(Calendar validFrom) {
		this.validFrom = validFrom;
	}

	public Calendar getValidUntil() {
		return validUntil;
	}

	public void setValidUntil(Calendar validUntil) {
		this.validUntil = validUntil;
	}

	public int getCertState() {
		return certState;
	}

	public void setCertState(int certState) {
		this.certState = certState;
	}


	public String getCodError() {
```

```
		return this.codError;
	}

	public String getDescError() {
		return this.descError;
	}

	public void setCodError(String codigo) {
		this.codError = codigo;
	}

	public void setDescError(String descripcion) {
		this.descError = descripcion;
	}

	public void setSubject(String subject) {
		this.subject = subject;
	}

	public String getSubject() {
		return subject;
	}
}
```

Class DatosValidarDocumentoFirmado

```
public class DatosValidarDocumentoFirmado implements IDescribeError {
	private static final long serialVersionUID = 3567193432574376046L;
	private String codError;
	private String descError;
	private String issuer;
	private String subject;
	private Calendar validFrom;
	private Calendar validUntil;
	private Calendar signDate;
	private String serialNumber;
	private int resultCode;

	public String getIssuer() {
		return issuer;
	}

	public void setIssuer(String issuer) {
		this.issuer = issuer;
	}

	public Calendar getValidFrom() {
		return validFrom;
	}

	public void setValidFrom(Calendar validFrom) {
		this.validFrom = validFrom;
	}

	public Calendar getValidUntil() {
		return validUntil;
	}

	public void setValidUntil(Calendar validUntil) {
		this.validUntil = validUntil;
	}

	public String getCodError() {
		return this.codError;
	}

	public String getDescError() {
		return this.descError;
	}

	public void setCodError(String codigo) {
		this.codError = codigo;
	}

	public void setDescError(String descripcion) {
		this.descError = descripcion;
	}

	public void setSubject(String subject) {
```

```
		this.subject = subject;
	}

	public String getSubject() {
		return subject;
	}

	public void setSignDate(Calendar signDate) {
		this.signDate = signDate;
	}

	public Calendar getSignDate() {
		return signDate;
	}

	public void setSerialNumber(String serialNumber) {
		this.serialNumber = serialNumber;
	}

	public String getSerialNumber() {
		return serialNumber;
	}

	public void setResultCode(int resultCode) {
		this.resultCode = resultCode;
	}

	public int getResultCode() {
		return resultCode;
	}
}
```

ANNEX II. WEB SERVICES WSDL DEFINITIONS

WSDL of the Web Service for integration with e-Apostille

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://impl.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://impl.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns:intf="http://impl.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns:tns1="http://utils.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema elementFormDefault="qualified"
targetNamespace="http://utils.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns="http://www.w3.org/2001/XMLSchema">
   <complexType name="InfConsulApostilla">
    <sequence>
     <element name="csv" nillable="true" type="xsd:string"/>
     <element name="numApostilla" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="DatosConsulApostilla">
    <sequence>
     <element name="autoridadApostillante" nillable="true" type="xsd:string"/>
     <element name="autoridadFirmante" nillable="true" type="xsd:string"/>
     <element name="calidadFirmante" nillable="true" type="xsd:string"/>
     <element name="codError" nillable="true" type="xsd:string"/>
     <element name="comparecenciaElectronica" type="xsd:boolean"/>
     <element name="cp" nillable="true" type="xsd:string"/>
     <element name="csv" nillable="true" type="xsd:string"/>
     <element name="descError" nillable="true" type="xsd:string"/>
     <element name="direccionSolicitante" nillable="true" type="xsd:string"/>
     <element name="emailSolicitante" nillable="true" type="xsd:string"/>
     <element name="estado" nillable="true" type="xsd:string"/>
     <element name="fechaAnulacion" nillable="true" type="xsd:dateTime"/>
     <element name="fechaCreacion" nillable="true" type="xsd:dateTime"/>
     <element name="fechaDescarga" nillable="true" type="xsd:dateTime"/>
     <element name="fechaEmision" nillable="true" type="xsd:dateTime"/>
     <element name="fechaFirma" nillable="true" type="xsd:dateTime"/>
     <element name="fechaNotificacion" nillable="true" type="xsd:dateTime"/>
     <element name="fechaValidacion" nillable="true" type="xsd:dateTime"/>
     <element name="fechaVigencia" nillable="true" type="xsd:dateTime"/>
     <element name="identificadorSolicitante" nillable="true" type="xsd:string"/>
     <element name="localizador" nillable="true" type="xsd:string"/>
     <element name="movRechazo" nillable="true" type="xsd:string"/>
     <element name="municipioSolicitante" nillable="true" type="xsd:string"/>
     <element name="nombreSolicitante" nillable="true" type="xsd:string"/>
     <element name="numApostilla" nillable="true" type="xsd:string"/>
     <element name="numRegEntrada" nillable="true" type="xsd:string"/>
     <element name="numRegSalida" nillable="true" type="xsd:string"/>
     <element name="organismo" nillable="true" type="xsd:string"/>
     <element name="paisDestino" type="xsd:int"/>
```

```
  <element name="paisDestinoDesc" nillable="true" type="xsd:string"/>
  <element name="paisSolicitante" type="xsd:int"/>
  <element name="paisSolicitanteDesc" nillable="true" type="xsd:string"/>
  <element name="provinciaSolicitante" type="xsd:int"/>
  <element name="provinciaSolicitanteDesc" nillable="true" type="xsd:string"/>
  <element name="telefonoSolicitante" nillable="true" type="xsd:string"/>
  <element name="tipoDocumento" type="xsd:boolean"/>
  <element name="url" nillable="true" type="xsd:string"/>
 </sequence>
</complexType>
<complexType name="InfSolApostilla">
 <sequence>
  <element name="DNI_solicitante" nillable="true" type="xsd:string"/>
  <element name="apAutoridadApostillante" nillable="true" type="xsd:string"/>
  <element name="apAutoridadFirmante" nillable="true" type="xsd:string"/>
  <element name="apCalidadFirmante" nillable="true" type="xsd:string"/>
  <element name="apComparecenciaElect" type="xsd:boolean"/>
  <element name="apCp" nillable="true" type="xsd:string"/>
  <element name="apDireccion" nillable="true" type="xsd:string"/>
  <element name="apEmail" nillable="true" type="xsd:string"/>
  <element name="apLocalizadorDpElect" nillable="true" type="xsd:string"/>
  <element name="apMunicipio" nillable="true" type="xsd:string"/>
  <element name="apOrganismo" nillable="true" type="xsd:string"/>
  <element name="apTelefono" nillable="true" type="xsd:string"/>
  <element name="apTipoFormatoDp" type="xsd:int"/>
  <element name="apUrlDpElect" nillable="true" type="xsd:string"/>
  <element name="apellidos_solicitante" nillable="true" type="xsd:string"/>
  <element name="descripcionDP" nillable="true" type="xsd:string"/>
  <element name="docPublico" nillable="true" type="xsd:base64Binary"/>
  <element name="extensionDP" nillable="true" type="xsd:string"/>
  <element name="idPais" type="xsd:int"/>
  <element name="idPaisDest" type="xsd:int"/>
  <element name="idProv" type="xsd:int"/>
  <element name="idSede" type="xsd:long"/>
  <element name="nombre_solicitante" nillable="true" type="xsd:string"/>
  <element name="usr_solicitante" nillable="true" type="xsd:string"/>
 </sequence>
</complexType>
<complexType name="DatosSolApostilla">
 <sequence>
  <element name="DNI_solicitante" nillable="true" type="xsd:string"/>
  <element name="apAutoridadApostillante" nillable="true" type="xsd:string"/>
  <element name="apAutoridadFirmante" nillable="true" type="xsd:string"/>
  <element name="apCalidadFirmante" nillable="true" type="xsd:string"/>
  <element name="apComparecenciaElect" type="xsd:boolean"/>
  <element name="apCp" nillable="true" type="xsd:string"/>
  <element name="apCsv" nillable="true" type="xsd:string"/>
  <element name="apDescripcionDp" nillable="true" type="xsd:string"/>
  <element name="apDireccion" nillable="true" type="xsd:string"/>
  <element name="apEmail" nillable="true" type="xsd:string"/>
  <element name="apHash" nillable="true" type="xsd:string"/>
  <element name="apLocalizadorDpElect" nillable="true" type="xsd:string"/>
  <element name="apMotivoRechazo" nillable="true" type="xsd:string"/>
  <element name="apMunicipio" type="xsd:long"/>
  <element name="apNumRegEntrada" nillable="true" type="xsd:string"/>
  <element name="apNumRegSalida" nillable="true" type="xsd:string"/>
```

```
      <element name="apNumeroApostilla" nillable="true" type="xsd:string"/>
      <element name="apOrganismo" nillable="true" type="xsd:string"/>
      <element name="apTelefono" nillable="true" type="xsd:string"/>
      <element name="apTipoFormatoDp" type="xsd:int"/>
      <element name="apUrlDpElect" nillable="true" type="xsd:string"/>
      <element name="apellidos_solicitante" nillable="true" type="xsd:string"/>
      <element name="codError" nillable="true" type="xsd:string"/>
      <element name="descError" nillable="true" type="xsd:string"/>
      <element name="descPais" nillable="true" type="xsd:string"/>
      <element name="descPaisDest" nillable="true" type="xsd:string"/>
      <element name="descProv" nillable="true" type="xsd:string"/>
      <element name="docPublico" nillable="true" type="xsd:base64Binary"/>
      <element name="extensionDP" nillable="true" type="xsd:string"/>
      <element name="idPais" type="xsd:int"/>
      <element name="idPaisDest" nillable="true" type="xsd:string"/>
      <element name="idProv" type="xsd:int"/>
      <element name="idSede" type="xsd:int"/>
      <element name="nombre_solicitante" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
    <complexType name="InfAnulApostilla">
     <sequence>
      <element name="csv" nillable="true" type="xsd:string"/>
      <element name="numApostilla" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
    <complexType name="DatosAnulApostilla">
     <sequence>
      <element name="codError" nillable="true" type="xsd:string"/>
      <element name="descError" nillable="true" type="xsd:string"/>
      <element name="resultadoAnulacion" type="xsd:boolean"/>
     </sequence>
    </complexType>
    <complexType name="InfObtenerApostilla">
     <sequence>
      <element name="csv" nillable="true" type="xsd:string"/>
      <element name="numApostilla" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
    <complexType name="DatosObtenerApostilla">
     <sequence>
      <element name="codError" nillable="true" type="xsd:string"/>
      <element name="descError" nillable="true" type="xsd:string"/>
      <element name="fichero" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
  </schema>
  <schema elementFormDefault="qualified"
targetNamespace="http://impl.ws.eApostille.eApp.apostilla.majus.ntj.mju"
xmlns="http://www.w3.org/2001/XMLSchema">
    <import namespace="http://utils.ws.eApostille.eApp.apostilla.majus.ntj.mju"/>
    <element name="consulta" type="tns1:InfConsulApostilla"/>
    <element name="consultarApostillaReturn" type="tns1:DatosConsulApostilla"/>
    <element name="solicitud" type="tns1:InfSolApostilla"/>
    <element name="solicitarApostillaReturn" type="tns1:DatosSolApostilla"/>
    <element name="anulacion" type="tns1:InfAnulApostilla"/>
```

```
  <element name="anularApostillaReturn" type="tns1:DatosAnulApostilla"/>
  <element name="apostilla" type="tns1:InfObtenerApostilla"/>
  <element name="obtenerApostillaReturn" type="tns1:DatosObtenerApostilla"/>
 </schema>
</wsdl:types>

  <wsdl:message name="anularApostillaResponse">
     <wsdl:part element="impl:anularApostillaReturn" name="anularApostillaReturn">
     </wsdl:part>
  </wsdl:message>

  <wsdl:message name="anularApostillaRequest">
     <wsdl:part element="impl:anulacion" name="anulacion">
     </wsdl:part>
  </wsdl:message>

  <wsdl:message name="solicitarApostillaRequest">
     <wsdl:part element="impl:solicitud" name="solicitud">
     </wsdl:part>
  </wsdl:message>

  <wsdl:message name="consultarApostillaRequest">
     <wsdl:part element="impl:consulta" name="consulta">
     </wsdl:part>
  </wsdl:message>

  <wsdl:message name="solicitarApostillaResponse">
     <wsdl:part element="impl:solicitarApostillaReturn" name="solicitarApostillaReturn">
     </wsdl:part>
  </wsdl:message>

  <wsdl:message name="consultarApostillaResponse">
     <wsdl:part element="impl:consultarApostillaReturn" name="consultarApostillaReturn">
     </wsdl:part>
  </wsdl:message>

  <wsdl:message name="obtenerApostillaRequest">
     <wsdl:part element="impl:apostilla" name="apostilla">
     </wsdl:part>
  </wsdl:message>

  <wsdl:message name="obtenerApostillaResponse">
     <wsdl:part element="impl:obtenerApostillaReturn" name="obtenerApostillaReturn">
     </wsdl:part>
  </wsdl:message>
```

```
   <wsdl:portType name="ServicioSolicitarApostillas">
      <wsdl:operation name="consultarApostilla" parameterOrder="consulta">
         <wsdl:input message="impl:consultarApostillaRequest"
name="consultarApostillaRequest">
       </wsdl:input>

         <wsdl:output message="impl:consultarApostillaResponse"
name="consultarApostillaResponse">
       </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="solicitarApostilla" parameterOrder="solicitud">

         <wsdl:input message="impl:solicitarApostillaRequest"
name="solicitarApostillaRequest">
       </wsdl:input>
         <wsdl:output message="impl:solicitarApostillaResponse"
name="solicitarApostillaResponse">
       </wsdl:output>
      </wsdl:operation>

      <wsdl:operation name="anularApostilla" parameterOrder="anulacion">
         <wsdl:input message="impl:anularApostillaRequest" name="anularApostillaRequest">
       </wsdl:input>
         <wsdl:output message="impl:anularApostillaResponse" name="anularApostillaResponse">
       </wsdl:output>
      </wsdl:operation>

      <wsdl:operation name="obtenerApostilla" parameterOrder="apostilla">
         <wsdl:input message="impl:obtenerApostillaRequest" name="obtenerApostillaRequest">
       </wsdl:input>
         <wsdl:output message="impl:obtenerApostillaResponse"
name="obtenerApostillaResponse">
       </wsdl:output>
      </wsdl:operation>

   </wsdl:portType>

   <wsdl:binding name="ServicioSolicitarApostillasSoapBinding"
type="impl:ServicioSolicitarApostillas">
      <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="consultarApostilla">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="consultarApostillaRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="consultarApostillaResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
```

```
      </wsdl:operation>
      <wsdl:operation name="solicitarApostilla">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="solicitarApostillaRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="solicitarApostillaResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>

      <wsdl:operation name="anularApostilla">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="anularApostillaRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="anularApostillaResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>

      <wsdl:operation name="obtenerApostilla">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="obtenerApostillaRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="obtenerApostillaResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>

   </wsdl:binding>

   <wsdl:service name="ServicioSolicitarApostillasService">
      <wsdl:port binding="impl:ServicioSolicitarApostillasSoapBinding"
name="ServicioSolicitarApostillas">
         <wsdlsoap:address
location="http://localhost:8080/ServicioSolicitarApostillasWS/services/ServicioSolicitarApost
illas"/>
      </wsdl:port>
   </wsdl:service>

</wsdl:definitions>
```

ANNEX III. RELATIONAL TABLES DEFINITIONS

Competent Authorities locations table

TC_SEDE				
Stores data of the locations of the Competent Authorities				
Column	**Data Type**	**Description**	**Compulsory**	**Values**
SE_CLAVE	NUMBER (12)	Unique identifier for locations of Competent Authorities	YES	
SE_NOMBRE	VARCHAR2 (50)	Code of the Competent Authority (obtained from the compatibility test page given by the Ministry of Justice)	YES	
SE_DESCRIPCION	VARCHAR2 (150)	Location's description	NO	
SE_MUNICIPIO	VARCHAR2 (100)	Location's town	YES	
SE_TELEFONO	VARCHAR2 (35)	Location's telephone	NO	
SE_FAX	VARCHAR2 (35)	Location's fax	NO	
SE_DIRECCION	VARCHAR2 (250)	Location's Address	NO	
SE_CP	VARCHAR2 (5)	Location's postal code	NO	
SE_EMAIL	VARCHAR2 (150)	Location's e-mail	NO	

Status table

TC_ESTADO				
Stores the different status of Apostilles				
Column	**Data Type**	**Description**	**Compulsory**	**Values**
ES_CLAVE	NUMBER (2)	Unique identifier of the status of an Apostille	YES	Values: - 1: INITIATED - 2: PENDING VALIDATION - 3: VALIDATED - 4: ISSUED - 5: NOTIFIED - 6: REJECTED - 7: DELETED - 8: END DOWNLOAD
ES_NOMBRE	VARCHAR2 (50)	Name of status	YES	
ES_DESCRIPCION	VARCHAR2 (150)	Description of the status of an Apostille	NO	

Country table

TC_PAIS				
Contains data for the country that the application needs to know at a given moment in the process of issuing an Apostille				
Column	**Data Type**	**Description**	**Compulsory**	**Values**
PA_CLAVE	NUMBER (3)	Unique identifier of the country	YES	
PA_NOMBRE	VARCHAR2 (100)	Name of the country	YES	

Province table

TC_PROVINCIA				
Contains data for the province that the application needs to know at a given moment in the process of issuing an Apostille				
Column	**Data Type**	**Description**	**Compulsory**	**Values**
PR_CLAVE	NUMBER (2)	Unique identifier of the province	YES	
PR_NOMBRE	VARCHAR2 (100)	Name of the province	YES	

Municipality table

TC_MUNICIPIO				
Contains the data for the municipality that the application needs to know at a given moment in the process of issuing an Apostille				
Column	**Data Type**	**Description**	**Compulsory**	**Values**
MU_CLAVE	NUMBER (4)	Unique identifier of the municipality	YES	
MU_NOMBRE	VARCHAR2 (100)	Name of the municipality	YES	
MU_PR_CF_PROV	NUMBER (2)	Unique identifier of the province to which the municipality belongs	YES	