

Ministry of Information and Communications Technology (MoICT)

RAD Environment Interfaces with the Existing Web Services Channels

Ver 1.2

*Ministry of Information and Communications (MoICT)*

*Information Technology Department*

Table of Contents

Table of Contents 3

1 Introduction 6

2 Purpose 6

2.1 Related Documents 6

2.2 Acronyms, Abbreviations, Terms and Definitions 7

3 GSB Browser 8

3.1 US-GB-1 Authenticated User List Web-services and Open WSDL/XSD File 9

3.2 List of Web-Services Sequence Diagram: 9

1.1.1 Open WSDL File Sequence Diagram: 10

1.1.2 Flow 10

1.1.3 Assumptions: 10

1.2 US-GB-2 Organization Admin Assign Permission for GSB Browser Tab 11

1.2.1 Flow 11

1.2.2 Assumptions: 11

4 Email Module 12

4.1 Typical usage scenario 12

4.2 Configuration 12

4.3 Dependencies 12

4.4 Java libraries: 12

4.5 Email Setup (Basic Configuration) 13

4.5.1 User Role: SystemAdmin 13

4.5.2 Menu System 13

4.6 Email Configuration Test 14

4.6.1 Menu System 14

4.7 Email Template 14

4.7.1 User Role: SystemAdmin 14

4.7.2 Menu System 14

5 PushSMS Module 16

5.1 Typical usage scenario 16

5.2 Configuration 16

5.3 Dependencies 16

5.4 Java libraries: 16

5.5 SMS Setup (Basic Configuration) 16

5.5.1 User Role: SystemAdmin 16

5.5.2 Menu System 16

5.6 SMS Configuration Test 17

5.6.1 User Role: SystemAdmin 17

5.6.2 Menu System 17

5.7 SMS Overview 18

5.7.1 User Role: SystemAdmin 18

5.7.2 Menu System 18

6 eFawateer Module 19

6.1 Typical usage scenario 19

6.2 Configuration 20

6.3 Dependencies 20

6.4 Mendix modules 20

6.5 Java libraries: 20

6.6 eFawateer Configuration (Basic Configuration) 20

6.6.1 User Role: SystemAdmin 20

6.6.2 Menu System 20

6.7 Certificate Details 22

6.8 iBiller Web Service 23

6.8.1 Operations: 23

6.8.2 WSDL: 29

6.9 Digital Signature and Message Signing in eFAWATEERcom 33

6.10 1. Certificate Generation 33

6.11 2. Certificate Authority's Root Certificate Installation 33

6.12 How to install Root Certificate 33

6.13 Installing the participant's Certificate 34

6.13.1 Request Signing 34

6.13.2 Digital Signature Algorithm: 34

7 E-Services List (Portal) Module 35

7.1 Typical usage scenario 35

7.2 Configuration 35

7.3 Services Setup (Basic Configuration) 35

8 Appendix A 37

Approvals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Designation | Date | Signature |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
| 10/03/2017 | Mahmoud Al Habbal | 1.0 | Initial draft |
| 11/07/2017 | Nael Mansi | 1.1 | Adding eService list and other amendments. |
| 30/07/2017 | Nael Mansi | 1.2 | Reflecting the changes after eFawateercom and MoICT meeting. |
| 31/07/2017 | Nael Mansi | 2.0 | After review from the MoICT team. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Distribution

|  |  |
| --- | --- |
| Name | Project Role |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

The implementation of RAD Platform has several components; some of them are products such as On-Premises Mendix Collaboration tool®, but in top of it there will be a customization of MOICT needs such as GSB Browser and integration interfaces, this document focuses on customized components.

# Purpose

As part of the implementing RAD Platform in MOICT, Jordan, the Software Requirement Specification document is provided to cover the development part of the platform that will be made to fulfill following subsystems:

1. GSB Browser
2. Email Module
3. PushSMS Module
4. eFawaterCOM Module
5. eService-list (Portal)

## Related Documents

* **Solution Architect document**

## Acronyms, Abbreviations, Terms and Definitions

|  |  |  |
| --- | --- | --- |
| No. | Acronyms | Description/Definition |
| 1 | MOICT | Ministry of Information and Communications Technology |
| 2 | RAD | Rapid Application Development |
| 3 | MOI | Ministry of Interior |
| 4 | DB | Database |
| 5 | SMS | Short Message Service |
| 6 | WSDL | Web Services Description Language |
| 7 | SOAP | Simple Object Access Protocol |
| 8 | REST | Representational State Transfer |
| 9 | GUI | Graphical User Interface |
| 10 | SOA | Service-oriented architecture |
| 11 | XML | Extensible Markup Language |
| 12 | SMTP | Simple Mail Transfer Protocol |
| 13 | SLA | Service Level Agreement |
| 14 | SGN | Secure Government Network |
| 15 | GPC | Government Private Cloud |
| 16 | UAT | User Acceptance Test |
| 17 | NCC | National Call Centre |
| 18 | VM | Virtual Machine |
| 19 | EDMS | Enterprise Document Management System |
| 20 | AD | Microsoft® Active Directory |
| 21 | SOW | Statement of Work |
| 22 | AAA | Authentication, Authorization and Accounting |
| 23 | IBM WSRR | IBM WebSphere Service Registry and Repository |
| 24 | GSB | Government Service Bus (ESB) |
| 25 | ESB | Enterprise Service Bus |
| 26 | NLB | Microsoft Network Load Balancer (software load balancer) |
| 27 | SVN | Apache Subversion Server |
| 28 | Subversion | Apache Subversion Server |
| 28 |  |  |

# GSB Browser

This component is responsible of listing web-services that deployed on Governmental Service Bus, the ESB.

This component allows authenticated users to access the Web-Services list according to their authorization and access rights.

From design perspective this components will built on top of Collaboration Platform and integrated using Collaboration API’s as a new webpages where user can access them from the same Collaboration GUI.

Therefore, User will be authorized in Collaboration standard way using username and password against Active Directory.

**GSB Browser Use-cases and Sequence Diagrams**

**Use-Case Diagram:**

There are three use cases of this component with the following details:

## US-GB-1 Authenticated User List Web-services and Open WSDL/XSD File

## List of Web-Services Sequence Diagram:

### Open WSDL File Sequence Diagram:



### Flow

1. Authenticated User, from top menu in collaboration main web page clicked on button “GSB Browser”
2. System will query group names of the web-services from LDAP using username of the User.
3. System will preview in the same page, a list of web-services for that entity, based on user entity group name along with a hyperlink access.
4. After listing the Web-Services, by clicking on one of the hyperlink of a Web-Service System queries the WDSL file from IBM WSRR system using predefined credential and this call should be a Web-Service call to WSRR API.
5. System shows the file as XML file in the web browser.
6. User-case finish

### Assumptions:

1. User already signed in
2. Organization admin can list and browse web service list.
3. LDAP is accessible using special credentials that has rights to read the groups of web-services.
4. LDAP configured to filter group according to user’s username.
5. WSRR is accessible using special credentials that has rights to read the WSDL files.
6. WSRR is returning latest approved version of WSDL file

## US-GB-2 Organization Admin Assign Permission for GSB Browser Tab

### Flow

1. Organization Admin, from top menu in collaboration main web page clicks on button “GSB Browser”
2. System open two tabs (Web-Services list, Permissions).
3. System admin select to navigate between the tabs.
4. Under permission tab, organization admin can define access to GSB Browser page for the organization members.
5. System admin can change the default access permission to GSB Browser page by unselecting the checkbox of the GSB Browser page access for all members.
6. User-case finish

### Assumptions:

1. User already signed in
2. LDAP configured to filter group according to user’s username
3. Entity admin can list all entity members to change the default GSB browser permissions.

# Email Module

This e-mail module lets you send e-mails with or without templates. E-mail templates can be managed with the possibility of using tokens. With the tokens available in the email template, attributes and references of an object will be filled in automatically and correctly.

## Typical usage scenario

Send emails with an email template system and the possibility of using tokens to handle all your outgoing standard emails.

## Configuration

Download the Mx Model reflection module from the App Store.

Download and configure the Encryption module from the App Store.

Add the snippet Administration to a custom page in a different module.

View the example in the '\_USE\_ME > \_Examples' folder, create a duplicate of it to your own module and make it fit your needs.

Sub\_CreateAndQueueEmail (preferred for normal environments, this will send the email in the background using a scheduled event)

Sub\_CreateAndSendEmail (preferred for sandbox environments, sends an email directly; this approach will block the users flow and doesn't include retry when sending failed).

Post-deployment: After deploying, you have to set up your email settings and insert your own email templates using the newly created navigation items under Administrator.

Go to the 'MxObjects\_Overview' and synchronize the objects. Make sure you do this every time when you have added new objects.

## Dependencies

**Mendix modules**

MXModelReflection module

Encryption module

## Java libraries:

commons-codec-1.10.jar

commons-email-1.4.0.jar

com.sun.mail.javax.mail-1.4.5.jar

## Email Setup (Basic Configuration)

### User Role: SystemAdmin

### Menu System

* **Configuration Email Setup**
* Setting Tab

Email settings page allowing user to setup SMTP configuration to enable email service. To complete email setup user required to fill following information and save.

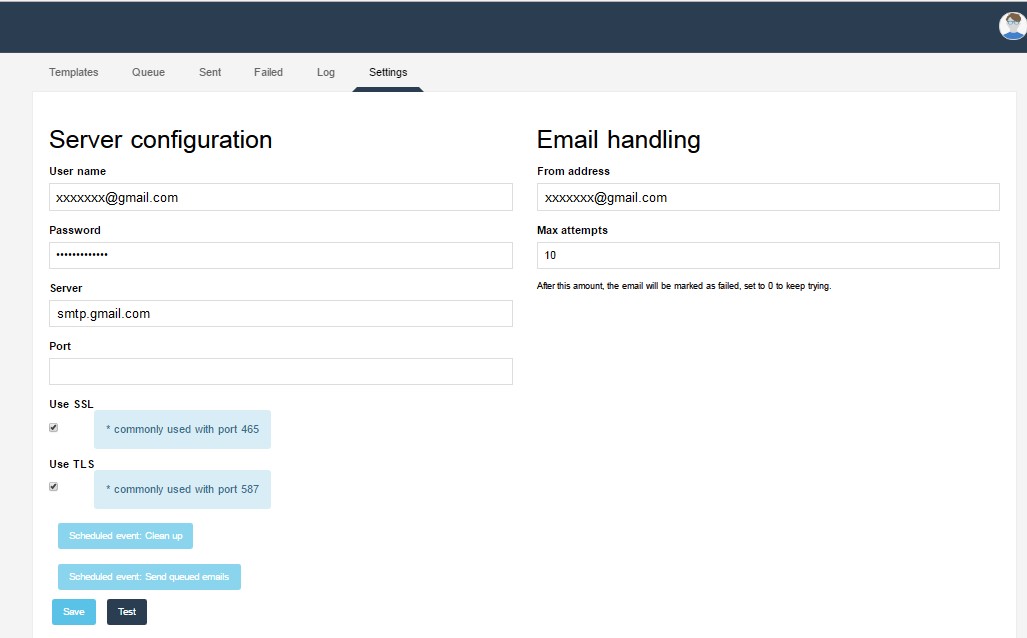
**User Name**: Specify sender user

**Password**: User password

**Server**: Mail server domain name or IP address.

**Port**: Mail server port.

**Use SSL**: Check incase using SSL default 465. **Use TLS**: Check incase using TLS default 587. **From Address**: Sender email address.

**Max Attempts**: No of attempts to be specify.

## Email Configuration Test

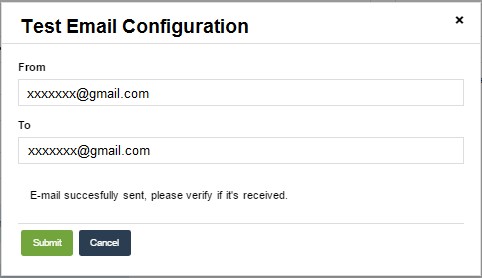
### Menu System

* **Configuration Email Setup**
* Setting Tab

After email service configuration user can test email service by clicking on “Test” button, would open a popup screen allowing user to fill receiver email address

**From**: Display sender email address as configure and Settings screen.

**To**: Specify receiver email address.



**Functional Microflow**: IVK\_SendEmail

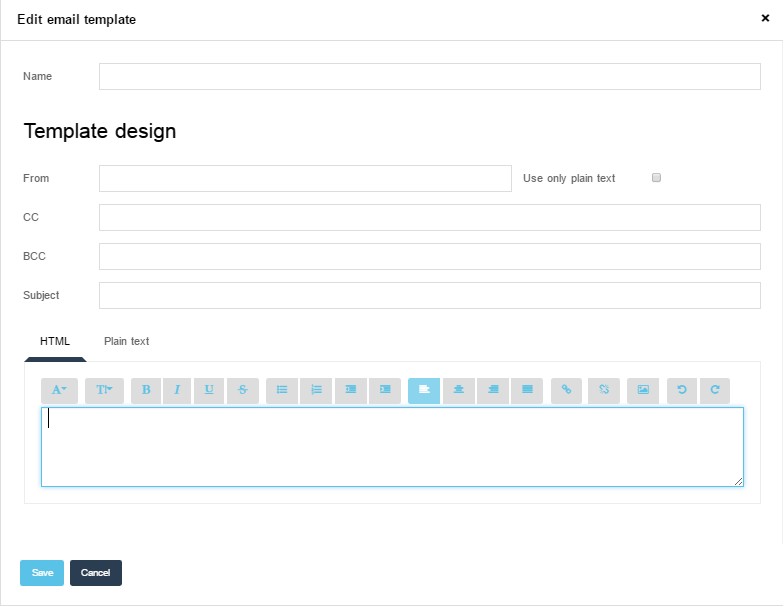
## Email Template

### User Role: SystemAdmin

### Menu System

* **Configuration Email Setup**
* **Templates Tab**

Email templates allowing user to create templates (html or plaint) for email to be send to destination.



# PushSMS Module

This SMS module lets you configure and send text SMS to the destination.

## Typical usage scenario

Send SMS handle all your outgoing standard SMS.

## Configuration

Download the Mx Model reflection module from the App Store.

Download and configure the Encryption module from the App Store.

Add the snippet Administration to a custom page in a different module.

Post-deployment: After deploying, you have to set up your SMS settings.

Go to the 'MxObjects\_Overview' and synchronize the objects. Make sure you do this every time when you have added new objects.

## Dependencies

**Mendix modules**

MXModelReflection module

Encryption module

## Java libraries:

commons-codec-1.10.jar

commons-httpclient-3.1.jar

## SMS Setup (Basic Configuration)

### User Role: SystemAdmin

### Menu System

* **Configuration SMS Setup**
* **Setting Tab**

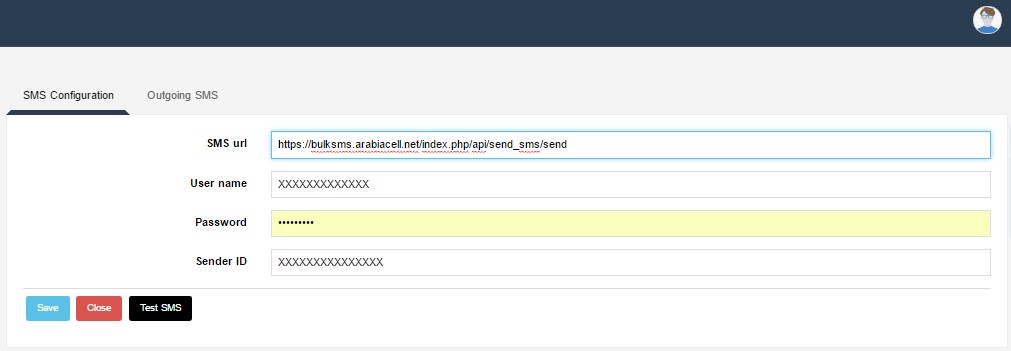
SMS settings page allowing user to setup SMS configuration to enable SMS service in portal. To complete SMS setup user required to fill following information and save.

**SMS URL**: Specify SMS service URL ([https://bulksms.arabiacell.net/index.php/api/send\_sms/send)](https://bulksms.arabiacell.net/index.php/api/send_sms/send)

**User Name**: Specify sender user name.

**Password**: Specify user password.

**Sender Id**: Specify sender Id given by SMS gateway service.



In addition to above configuration, Admin needs to provide application URL to SMS gateway service provider to allow traffic from application URL and register as a SMS consumer.

**Contact**

ArabiaCell Dabouq – King Hussein Business Park, Building 8, 1st Floor.

TEL: +962 6 580 60 80

FAX: +962 6 581 06 89

P.O. Box 17492

Amman-11195

Jordan

## SMS Configuration Test

### User Role: SystemAdmin

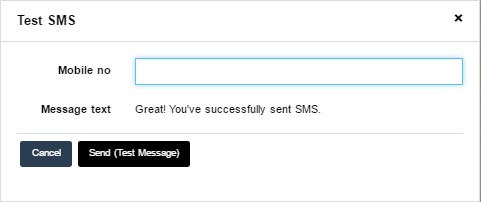
### Menu System

* **Configuration SMS Setup**
* **Setting Tab**

After SMS service configuration user can test SMS service by clicking on “Test SMS” button, would open a popup screen allowing user to fill receiver mobile number.

**Mobile no**: User needs to specify receiver valid mobile number. Mobile number should be in the international format, example 962790000000

**Sample text message**: Some constant message (Non editable)



**Functional Microflow**: Following functional microflow can be used to send SMS.

PushSMS

SendDirectSMS

SendScheduledSMS

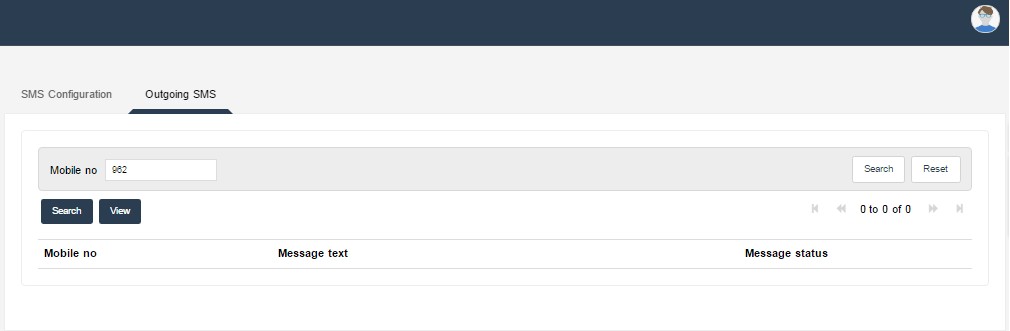
## SMS Overview

### User Role: SystemAdmin

### Menu System

* Configuration SMS Setup
* **Outgoing SMS Tab**

Outgoing SMS screen provides a list of all generated SMS along with message and status. User could filter list of SMS by mobile no as search criteria.



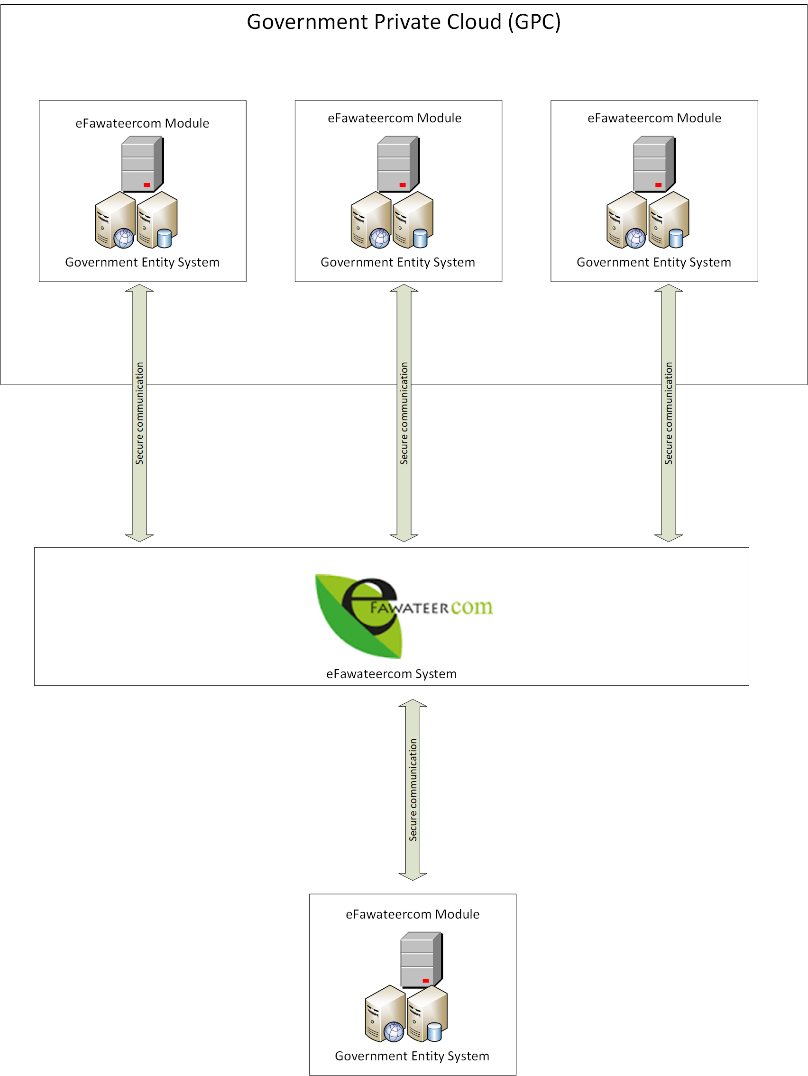
# eFawateer Module

The eFawateer module provides a feature to expose a web service to client to receive and send financial information for the purpose of bills presentment and payment notification information in xml shape.

These requests would be verified and processed in the system, which is responsible for handling all transactions in a secured environment.

This solution provides features for tracing the state of the transactions at all stages.

The main feature of eFawatercom component, that it is reusable, which mean it allows the authorized entities to use efawatercom module by deploying the required files in the Mendix application server and after signing the agreement with MOICT, then they have to configure the required parameters, after that they can use eFawateercom module.



eFawateercom Integration Module

## Typical usage scenario

This module offers client to a comprehensive payment bill representation and update payment notification service to be consumed by them for payment representation and payment notification.

eFawateer module expose a web service providing BillPull and PaymentNotification services.

## Configuration

Import the eFawateer module in Mendix workspace.

Go to the 'MxObjects\_Overview' and synchronize the objects. Make sure you do this every time when you have added new objects.

## Dependencies

NA

## Mendix modules

eFawateer

## Java libraries:

OmnixXmlParser.jar

## eFawateer Configuration (Basic Configuration)

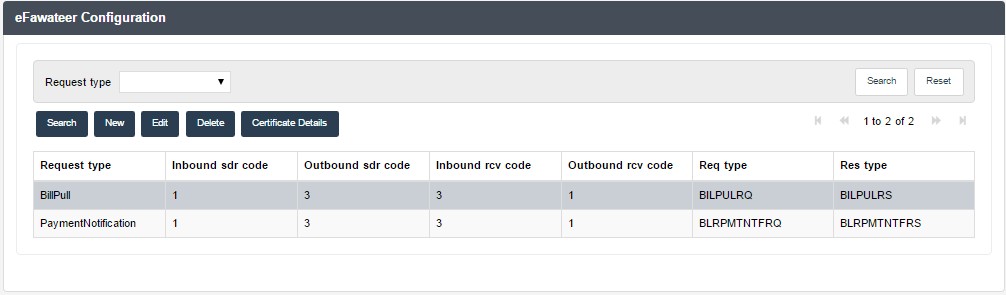
### User Role: SystemAdmin

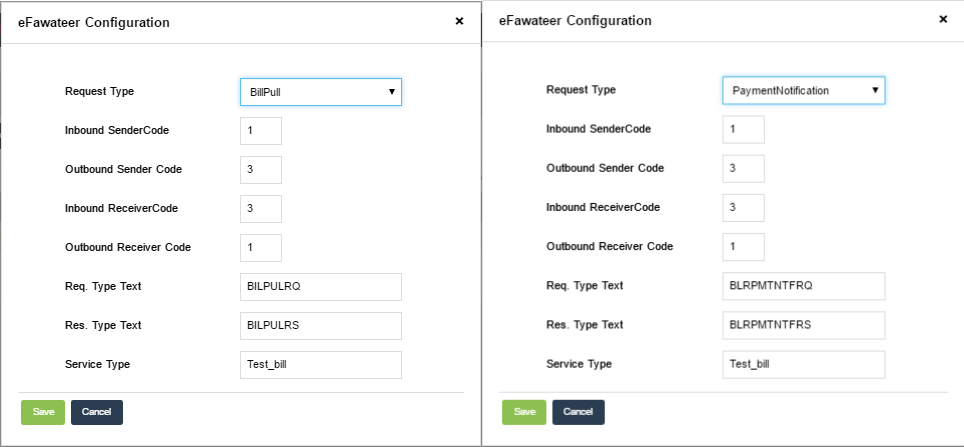
### Menu System

* Configuration eFawateer
* Config eFawateer Configuration (Overview)

eFawateer configuration page allowing user to setup configuration to enable eFawateer (BillPull and

PaymentNotification) service in portal.

To complete configuration setup user required to create request type information and save.

New button to add new request type Edit to update information and Delete to delete request types from configuration, since we are using only two type of transactions BillPull and PaymentNotification we need to add two request types.

Request xml to be verified by above configuration for BillPull and PaymentNotification. Request Type: BillPull/PaymentNotification

Inbound Sender Code: SdrCode to be accepted from request xml. (Provided by eFAWATEERcom) Outbound Sender Code: SdrCode to be filled in response xml. (Provided by eFAWATEERcom) Inbound Receiver Code: RcvCode to be accepted from request xml. (Provided by eFAWATEERcom) Outbound Receiver Code: RcvCode to be filled in response xml. (Provided by eFAWATEERcom) Request Type Text: BILPULRQ/BLRPMTNTFRQ

Response Type Text: BILPULRS/BLRPMTNTFRS

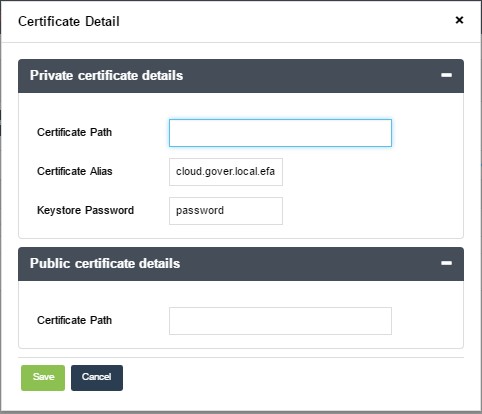
Service Type: Would be provided by eFAWATEERcom

## Certificate Details

All requests would be digitally sign and verify the signature value in xml request and responses for each process based on a pre-defined xpath message body.

eFAWATEERcom will provide Billers with its digital certificate that contains only the public key which is based on PKI infrastructure that supports a key length of 2048 bit and signature hash algorithm SHA-2.

Further details about the certificate requirements will be shared in a separate document.

Certificate details screen allow user to specify private and public certificate path along with credentials.

Private certificate: To sign outbound message body and create digital signature to be sent with xml message, to verify message body with signature.

Certificate Path: Specify path of private certificate.

Certificate Aleas: Alias of private certificate.

Keystore Password: Key store password.

Public certificate: To verify inbound xml message body with public certificate key prior to further processing.

Certificate Path: Specify path of private certificate.

**Functional Microflow**: Following functional microflow can be used to generate bill, Further this bill could be processed by BillPull and PaymentNotification operations by IBiller web service. CreateBilling

## iBiller Web Service

### Operations:

**BillPull:** eFAWATEERcom may ‘pull’ bill data from the Biller site by sending the Bill Pull xml message. The application must respond with the bill data of the requested bill category and service type.

Operation Name: BillPull

Called by: eFAWATEERcom

***BillPullRequest***

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter Name** | **Type** | **Optional** | **Sample** |
| Username | String | Yes |  |
| Password | String | Yes |  |
| RequestMessage | CDATA | No | Figure-1 |
| GUID | String | Yes |  |

***BillPullResponse*** 🡪 BillPullResult

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter Name** | **Type** | **Optional** | **Sample** |
| BillPullResXml | CDATA | No | Figure-2 |

Sample Request:

<soapenv:Envelope xmlns:soapenv=*"http://schemas.xmlsoap.org/soap/envelope/"*

xmlns:ibil=*"http://moict.omnix.ae/IBillerServices/"*>

<soapenv:Header/>

<soapenv:Body>

<ibil:BillPull>

<BillPullRequest>

<Username></Username>

<Password></Password>

<RequestMessage><![CDATA[Request XML]]></RequestMessage>

<GUID></GUID>

</BillPullRequest>

</ibil:BillPull>

</soapenv:Body>

</soapenv:Envelope>

Sample Response:

<soapenv:Envelope xmlns:soapenv=*"http://schemas.xmlsoap.org/soap/envelope/"*

xmlns:ibil=*"http://moict.omnix.ae/IBillerServices/"*>

<soapenv:Header/>

<soapenv:Body>

<ibil:BillPullResponse>

<BillPullResult>

<BillPullResXml><![CDATA[Response XML]]></BillPullResXml>

</BillPullResult>

</ibil:BillPullResponse>

</soapenv:Body>

</soapenv:Envelope>

**ReceivePaymentNotification:** Payment Notification service is used to intimate the application that, the funds has been collected on a given date.

Operation Name: ReceivePaymentNotification

Called by: eFAWATEERcom

ReceivePaymentNotificationRequest

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter Name** | **Type** | **Optional** | **Sample** |
| Username | String | Yes |  |
| Password | String | Yes |  |
| RequestMessage | CDATA | No | Figure-3 |
| GUID | String | Yes |  |

ReceivePaymentNotificationResponse 🡪 ReceivePaymentNotificationResult

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter Name** | **Type** | **Optional** | **Sample** |
| BillPullResXml | CDATA | No | Figure-4 |

Sample Request:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"

xmlns:ibil="http://moict.omnix.ae/IBillerServices/">

<soapenv:Header/>

<soapenv:Body>

<ibil:ReceivePaymentNotification>

<ReceivePaymentNotificationRequest>

<XmlMessage><![CDATA[Request XML]]></XmlMessage>

<Username></Username>

<Password></Password>

<GUID></GUID>

</ReceivePaymentNotificationRequest>

</ibil:ReceivePaymentNotification>

</soapenv:Body>

</soapenv:Envelope>

Sample Response:

<soapenv:Envelope xmlns:soapenv=*"http://schemas.xmlsoap.org/soap/envelope/"*

xmlns:ibil=*"http://moict.omnix.ae/IBillerServices/"*>

<soapenv:Header />

<soapenv:Body>

<ibil:ReceivePaymentNotificationResponse>

<ReceivePaymentNotificationResult> <ReceivePaymentNotificationResXml>

<![CDATA[Response XML]]>

</ReceivePaymentNotificationResXml>

</ReceivePaymentNotificationResult>

</ibil:ReceivePaymentNotificationResponse>

</soapenv:Body>

</soapenv:Envelope>

**XML Request Response:**

****

**Fiure-1**

****

**Figure-2**

****

**Figure-3**

****

**Figure-4**

### WSDL:

<wsdl:definitions name=*"IBillerServices"* targetNamespace=[*"http://moict.omnix.ae/IBillerServices/"* xmlns:wsdl=](http://moict.omnix.ae/IBillerServices/)[*"http://schemas.xmlsoap.org/wsdl/"* xmlns:xsd=](http://schemas.xmlsoap.org/wsdl/)[*"http://www.w3.org/2001/XMLSchema"* xmlns:soap=](http://www.w3.org/2001/XMLSchema)[*"http://schemas.xmlsoap.org/wsdl/soap/"* xmlns:http=](http://schemas.xmlsoap.org/wsdl/soap/)[*"http://schemas.xmlsoap.org/wsdl/http/"* xmlns:tns=](http://schemas.xmlsoap.org/wsdl/http/)[*"http://moict.omnix.ae/IBillerServices/"*>](http://moict.omnix.ae/IBillerServices/)

<wsdl:types>

<xsd:schema targetNamespace=[*"http://moict.omnix.ae/IBillerServices/"*>](http://moict.omnix.ae/IBillerServices/)

<xsd:element name=*"BillPull"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"BillPullRequest"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"Username"* minOccurs=*"0"*>

<xsd:simpleType>

<xsd:restriction base=*"xsd:string"*>

<xsd:maxLength value=*"200"* />

</xsd:restriction>

</xsd:simpleType>

</xsd:element>

<xsd:element name=*"Password"* minOccurs=*"0"*>

<xsd:simpleType>

<xsd:restriction base=*"xsd:string"*>

<xsd:maxLength value=*"200"* />

</xsd:restriction>

</xsd:simpleType>

</xsd:element>

<xsd:element name=*"RequestMessage"* type=*"xsd:string"* />

<xsd:element name=*"GUID"* minOccurs=*"0"*>

<xsd:simpleType>

<xsd:restriction base=*"xsd:string"*>

<xsd:maxLength value=*"200"* />

</xsd:restriction>

</xsd:simpleType>

</xsd:element>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:element name=*"BillPullResponse"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"BillPullResult"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"BillPullResXml"* type=*"xsd:string"* />

</xsd:sequence>

</xsd:complexType>

</xsd:element>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:element name=*"ReceivePaymentNotification"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"ReceivePaymentNotificationRequest"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"XmlMessage"* type=*"xsd:string"* />

<xsd:element name=*"Username"* minOccurs=*"0"*>

<xsd:simpleType>

<xsd:restriction base=*"xsd:string"*>

<xsd:maxLength value=*"200"* />

</xsd:restriction>

</xsd:simpleType>

</xsd:element>

<xsd:element name=*"Password"* minOccurs=*"0"*>

<xsd:simpleType>

<xsd:restriction base=*"xsd:string"*>

<xsd:maxLength value=*"200"* />

</xsd:restriction>

</xsd:simpleType>

</xsd:element>

<xsd:element name=*"GUID"* minOccurs=*"0"*>

<xsd:simpleType>

<xsd:restriction base=*"xsd:string"*>

<xsd:maxLength value=*"200"* />

</xsd:restriction>

</xsd:simpleType>

</xsd:element>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:element name=*"ReceivePaymentNotificationResponse"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"ReceivePaymentNotificationResult"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"ReceivePaymentNotificationResXml"*

type=*"xsd:string"* />

</xsd:sequence>

</xsd:complexType>

</xsd:element>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

</xsd:schema>

</wsdl:types>

<wsdl:message name=*"BillPullRequest"*>

<wsdl:part name=*"parameters"* element=*"tns:BillPull"* />

</wsdl:message>

<wsdl:message name=*"BillPullResponse"*>

<wsdl:part name=*"result"* element=*"tns:BillPullResponse"* />

</wsdl:message>

<wsdl:message name=*"ReceivePaymentNotificationRequest"*>

<wsdl:part name=*"parameters"* element=*"tns:ReceivePaymentNotification"* />

</wsdl:message>

<wsdl:message name=*"ReceivePaymentNotificationResponse"*>

<wsdl:part name=*"result"* element=*"tns:ReceivePaymentNotificationResponse"* />

</wsdl:message>

<wsdl:portType name=*"IBillerServicesPortType"*>

<wsdl:operation name=*"BillPull"*>

<wsdl:documentation></wsdl:documentation>

<wsdl:input message=*"tns:BillPullRequest"* />

<wsdl:output message=*"tns:BillPullResponse"* />

</wsdl:operation>

<wsdl:operation name=*"ReceivePaymentNotification"*>

<wsdl:documentation></wsdl:documentation>

<wsdl:input message=*"tns:ReceivePaymentNotificationRequest"* />

<wsdl:output message=*"tns:ReceivePaymentNotificationResponse"* />

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name=*"IBillerServicesSoap"* type=*"tns:IBillerServicesPortType"*>

<soap:binding style=*"document"* transport= [*"http://schemas.xmlsoap.org/soap/http"* />](http://schemas.xmlsoap.org/soap/http)

<wsdl:operation name=*"BillPull"*>

<soap:operation soapAction=[*"http://moict.omnix.ae/IBillerServices/BillPull"*](http://moict.omnix.ae/IBillerServices/BillPull)

/>

<wsdl:input>

<soap:body parts=*"parameters"* use=*"literal"* />

</wsdl:input>

<wsdl:output>

<soap:body use=*"literal"* />

</wsdl:output>

</wsdl:operation>

<wsdl:operation name=*"ReceivePaymentNotification"*>

<soap:operation soapAction=[*"http://moict.omnix.ae/IBillerServices/ReceivePaymentNotification"*](http://moict.omnix.ae/IBillerServices/ReceivePaymentNotification)/>

<wsdl:input>

<soap:body parts=*"parameters"* use=*"literal"* />

</wsdl:input>

<wsdl:output>

<soap:body use=*"literal"* />

</wsdl:output>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name=*"IBillerServices"*>

<wsdl:port name=*"IBillerServicesPort"* binding=*"tns:IBillerServicesSoap"*>

<soap:address location=[*"http://localhost:8081/ws/IBillerServices"* />](http://localhost:8081/ws/IBillerServices)

</wsdl:port>

</wsdl:service>

</wsdl:definitions>

## Digital Signature and Message Signing in eFAWATEERcom

## 1. Certificate Generation

1.1 MFEP will install its own private key on the presentation servers

1.2 MFEP will send the public key for each participant

1.3 Each participant should generate certificate by contacting MFEP support team

1.4 MFEP support team will use the PKI tools at EMP to generate the required certificate

1.5 MFEP team will install the new participant certificate-public key on the presentation servers

1.6 MFEP team will send the new participant certificate with private key to the participant

How to create certificate request in Windows 2012 Server: (Check Appendix A)

## 2. Certificate Authority's Root Certificate Installation

2.1 “EMPJ-ORCA\_EMPJ-RCA” needs to be imported in “Trusted Root Certificate Authorities”

2.2 “EMPJ-ICA01.vjcs.com.jo\_EMPJ-CA01” needs to be imported in Intermediate Certificate

Authorities

2.3 “EMPJ-ICA02.vjcs.com.jo\_EMPJ-CA02” needs to be imported in Intermediate Certificate

Authorities

## How to install Root Certificate

From the Start menu, click Run. Type "mmc" in the text box and click OK. An MMC snap-in

Console window launches.

1. On the Console menu, click Add/Remove Snap-in.

2. Click Add to add a snap-in to the current console.

3. Select Certificates in the Snap-in list, click Add, and click Close.

4. Click OK to close the Add/Remove Snap-in dialog box. The Certificates directory is now added to the MMC console.

Note: If you are on a domain controller, when you select Certificates, a dialog box appears asking you whether you would like to manage certificates for my user account, Service account, or Computer account. For this scenario, select my user account, click Finish, and continue.

5. On the Console menu, click Save as, and type Certificates as the file name of this console. Click Save. To access the Certificates console in the future, click Start, point to Programs, point to Administrative Tools, and then click Certificates.

## Installing the participant's Certificate

3.1 The participant should install the certificate and start using it for signing

3.2 The participant should install the certificate to decrypt incoming traffic

How install/complete the digital certificate request in Windows 2012: (Check Appendix A)

### Request Signing

1. Each requester should singe the message body by his private key

2. The receiver should verify the signature of incoming traffic by using the requester public key

### Digital Signature Algorithm:

Message body of the XML element (as specified in the Con2Us document) should be done using

SHA-256 (SHA-2) hashing algorithm.

# E-Services List (Portal) Module

This module contains a web service that is published to return all the services’ details defined in your application.

## Typical usage scenario

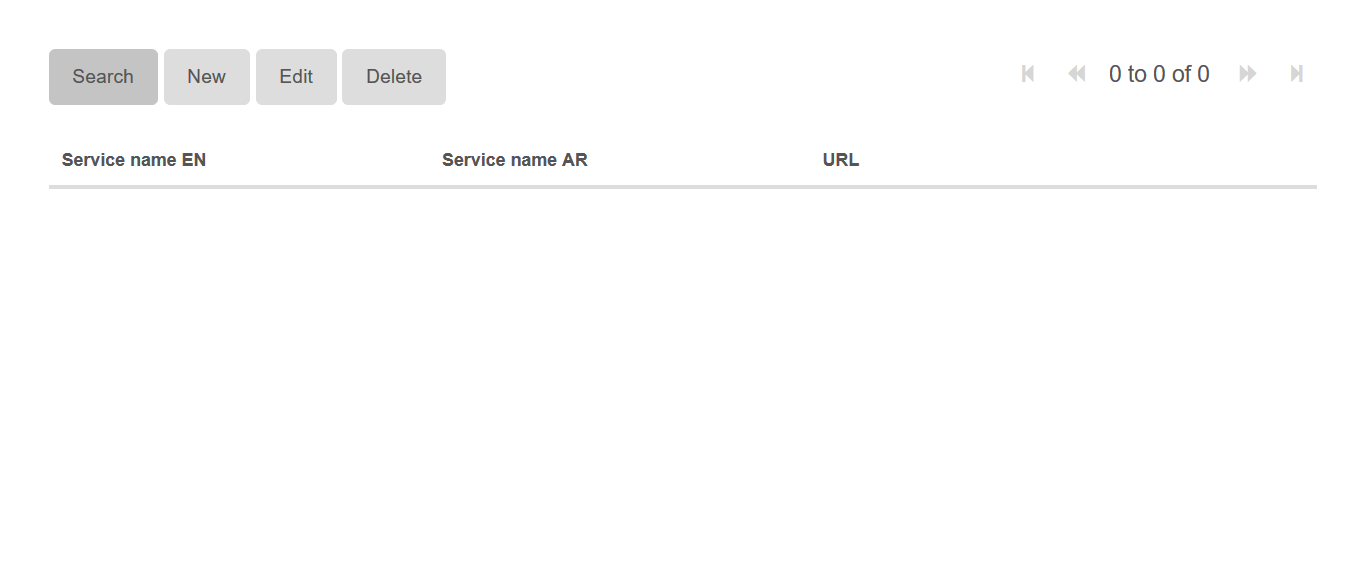
To enrich the eGovernment Portal with the eService list of each entity, this module contains a web service exposed to be consumed by the Portal. This module will also provide additional screen to view all the services listed by the entity which they will use in their system and it can be accessed from outside the Mendix collaboration web by different end point.

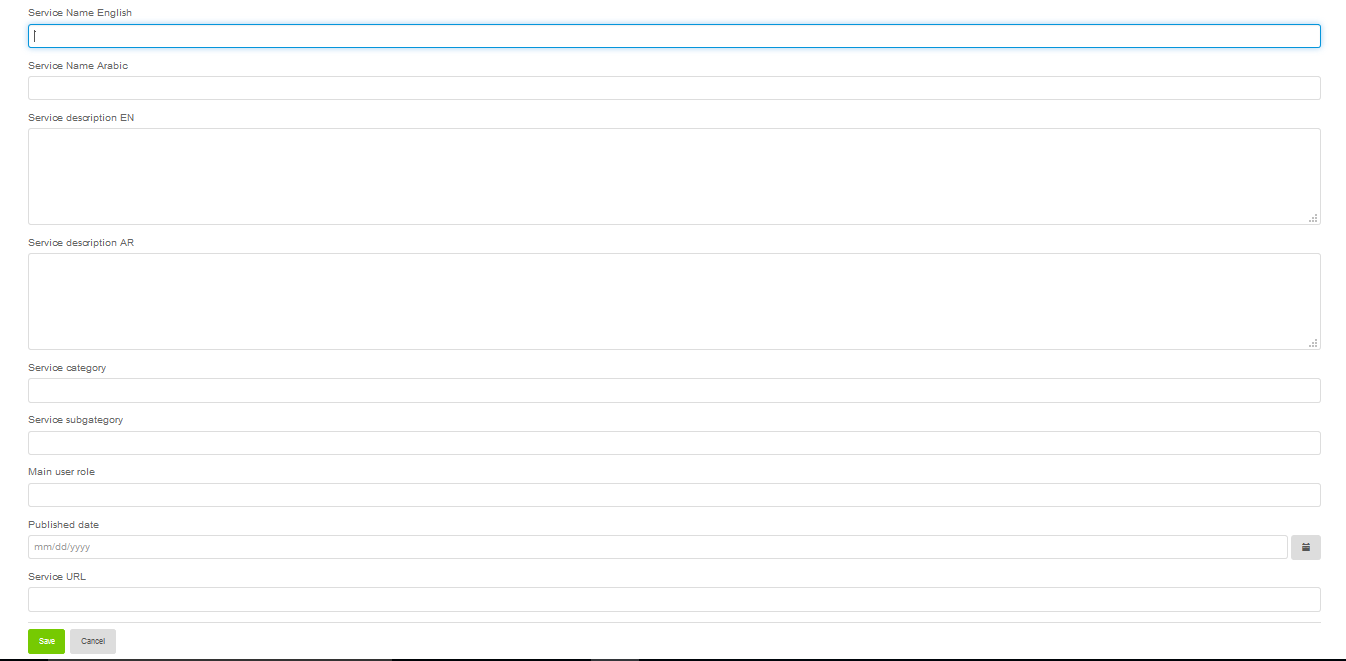
## Configuration

* Assign the ‘Admin’ Module Role to the user role that shall have the permission to define the services in the application.
* Assign the ‘User’ Module Role to the user role that will be consuming the web service.
* Link the form (page) ‘Service\_Overview’ in the ‘\_USE\_ME’ Folder to the navigation menu

## Services Setup (Basic Configuration)

* Click the ‘New’ button to add each service details through the newly created navigation item.
* Click the ‘Edit’ button to update the selected service details
* Click ‘Save’ button to save the entered details
* Click ‘Cancel’ button to cancel the changes
* Click the ‘Delete’ button to delete the selected service







# Appendix A

How to create Digital Certificate request in Windows 2012 Server:

Perquisites:

- Windows Server 2012

- IIS

Steps:

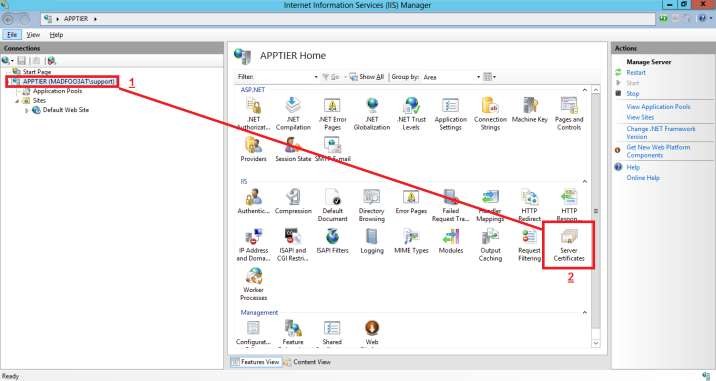
1. Press the Windows key + Q to show the Metro search screen

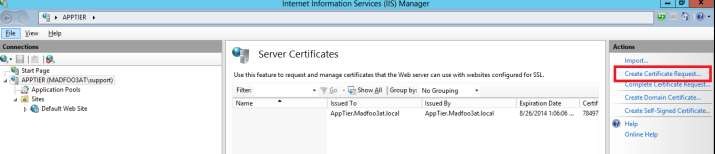
2. Type "IIS" in the App search box

3. Open "Internet Information Services (IIS) Manager"



4. Select <servername> then double-click "Server Certificates"



5. Choose "Create Certificate Request" from the "Actions" menu on the right.

6. In the "Request Certificate" Dialog fill in the following information: Common Name: Participant's Name

Organization: Participant's Organization Name Organizational Unit: Participant's Unit Name City/locality: Amman

State/Province: Amman

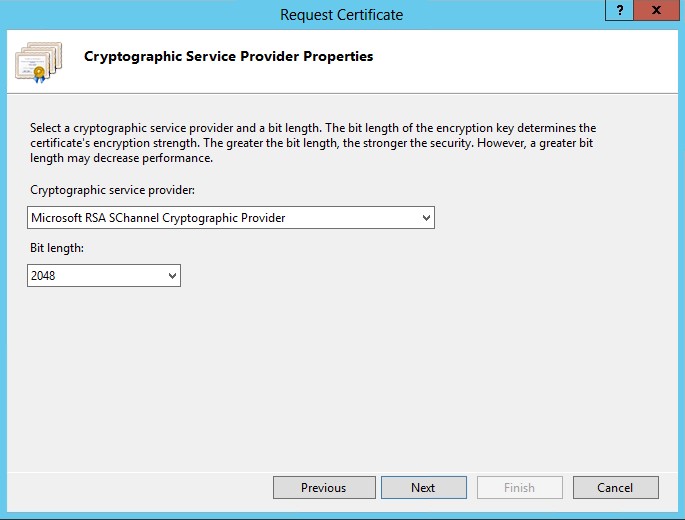
Country/region: JO And press "Next"

7. From the next window choose the "Cryptographic service provider": Microsoft RSA SChannel Cryptographic Provider

And "Bit Length":

2048

And press "Next"



8. From the next window, choose the output path and file name of the Digital Certificate request.

This could be something like "C:\Certificate\MyRequest.txt"

9. Send the request file to [support@madfoo3at.com](mailto:support@madfoo3at.com)

How to complete Digital Certificate Request in Windows Server 2012:

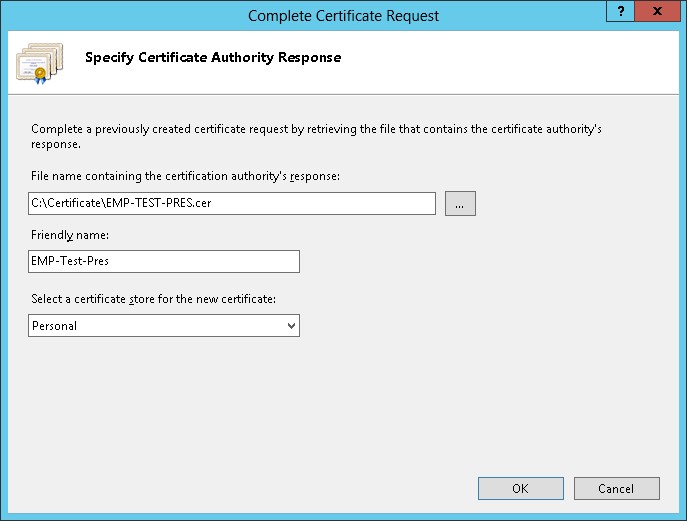
1-4. Steps 1 to 4 are similar to the above steps.

5. Press the "Complete Certificate Request" from the top right corner.

6. Fill in the following fields in the window that pops and select OK: File Name: Location of the Digital Certificate.

Friendly Name: An alias for the Digital Certificate.

Select a Certificate Store: Location to store the Certificate

7. After the certificate is created, double click the new certificate and make sure the certificate/path is installed correctly:

