

Electronic Postmark™ Service SDK V1

USPS® Secure Digital Platform Application Programming Interface User's Guide

Version 1.1H (03/08/2017)



Table of Contents

1. PURPOSE	3
2. ELECTRONIC POSTMARK SYSTEM OVERVIEW	4
2.1 EPM SERVICE OVERVIEW	4
3. USE CASES	5
3.1 GENERAL ELECTRONIC DATA POSTMARKING	5
3.2 EPM GENERATION	5
3.2.1 Process Flow	5
3.3 EPM VALIDATION	6
3.3.1 Process Flow	6
3.4 TRUSTMARK VALIDATION	7
3.4.1 Process Flow	7
3.5 QUERY EPM VALIDATION RESULTS	8
3.5.1 Process Flow	8
4. ELECTRONIC POSTMARK (EPM) DATA STRUCTURES AND PROCESSING	9
4.1 EPM DATA STRUCTURE AND FORMAT	9
4.1.1 Example EPM XML response	9
4.1.2 Electronic Postmark	11
4.1.3 EPM Trustmark image	14
5. WEB SERVICES	15
5.1 SOFTWARE DEVELOPMENT KIT (SDK)	15
5.1.1 Requirements	15
5.2 EPM SERVICE SECURITY	16
5.2.1 Using the keystore with the Java SDK	16
5.2.2 Using the certificate with the C# / .net SDK	17
5.3 SDK INTERFACE	19
5.3.1 Process to use SDK	21
5.4 TROUBLE SHOOTING COMMON ERROR CONDITIONS	25
5.4.1 Issues with your certificate	25
5.4.2 Issues with your message	26
5.4.3 Validation error for EPM service V1	26
5.5 USING THE EPM WEB SERVICE	27
5.5.1 Customer Registration	27
6. APPENDIX A – EPM XML SCHEMAS	28
6.1 LINKS TO WSDL AND SCHEMA LOCATIONS	28
7. APPENDIX B - SETUP THE SAMPLE JAVA CLIENT USING THE SDK IN ECLIPSE	29
7.1 STEPS	29
8. APPENDIX C – SETUP THE SAMPLE .NET CLIENT USING VISUAL STUDIO 2015	34
8.1 STEPS	34

1. Purpose

The purpose of this Interface Control Document is to provide a detailed description of the interface between the USPS EPM™ web service and client systems. This document describes the methods to access EPM functionality as well as the general behavior of the system.

2. Electronic Postmark System Overview

The Electronic Postmark System provides the means to digitally sign and timestamp various types of electronic data through an official USPS authority. This process results in the creation of a USPS Electronic Postmark (EPM). While many forms of secure communication provide security for electronic data in transit, the EPM provides authenticity, data integrity, and non-repudiation for electronic data in transit and at rest. This allows the data to be stored and/or transmitted with assurance of integrity and a verifiable audit trail.

2.1 EPM Service Overview

The EPM service supports both EPM generation and EPM validation operations. An EPM can generally be applied to most forms of electronic data. The EPM can be used to verify the integrity and authenticity of electronic data. When generating an EPM, EPM data is returned to the EPM service client. When validating an EPM, an indicator of EPM validity is returned to the EPM service client.

3. Use Cases

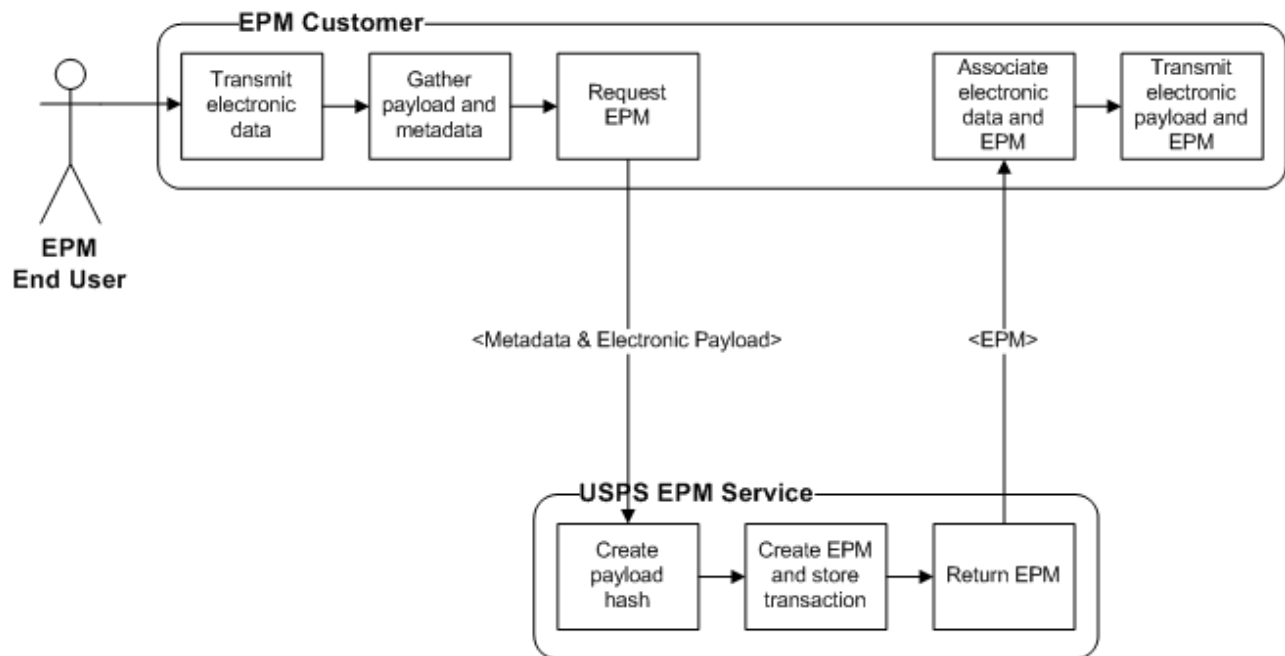
3.1 General Electronic Data Postmarking

A USPS EPM customer provides its end users with the ability to store and/or transmit documents and other forms of electronic data. The electronic data may be sensitive and/or valuable to the end users so the customer needs to be able to ensure non-repudiation and authenticity as part of its storage and transmission functionality. In order to ensure non-repudiation and authenticity, an EPM must be generated for the electronic data through a USPS approved authority.

3.2 EPM Generation

The customer utilizes a service API that accepts the metadata and electronic payload as inputs in order to generate the EPM. Once inputs are received, the system generates a digital hash over the payload data, combines the hash, metadata, and a timestamp into an EPM data structure, then digitally signs the EPM data structure and stores it for audit purposes. The EPM data structure is then returned to the customer so that the customer can save the EPM to a local file or data store along with the original data and/or transmit both to a 3rd party. If the customer submits invalid or insufficient data, the EPM is not generated and the system returns an error to the customer.

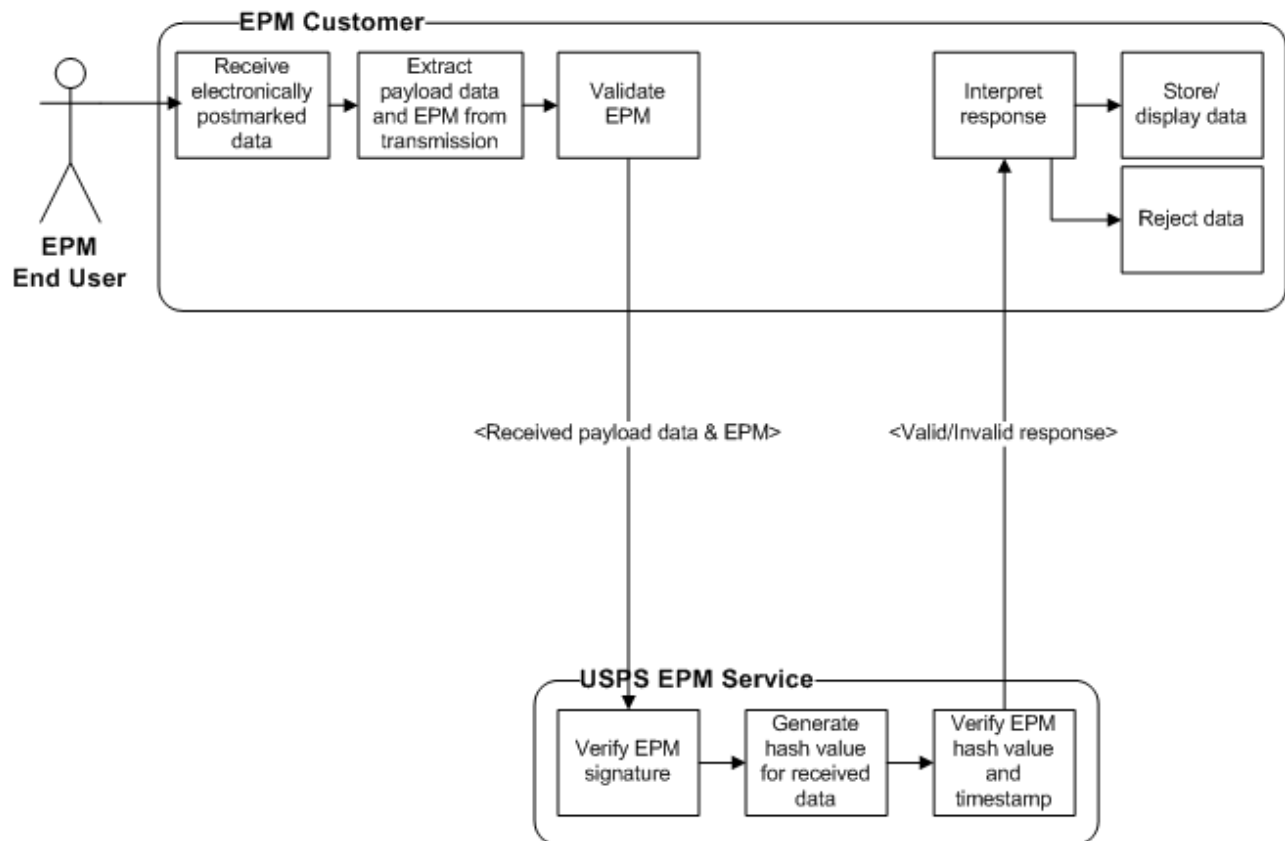
3.2.1 Process Flow



3.3 EPM validation

A customer may validate data against it's corresponding EPM in order to ensure that the data has not been altered in any way. The customer utilizes a service API that accepts payload data and an EPM string as inputs. An indicator of EPM validity is then returned to the customer. The system returns an error if the customer submits invalid or insufficient data.

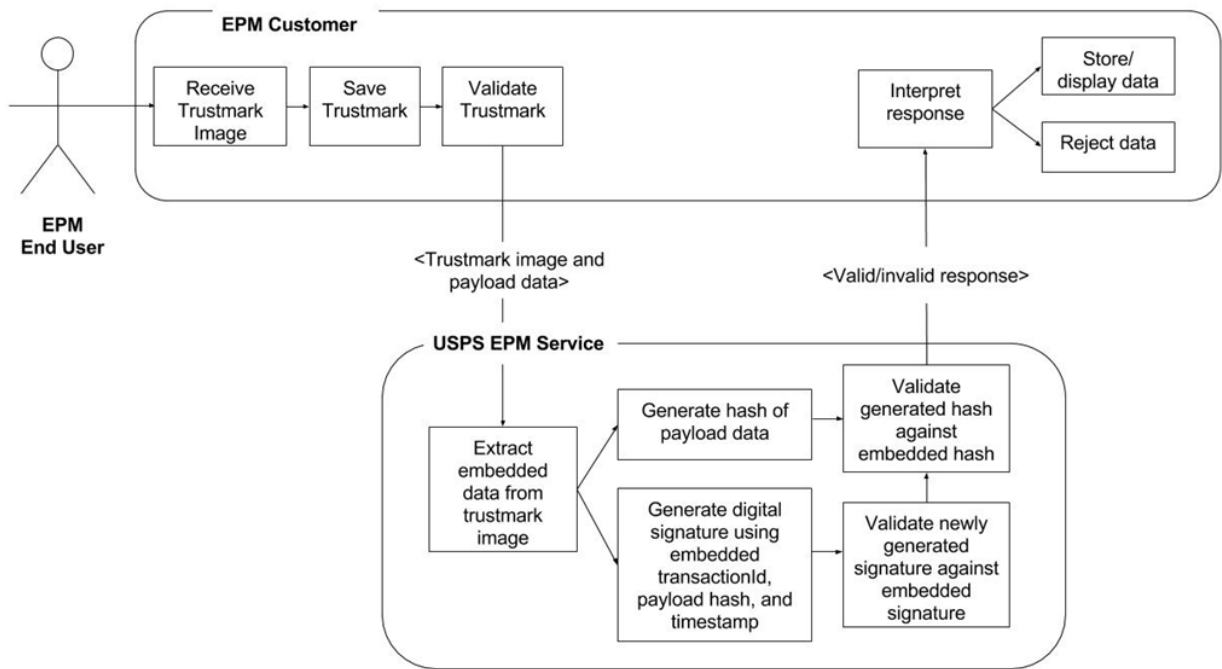
3.3.1 Process Flow



3.4 Trustmark validation

The EPM recipient also receives a trustmark image. Like the EPM, the trustmark image may also be used to validate that its corresponding data has not been altered in any way. The customer utilizes a service API that accepts payload data and the trustmark image as inputs. An indicator of trustmark validity is then returned to the customer. The system returns an error if the customer submits invalid or insufficient data.

3.4.1 Process Flow

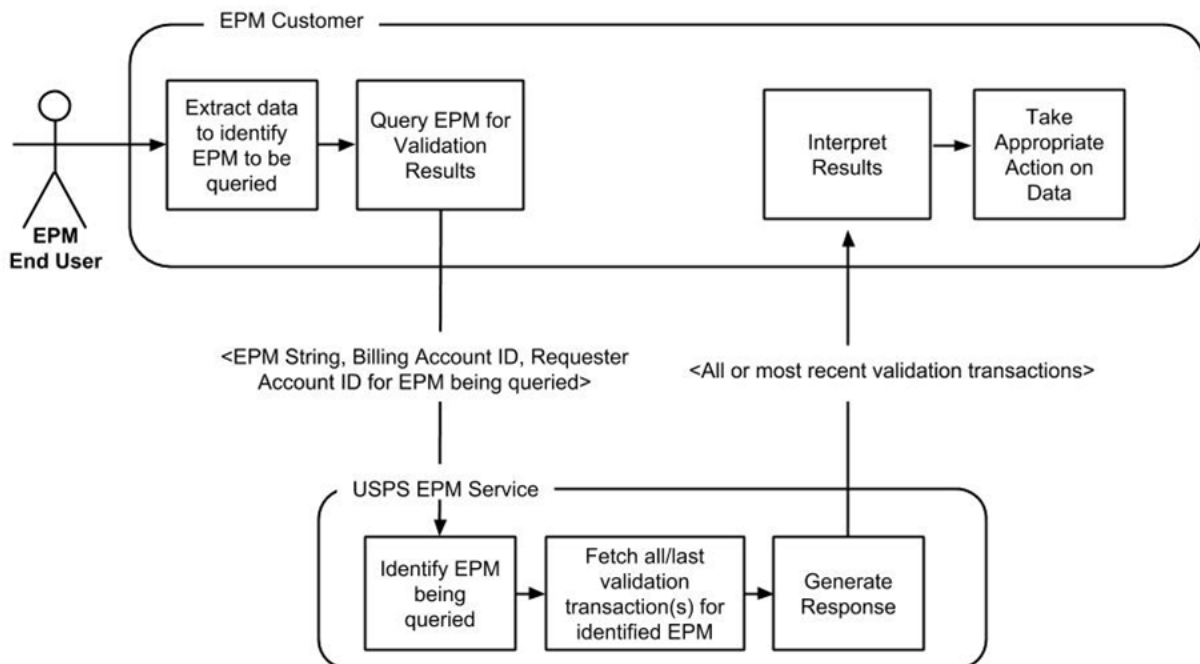


3.5 Query EPM validation results

A customer may query for validation requests made for an EPM previously generated in order to confirm that the recipient received a valid EPM and was able to successfully validate it. The customer utilizes a service API that accepts transaction number, billing account ID and requester account ID as inputs. The system uses these inputs to fetch all related validation transactions and then depending upon the operation used, returns either the most recent validation transaction, or all of the validation transactions for the provided EPM. The system returns an error if the customer submits invalid or insufficient data.

Note: the validations results are specific to the EPM. If multiple EPMs are generated for a particular file the validations list will only return validations for the specific EPM that is sent.

3.5.1 Process Flow



4. Electronic Postmark (EPM) Data Structures and Processing

An EPM is generated to guarantee non-repudiation and authenticity of a set of data, referred to as the EPM Payload. The EPM Payload represents almost any block of electronic data that can be expressed as a finite-size array of bytes. EPM Payload data may exist in almost any electronic form, including structured text, documents, images, binary executables, and compressed archive files.

4.1 EPM Data Structure and Format

The EPM is a set of XML structured/formatted data defined in the USPS EPM XML Schema. At a high level, the EPM data structure consists of the following components:

- EPM attributes
- System audit information
- EPM payload metadata

An EPM does not contain actual EPM Payload data. An EPM exists external to the EPM Payload for which it was generated. In order to ensure that an EPM is associated to the correct EPM Payload, the EPM customer must make sure to store and/or transmit the EPM Payload data and EPM in such a way that the relationship between the two is maintained. See the EPM verification section below for more details on the standards for associating and transmitting EPM Payload data and EPMs as files.

IMPORTANT NOTE: In order to ensure the continuing validity of an EPM, both the EPM and the EPM Payload data must be stored and/or transmitted without modification. The EPM Payload data cannot be modified in any way that might alter the raw binary contents of the EPM Payload. For example, changing even a single character in a text file or a single property of an image or document will invalidate the EPM. Also, if the EPM Payload is a file, then the filename can only be altered if the filename is not included as part of the file contents (as some applications include the filename in properties which are contained in the contents of the file). Similarly, the characters and formatting of an EPM cannot be altered in any way from the point where the EPM is returned in a web service response.

4.1.1 Example EPM XML response

```
<ElectronicPostmark xmlns="http://des.usps.com/model/shared/v2"
xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"
payloadHashValue="Oum99SI7NC0jSWHPVtfTY4MtqzgPLV3TCb2dmXMHJWOnO8UccVB38jn6XHWLpIK
vJe++0o3tDIPg4wTxKpxpJQ==" timestamp="2014-09-12T15:27:49.539Z" transactionNumber="7">
  <PayloadMetadata epmPayloadCategory="HEALTH_CARE">
    <HealthcareMetadata acceptAcknowledgmentType="Q"
alternateCharacterSetHandlingScheme="V" applicationAcknowledgmentType="R" characterSet="T"
continuationPointer="P" countryCode="US" encodingCharacters="?" fieldSeparator="|"
messageControlId="L" messageDateTime="I" messageProfileId="W" messageType="K"
principalLanguageOfMessage="U" processingId="M" receivingApplication="E" receivingFacility="F"
receivingNetworkAddress="H" receivingResponsibleOrganization="G" security="J" sendingApplication="A"
sendingFacility="B" sendingNetworkAddress="D" sendingResponsibleOrganization="C"
sequenceNumber="O" versionId="N"/>
    <MailPieceMetadata mailerId="123456" providerId="123456">
      <Origin emailAddress="edsmith@mail.com" ipV4Address="192.168.0.1"
ipV6Address="FE80:0000:0000:0000:0202:B3FF:FE1E:8329" macAddress="01:23:45:67:89:ab"
```

```

originatorId="1" personFirstName="Ed" personLastName="Smith" personMiddleName="A"
phoneNumber="555-555-4444" smsNumber="555-555-4444">
    <Address addressLocation="123 Main Street" addressType="DOMESTIC"
attentionLine="Ed" countryCode="US" locality="Anywhere" postalCode="29464" stateOrRegion="SC"/>
    </Origin>
    <Recipient emailAddress="jdoe@mail.com" personFirstName="John"
personLastName="Doe" personMiddleName="A" smsNumber="1231231234">
        <Address addressLocation="2387 Clements Ferry Road"
addressType="DOMESTIC" attentionLine="John" countryCode="US" locality="Charleston"
postalCode="29492" stateOrRegion="SC"/>
    </Recipient>
</MailPieceMetadata>
</PayloadMetadata>
<SystemAuditInformation desCategoryCode="A" desFederalAgencyCode="123"
desFederalAgencySubCode="45678" desSerialNumber="AB-12345-CD-67890"
desSoftwareVersion="1.0"/>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315#WithComments"/>
            <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-
sha512"/>
            <Reference URI="">
                <Transforms>
                    <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                </Transforms>
                <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
                    <DigestValue>9pUCGTB2wEgQUXBY9z/eQN6Crq9xWyqejXztXEEOAPyzZ2qMMJ8fiFIISXfGFtgJZ
0fRljTpNeBvaE6sCQOGIA==</DigestValue>
                </Reference>
            </SignedInfo>
            <SignatureValue>URaqC61PQj9R4xLSAwskbDn5aeck4EZH5HjqPSBEcOwC8FjjsXxw9K7WUfRM
TrfKthWbdM3qhO0cUWdW5Xlt/pTBcVK74qsyOPYfpX6sbUz0HahDJ+M2RP8dYybX0KfJ1UMe6Vk5hqbau
pL3iMgfRapr8aysDH1735563bRGI7rE/ZExHDV3m8Q2m80amzlcAgJWcTjxCiELgrxpJ1+FFBSaBXXR0/Tz0
4XjPNCcRaJ8MzjRK0Y0yiMLtibtyRfkjREbmQl7jRdQJnlmLk7vL41i6/NecyCz5/0c5zl60oi1KK2i2v/8hDZQTj
2kQaiYxV6L2AN64y1XUOeKjIHrpU3d8WdkJIBUwqTkiEoQChwrQ479gwZB8vTAzhs5dP2WogUQhtfzb+UD
Ba+gS3EqrJR6xuHmqdgM747DqKtfnN7dGYnkhGYt9pbVH03HEN+YcHDnrPSEJYuPC39pi47u82unznDfZ9
skjhlx2YT3X8qdsIBUbiWiHo8D4pk2BTKg8+ofirc6EBrd+bvPQ9ubbRMlp8eiSKJ2RoSpYRPyHy/UDdtz0ro
mYXSBjCjdNV3tRi0mx5O3NSLLj3z7j1IDECXx6yKlt6LiHIVyxJ2ltmxZroZ7/VOH9cG/KET1/Enb1oQhbMCb3
19HUjB78fSMq6VEdZKU0uokOlV9G6F6sU=</SignatureValue>
        <KeyInfo>
            <X509Data>
                <X509IssuerSerial>
                    <X509IssuerName>CN=Electronic Postmark Authority,OU=Digital
Evidencing System,O=United States Postal Service,L=Washington,ST=DC,C=US</X509IssuerName>
                    <X509SerialNumber>1927522781</X509SerialNumber>
                </X509IssuerSerial>
            </X509Data>
        </KeyInfo>
    </Signature>
</ElectronicPostmark>

```

4.1.2 Electronic Postmark

The fields included in the EPM XML data structure are outlined below. For more information on exact data type and format restrictions, data content limitations or enumeration values, and requirements on the order of appearance for each of the attributes and elements listed below, please refer to the ElectronicPostmark definition in the USPS EPM XML Schema document.

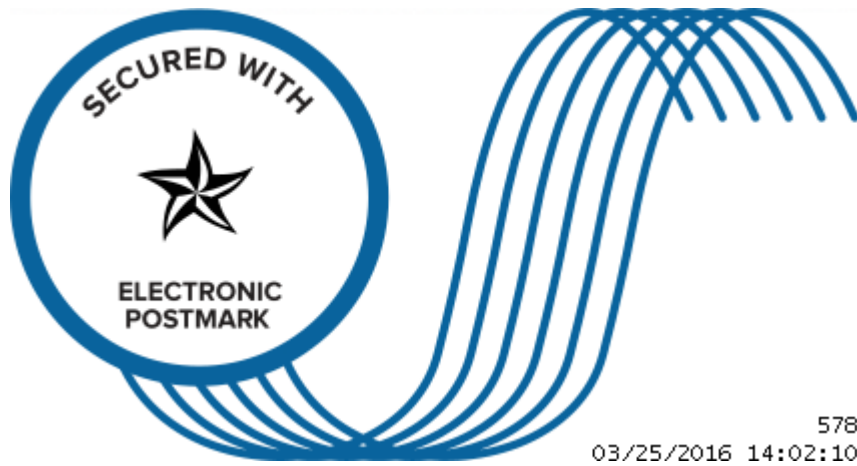
Field Name	Attribute /Element	Required	Description
Element: ElectronicPostmark			
payloadHashValue	Attribute	Y	The EPM Payload data hash value is a SHA512 cryptographic hash that is generated over the base 64 encoded EPM Payload data. Consumers of the EPM can use this value to determine whether or not the payload contents has changed since the EPM was generated, ensuring non-repudiation.
timestamp	Attribute	Y	The EPM timestamp indicates the date and time (in UTC format) when the EPM was created.
transactionNumber	Attribute	Y	The current count of the number of transactions processed by a particular EPM customer
testTransaction	Attribute	N	Indicates whether or not the current EPM transaction is to be considered a test transaction for payment purposes. Will only be present in the EPM response if the current transaction is a test transaction.
Signature	Element	Y	The EPM digital signature is generated in the form of an industry standard XML Digital Signature based on the W3C XML Digital Signature recommendation found here: http://www.w3.org/TR/xmlsig-core/ . The digital signature is generated over the entire EPM and its contents and is used to provide non-

Field Name	Attribute /Element	Required	Description
			repudiation and authenticity of the EPM and the data it contains.
PayloadMetadata	Element	Y	The EPM payload metadata consists of various attributes about the data set for which the EPM is being generated. This provides context to the payload data, to the parties involved with the payload data creation, and possibly to the parties involved with its use. EPM consumers can potentially use this information to identify the original owner/creator of the payload data, identify intended recipients for the data, and determine the general type of data that was postmarked. More detail on the individual metadata attributes is provided in the request details section of the document below.
SystemAuditInformation	Element	Y	System Audit Information includes various attributes about the EPM and the DES system that created the EPM. This data is added by the system during the generation process and is used for transaction identification, system identification, and other audit purposes.
Element: ElectronicPostmark.SystemAuditInformation			
desSerialNumber	Attribute	Y	DES serial number
desSoftwareVersion	Attribute	Y	DES software version
desCategoryCode	Attribute	N	Indicates type of payment method: 'C' = Commercial 'G' = Qualifying federal agency 'P' = Post office 'A' = Contract station

Field Name	Attribute /Element	Required	Description
			'M' = Military 'S' = Specimen meter 'B' = Same day billing 'L' = Don't have an account using a shared device 'T' = Multiple shared PES 'X' = Reserved for USPS use only
desFederalAgencyCode	Attribute	N	Specimen = 500 Contract Station = 600 Post Office = 700, 400, 410, 420 Military = 800 Commercial = 'Blank' Same day billing = 'Blank' Internet Based Delivery Service = 900
desFederalAgencySubCode	Attribute	N	If Meter Payment Type = A, M or P, then the sub-code is the unit ID of the Post Office. If Meter Payment Type = S the sub-code is 1000
securityModuleSerialNumber	Attribute	Y	HSM serial number
securityModuleModelId	Attribute	Y	HSM modNel ID
securityModuleSoftwareVersion	Attribute	N	HSM software version
securityModuleFirmwareVersion	Attribute	N	HSM firmware version
securityModuleTokenId	Attribute	N	HSM token ID

4.1.3 EPM Trustmark image

In version 2 of the service the generate EPM will also return the EPM trustmark image. This image can also be used in validation, but offers the convenience and aesthetics of a trustmark image instead of an XML string. This following is a sample trustmark image. This image may change and is only a sample. The image will always include a transaction id, 578 in this case, and a date of the postmark.



5. Web Services

The EPM system is a hosted system that exposes EPM-specific operations through a web service. The web service uses standard SOAP protocols in order to identify which operations are to be performed and in order to communicate data. The service contract is defined by a standard Web Service Definition Language (WSDL) document, which identifies web service, operation, and parameter names and locations. A standard Extensible Markup Language (XML) Schema is also referenced, which defines all of the USPS EPM data structures required to make use of the service.

The SDK is designed to abstract these details and provide a simple interface to communicate with the USPS EPM web service. For details on the SOAP service please reference the version 1 or 2 interface control document for the service.

5.1 Software Development Kit (SDK)

As a user of the SDK you can access either the version 1 or version 2 of the EPM web service. The primary difference between the two versions is the Meta data and the security encryption level of the service communication itself.

5.1.1 Requirements

Both a Java and a C#/.net version of the SDK are available for use in communicating with the EPM web service.

A sample client for using either the Java or C#/.net SDK is included later in this document.

5.1.1.1 Java Requirements

When using the java version of the SDK you will need the following.

- The EPostmarkSDK.jar file
- Java version 1.8 JDK or greater
- If you would like to use version 2 of the service you will also need the “Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files” from Oracle for the java JDK you are using.

5.1.1.2 C# / .net requirements

When using the C# / .net version of the SDK you will need the following.

- FilePostmark.dll
- Visual Studio 2015 or greater
- The C# / .net SDK does not support version 2 of the service at this time.

5.1.1.3 General client inputs

The following are required for the client regardless of the technology platform.

- All messages between your organization and the USPS will be signed and encrypted. Part of this process requires the use of your organization’s private/public key. One of the inputs to the service is this key pair which will be used for signature and encryption/decryption purposes. For the Code-A-Thon you will be provided with a java keystore that already contains this key pair. You can then provide the SDK the required properties to use this key store. The details of which are covered below per technology platform.
- The secondary information you will need to provide the SDK are the credentials provided to you from the USPS during registration. These include your username and password, your account ID, and your billing ID.

- The third required item are the actual files or messages that you wish to postmark. These can be provide as a string, in the case of a message, or a path to the file, in the case of a file.
- The final, and optional, item is the meta data. You can create an object containing metadata related to your file. The supported meta data is either mail piece meta data or health care meta data. The details of the metadata are covered in the USPS EPM Service V1.X Interface Document.

5.2 EPM Service Security

The USPS EPM system uses web service security standards (WS-SecurityPolicy) to secure the messages being sent between the customer and the service. The security policy is defined in the web service WSDL as part of the service contract and all clients must adhere to the security policy. In order to make use of the service, customers must create or obtain a valid X.509 certificate and associated private key and provide the public certificate portion to the USPS as part of the registration process.

The SDK interface allows you to include your client certificate as one of the paramters for interacting with the service. The sections below provide some guidelines on setting up your certificate for either the Java or C#/.net clients.

5.2.1 Using the keystore with the Java SDK

You will need to provide a couple of key pieces of information to the Java SDK about your keystore. These include the information about the keystore like the key alias and passwords you used to secure the key. The below is provided as an example and follows the client example below.

You will need a properties file. In this case I am calling it “client_sign.properties”

The contents describe your keystore.

```
org.apache.ws.security.crypto.provider=org.apache.ws.security.components.crypto.Merlin
org.apache.ws.security.crypto.merlin.keystore.type=jks
(if you select a different keystore type like bouncy castle, you will specify the type here)
org.apache.ws.security.crypto.merlin.keystore.password=client-pass
(this is your password for the keystore itself)
org.apache.ws.security.crypto.merlin.keystore.private.password=key-pass
(this is your password for your private key in the keystore)
org.apache.ws.security.crypto.merlin.keystore.alias=clientx509v1
(this is the alias you used in creating your key, see above -alias)
org.apache.ws.security.crypto.merlin.keystore.file=ClientKeyStore.jks
(this is the name of or your keystore file)
```

When you create the SDK client you will specify the name of the properties file and your signature username, which is the same as your alias name for your certificate.

```
service.setClientKey(propertiesFileName, signatureUN);
```


5.2.2 Using the certificate with the C# / .net SDK

You will need to import the certificates into the windows certificate registry for the local user

First import the EPM Service certificate (public key)

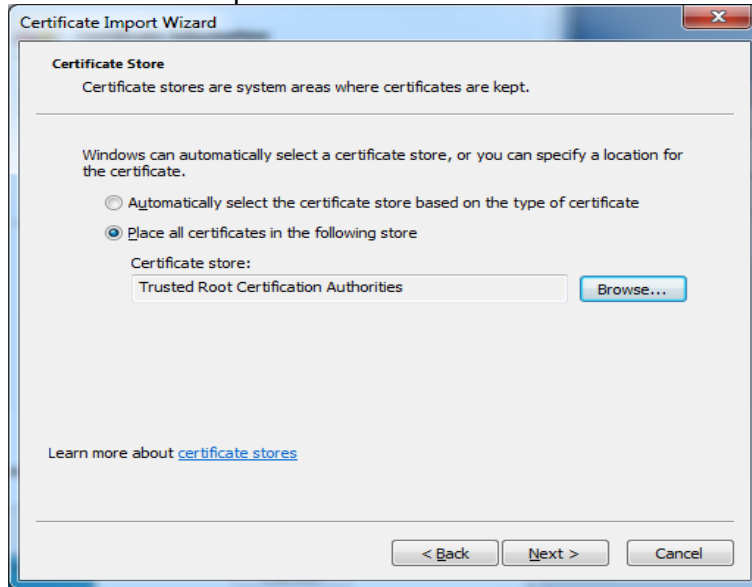
Second import the client certificate (private and public key pair)

double click the cat-wss-server certificate and select "open"

push "install certificate" click next

on certificate store screen select "place all certificates in the following store" option

click browse and pick "trusted root certification authorities"

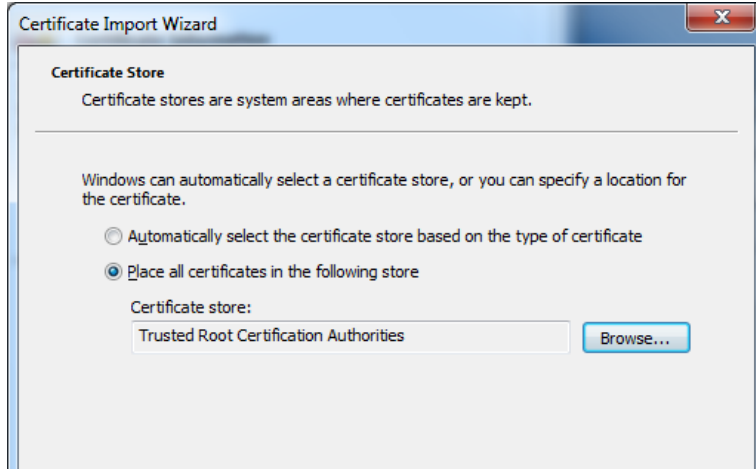


click "next"

click "finish"

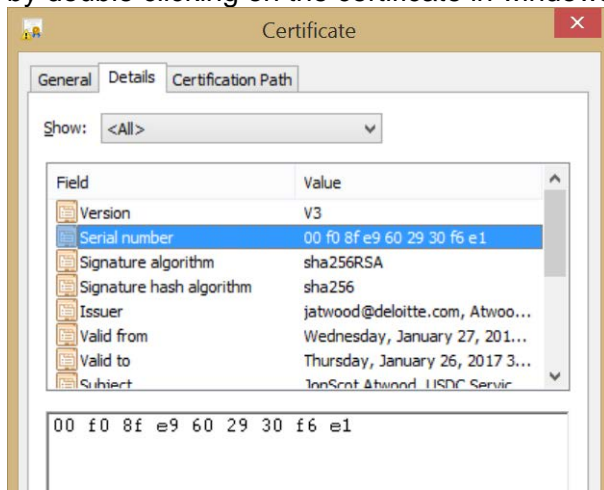
Click "yes" on the warning

now install the client certificate
 double click dotNetSDK and click next twice
 for the password enter "epmUSPS"
 on certificate store screen select "place all certificates in the following store" option
 click browse and pick "trusted root certification authorities"



click "next"
 click "finish"
 click "yes" on both warnings

For the C# / .net SDK you need to set the certificate serial number. You can get the serial number by double clicking on the certificate in windows and then clicking on the details tab.



In this case the value is "00F08FE96029930F6E1". This is set in the SDK using the command.

```
postmark.setClientCertificateSerialNumber("00F08FE9602930F6EA");
```

5.3 SDK Interface

The following operations are available from the EPM SDK for both the Java and C#/.net clients.

setCredentials(accountID, billingID, username, password);

This method allows for you to set your USPS credentials. The CredentialsConfig consists of your username, password, accountID, and billingID.

getPostmarkForFile(fileName)

This method allows you to generate an EPM for a file. The filename is a local file path to the file you want to add. The EPM postmark string is returned.

getPostmarkForByteArray(byte[])

This method allows you to generate an EPM for a byte array. The EPM postmark string is returned.

getPostmarkForMessage(message)

This method allows you to generate an EPM for a string message. The EPM postmark string is returned.

Version 2 only methods

getEPostmarkForFile(fileName)

This method is supported for version 2 of the EPM Service. This method allows you to generate an EPM for a file. The filename is a local file path to the file you want to add. The EPM postmark string and the trustmark image are returned.

getEPostmarkForByteArray(byte[])

This method is supported for version 2 of the EPM Service. This method allows you to generate an EPM for a byte array. The EPM postmark string and the trustmark image are returned.

getEPostmarkForMessage(message)

This method is supported for version 2 of the EPM Service. This method allows you to generate an EPM for a string message. The EPM postmark string and the trustmark image are returned.

There are also some batch methods that let you generate or validate EPM postmarks in a batch.

addFileToPostmark(fileName);

This method allows you to add a file to generate an EPM. The filename is a local file path to the file you want to add.

addByteArrayToPostmark(messageID, byte[]);

This method allows you to add a generic byte array to generate an EPM. The messageID is a unique identifier for the byte array.

addMessageToPostmark(messageID, Message);

This method allows you to add a string message to generate an EPM. The messageID is a unique identifier for the message.

getAllPostmarks ();

This method will generate a EPM for each file, byte array, and message added. This will be returned in a map/dictionary collection where the key is the filename or messageId and the value is the EPM string.

getAllEPostmarks ();

This is a version 2 only method. This method will generate a EPM for each file, byte array, and message added. This will be returned in a map/dictionary collection where the key is the filename or messageId and the value is an object containing the EPM string and the trustmark image.

addFileToValidate(fileName, epmString);

This method allows you to add a file and EPM string to validate against. The filename is a local file path to the file you want to add.

addByteArrayToValidate(messageID, byte[], epmString);

This method allows you to add a generic byte array and EPM string to validate against. The messageID is a unique identifier for the byte array.

addMessageToValidate(messageID, Message, epmString);

This method allows you to add a string message and EPM string to validate against. The messageID is a unique identifier for the message.

validateAllEPM

This method will validate the EPMs for each file, byte array, and message added. This will be returned in a map/dictionary collection where the key is the filename or messageId and the value is a Boolean value of true or false.

Version 2 of the service also allows you to validate the EPM using the trustmark image. The following methods have been added to support this.

validatePostmarkImageForFile(String fileName, byte[] image)

This method allows you to add a file and EPM trustmark image to validate against. The filename is a local file path to the file you want to add.

validatePostmarkImageForByteArray(byte[] bytes, byte[] image)

This method allows you to add a generic byte array and EPM trustmark image to validate against..

validatePostmarkImageForMessage(String message, byte[] image)

This method allows you to add a string message and EPM trustmark image to validate against.

getLastestValidationForEPM(epmString)

This method is available in version 2 of the service and allows the client to retrieve the most recent validation receipt for the given EPM XML string. The receipt includes the requestor name, if the validation was successful, the EPM transaction ID, and the timestamp of the validation request. A NULL value will be returned if no validation receipts exist for the provided EPM string.

getAllValidationsForEPM(epmString)

This method is available in version 2 of the service and allows the client to retrieve all of the validation receipts for the given EPM XML string.

Methods specific to the Java SDK

The Java SDK requires that you provide a keystore with your client certificate. This is covered above. The method to provide this is.

setClientKey(propertiesFileName, signatureUN)

Methods specific to the C# / .net SDK

The C# / .net SDK requires that your client certificate be in the windows certificate keystore. You then use the following method with the SDK to identify the certificate.

setClientCertificateSerialNumber(NumberAsString);

The C#/.net SDK also allows the client to work with files in interacting with the EPM service. This functionality will store the EPM results in a file that can then be used in subsequent validations. The methods used for this interaction are listed below.

setWorkingDirectory(workingDirectory)

Sets the working directory for creating the EPM file

addPostmark()

After all files, links, and the message are added, this method generates the EPM file in the directory set above. call **getEpmFileName()** to EPM file name of the generated file. The EPM file will contain all of the EPMs generated in one file.

getEpmFileName()

returns the name of the generated file.

getPostmarkedFiles()

Gets a list of all data files, links and message to be postmarked. This also includes the status of the EPM if it was requested. The list is a collection of EpmFile objects.

validatePostmark()

After all files, links, the message and the EPM file are added, this method makes sure all data files are valid. If any of the files of the files is not valid, an exception is thrown. If any file cannot be validated, error details are returned in the postmarked files list. This requires that the EPM file name be specified using **setEPMFile**.

setEPMFile(fileName)

This sets the name of the EPM to use with the validation. Throws an exception if file name is blank or if not a valid EPM file name

5.3.1 Process to use SDK

The process to use the SDK is simple and is outlined below.

The sample code in the following sections mirrors this process.

- The first step is to set you key configuration. This allows the SDK to access and use your private/public key pair in communicating with the EPM service.

- The second step is to set your credentials. These are the credentials that the USPS gave you as part of the registration process.
- The third step is to add the files and/or messages that you want to postmark.
- After this you can call the service method to getAllEPostmarks. This will take each file or message and use the EPM service to obtain a postmark for it. It will then return a hash map with a key of the message or file name, and a value of the Electronic postmark in XML. If you are using the version 2 method you will also get the EPM trustmark image.
- You can then send or store this pair for future use.

The validation process is also very simple.

- A client can use the SDK to request validation of a postmark against a file, message, or byte array by using either the appropriate methods.
- In version 2, you can also use the methods to validate using the trustmark image.
- This will return a simple true or false if the file successfully validates

5.3.1.1 Sample Java Client

```
import java.util.Map;
import java.util.Map.Entry;

import com.usps.epm.EPMSERVICEFactory;
import com.usps.epm.EPMSERVICEV1;

public class EPMClientV1 {

    public static void main(String args[]) throws java.lang.Exception {

        EPMSERVICEFactory serviceFactory = new EPMSERVICEFactory();

        EPMSERVICEV1 service = serviceFactory.getEPMSERVICEV1();
        // Setup the information about your client PKI certificate properties
        service.setClientKey("client_sign.properties", "clientx509v1");
        // Setup the information about your USPS EPM service credentials
        service.setCredentials("CODE-A-THON-JAVA-SDK-001", "CODE-A-THON-JAVA-SDK-001", "code-a-thon-
java-sdk", "dw98@B2ekEGYm5y");

        // example add a text file
        String fileID1 = "C:/Dev/hello2.txt";
        service.addFileToPostmark(fileID1);
        // example add a binary file
        String fileID2 = "C:/Dev/gatorhood.jpg";
        service.addFileToPostmark(fileID2);
        // example add a string message
        String messageID1 = "myMessage";
        String messageContent = "The Apple";
        service.addMessageToPostmark(messageID1, messageContent);
        // example add a byte array
        String messageID2 = "myBytes";
        byte[] messageContent2 = "The quick brown Fox".getBytes();
        service.addByteArrayToPostmark(messageID2, messageContent2);
        // Get a MAP with key of your file or message ID and a value of the EPM XML string
        Map<String, String> allEPostmarks = service.getAllPostmarks();

        for(Entry<String, String> entry : allEPostmarks.entrySet()) {
            // loop through the results and print the postmarks
            // normally the postmark would be saved with the original file for future validation
            String key = entry.getKey();

            String value = entry.getValue();

            System.out.println("For File = [" + key + "]);
            System.out.println("Postmark = [" + value + "]);
        }
        System.exit(0);
    }
}
```

5.3.1.2 Sample C#/.net client

This sample application instantiates the postmark object, adds a data file to be postmarked, and then call the addPostmark() method so the postmark file is created. It then calls getAllPostmarks to get the file name and the corresponding postmark hash.

```
using System;
using System.IO;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine("Testing File EPM");
                string filePath = @"C:\dev\hello2.txt";
                Postmark postmark = new Postmark();
                postmark.setCredentials("CODE-A-THON-DOTNET-SDK-001", "CODE-A-THON-DOTNET-SDK-001", "code-a-thon-dotnet-sdk", "dw98@B2ekEGYm5y");
                postmark.setClientCertificateSerialNumber("00dfc5608efcd6b61e");
                string epmString = postmark.getPostmarkForFile(filePath);
                Console.WriteLine(epmString);
                Console.WriteLine();
                bool valid = postmark.validatePostmarkForFile(filePath, epmString);
                Console.WriteLine("Validation: " + Convert.ToString(valid));
                Console.WriteLine();
                Console.WriteLine("Testing message EPM");
                string testMessage = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
                epmString = postmark.getPostmarkForMessage(testMessage);
                Console.WriteLine(epmString);
                Console.WriteLine();
                valid = postmark.validatePostmarkForMessage(testMessage, epmString);
                Console.WriteLine("Validation: " + Convert.ToString(valid));
                Console.WriteLine();
                Console.WriteLine("Testing byte array EPM");
                byte[] byteArray = { 2, 3, 5, 7, 11, 13, 17, 23 };
                epmString = postmark.getPostmarkForByteArray(byteArray);
                Console.WriteLine(epmString);
                Console.WriteLine();
                valid = postmark.validatePostmarkForByteArray(byteArray, epmString);
                Console.WriteLine("Validation: " + Convert.ToString(valid));
                Console.WriteLine();
                byte[] imageEPM = postmark.getEPostmarkForFile(filePath);
                File.WriteAllBytes(@"C:\dev\TestEPM1.jpg", imageEPM);
                valid = postmark.validatePostmarkImageForFile(filePath, imageEPM);
                Console.WriteLine("Image File Validation: " +
                    Convert.ToString(valid));
                imageEPM = postmark.getEPostmarkForMessage(testMessage);
                File.WriteAllBytes(@"C:\dev\TestEPM2.jpg", imageEPM);
                valid = postmark.validatePostmarkImageForMessage(testMessage,
                    imageEPM);
                Console.WriteLine("Image Message Validation: " +
                    Convert.ToString(valid));
                imageEPM = postmark.getEPostmarkForByteArray(byteArray);
                File.WriteAllBytes(@"C:\dev\TestEPM3.jpg", imageEPM);
                valid = postmark.validatePostmarkImageForByteArray(byteArray,
                    imageEPM);
                Console.WriteLine("Image Byte Array Validation: " +
                    Convert.ToString(valid));
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.Message);
            }
            Console.ReadKey();
        }
    }
}
```


5.4 Troubleshooting common error conditions

The EPM web service uses the SOAP fault data structure to return errors to the client. The SDK catches these errors and prints out the error code, ID, and message to the console. The errors will appear in this format.

```
Failed to generate EPM: A service error has occurred.
Error Code = 10, Error ID = 15188,
Error Message = Username or password is invalid.
```

The following table lists the possible errors.

Error Code	Error Message	Error Condition / Notes
100	Requester account id is invalid.	The requester account id specified for the requesting party is not valid.
101	Requester account id is not authorized to make this request.	The account id specified for the requesting party is not authorized to perform the function as defined by the incoming request.
110	Given customer account id is invalid for this request.	The billing account id provided in the request is not associated with a billing account in the system or is not allowed to perform the requested operation.
111	Specified customer account has not been activated.	The billing account id provided in the request is associated with a customer account that has not yet been activated or has been retired.
112	Specified customer account has been locked.	There is an issue with the customer account associated with the provided billing account id and the issue must be resolved before any requests will be processed.
200	Missing required field(s) or invalid field value(s).	One or more pieces of input data have failed validation. Required data may be missing or a data value may be invalid. If this error code is returned, the FaultDetail element will contain a child DataAttributeError element. The DataAttributeError element will identify the name of the input field that failed validation and the reason for the failure. Failure reasons are defined in the EPM XML schema and include: MISSING_REQUIRED_VALUE and INVALID_DATA.
900	Unable to process request due to an internal system error.	An unexpected system error has occurred during processing that prevented the transaction from completing.

5.4.1 Issues with your certificate

If you receive an error like the following, there may be an issue with your certificate.

```
Exception in thread "main" javax.xml.ws.soap.SOAPFaultException: A security
error was encountered when verifying the message
```

5.4.2 Issues with your message

If you are missing information in your message you may receive an error like the following.

```
Failed to generate EPM: A service error has occurred.
Error Code = 200, Error ID = 15191,
Error Message = Missing required field(s) or invalid field value(s),
Fields: [payloadData]
```

In this case I did not attach a file to postmark

5.4.3 Validation error for EPM service V1

Version 1 of the EPM service reports a SOAP fault when validation fails. If you receive the following error, then it means that the file and EPM you sent for validation do not match.

```
Failed to validate EPM: A service error has occurred.
Error Code = 220, Error ID = null,
Error Message = Validation of EPM data has failed
```

This will also return a false value. For the V2 of the service you won't see the above message in the logs and only the false value will be returned.

As part of the SDK design all SOAP faults on the validation call will return a value of **false** for validation, regardless of the type of SOAP fault.

5.5 Using the EPM web service

5.5.1 Customer Registration

Customer registration is a process by which the USPS creates a billing account for the customer and the customer is provided with a unique identifier for use in billing and making requests to the EPM web service. In general, the registration process consists of:

1. Setting up a billing account through the USPS billing process and obtaining a set of USPS EPM credentials (username and password)
2. Creating or obtaining a valid X.509 certificate and associated private key and providing the certificate to the USPS
3. Obtaining the USPS EPM X.509 certificate

5.5.2 Credentials for the Healthcare Code-A-Thon

.NET SDK

u/n: code-a-thon-dotnet-sdk
p/w: dw98@B2ekEGYm5y

RequesterAccountId: CODE-A-THON-DOTNET-SDK-001
BillingAccountId: CODE-A-THON-DOTNET-SDK-001

Java SDK

u/n: code-a-thon-java-sdk
p/w: dw98@B2ekEGYm5y

RequesterAccountId: CODE-A-THON-JAVA-SDK-001
BillingAccountId: CODE-A-THON-JAVA-SDK-001

6. Appendix A – EPM XML Schemas

6.1 Links to WSDL and Schema locations

- Main WSDLs
 - <https://epm.usps.post/epm/services/epmWebService-v1?wsdl>
 - <https://epm.usps.post/epm/services/epmWebService-v2?wsdl>
- WS-Security policy
 - <https://epm.usps.post/epm/services/epmWebService-v1?wsdl=des-ws-security-policy-v2.0.wsdl>
 - <https://epm.usps.post/epm/services/epmWebService-v2?wsdl=epm-ws-security-policy-v3.0.wsdl>
- XSD with element and metadata
 - <https://epm.usps.post/epm/services/epmWebService-v1?xsd=usps-des-shared-model-v2.0.xsd>
 - <https://epm.usps.post/epm/services/epmWebService-v2?xsd=usps-epm-shared-model-v3.0.xsd>
 - <https://epm.usps.post/epm/services/epmWebService-v2?xsd=usps-epm-shared-model-common.xsd>

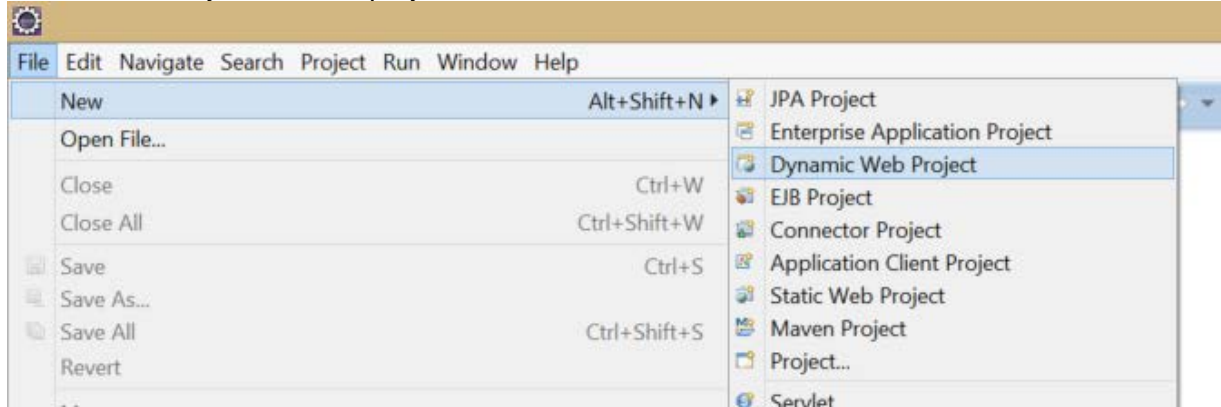
7. Appendix B - Setup the sample Java client using the SDK in Eclipse

For this example I am using

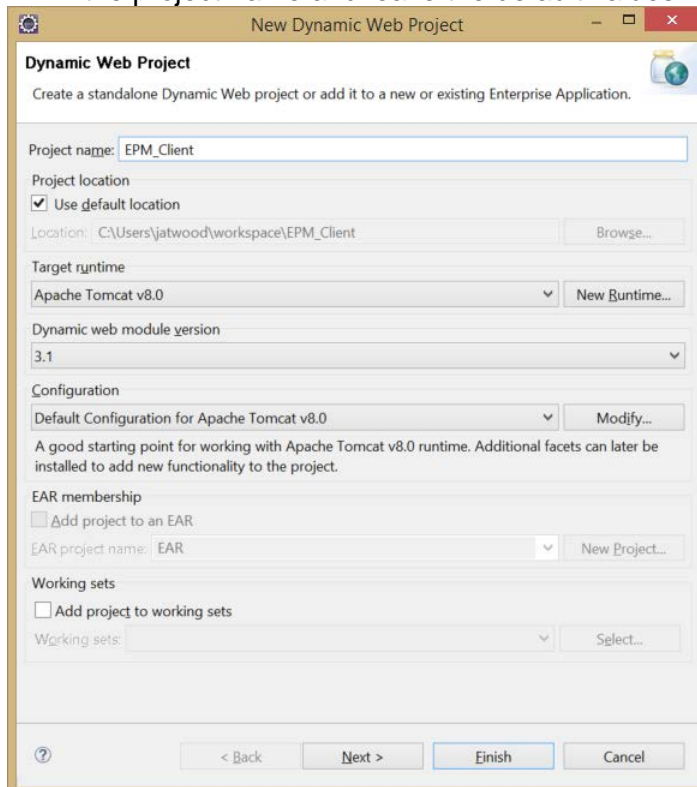
- Eclipse Mars release
- Sun Java JDK 1.8

7.1 Steps

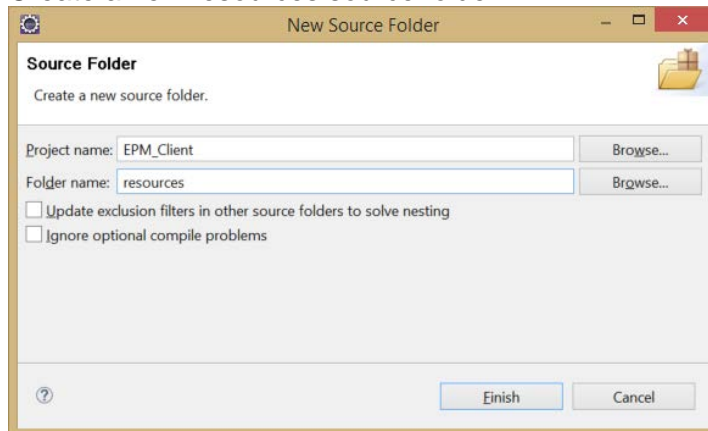
Create a new dynamic web project



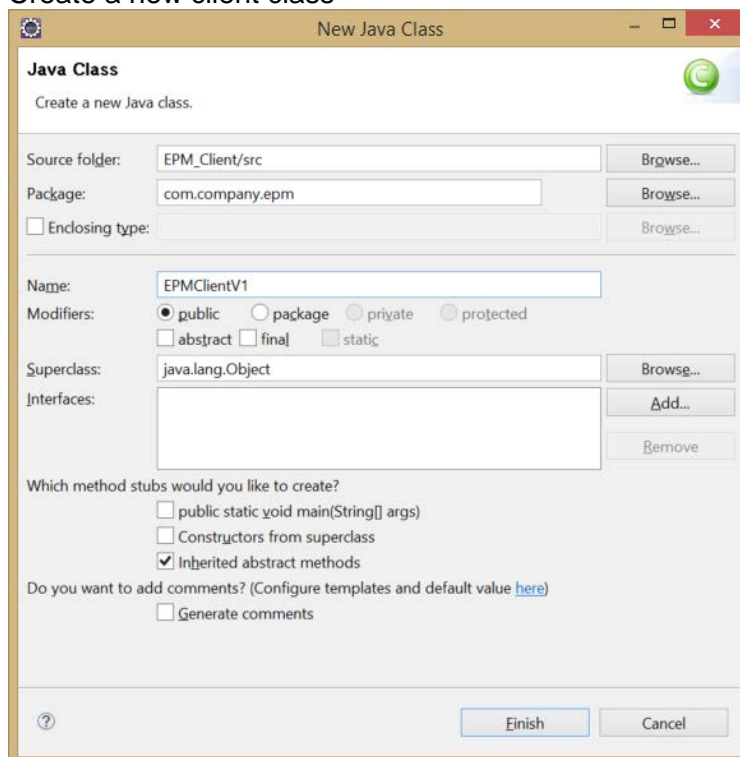
Fill in the project name and leave the default values.



Create a new resources source folder



Create a new client class



Add the sample client code from above, adjust the samples files as needed

Java EE - EPM_Client/src/com/company/epm/EPMClientV1.java - Eclipse

```

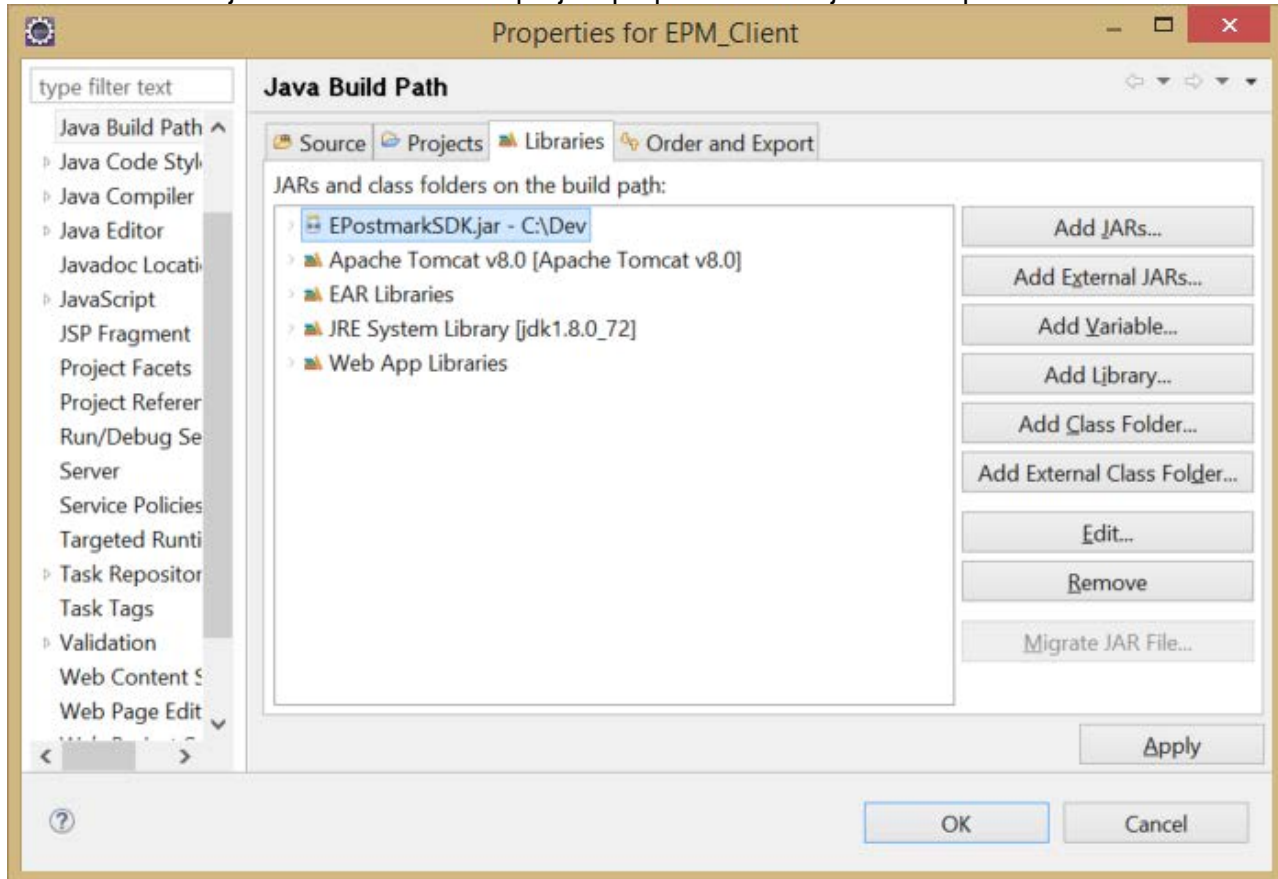
1 package com.company.epm;
2
3 import java.util.Map;
4 import java.util.Map.Entry;
5
6 import com.usps.epm.CredentialsConfig;
7 import com.usps.epm.EPMServicesV1;
8 import com.usps.epm.EPMServicesV1Impl;
9 import com.usps.epm.KeyConfig;
10
11 public class EPMClientV1 {
12
13     public static void main(String args[]) throws java.lang.Exception {
14
15         EPMServicesV1 service = new EPMServicesV1Impl();
16         // Setup the information about your client PKI certificate properties
17         service.setClientKey(createKeyConfig());
18         // Setup the information about your USPS EPM service credentials
19         service.setCredentials(createCredentialsConfig());
20
21         // example add a text file
22         String fileID1 = "C:/Dev/hello2.txt";
23         service.addFileToPostmark(fileID1);
24         // example add a binary file
25         String fileID2 = "C:/Dev/gatorhood.jpg";
26         service.addFileToPostmark(fileID2);
27         // example add a string message
28         String messageID1 = "myMessage";
29         String messageContent = "The quick brown fox jumped over the lazy dog!";
30         service.addMessageToPostmark(messageID1, messageContent);
31         // Get a MAP with key of your file or message ID and a value of the EPM XML string
32         Map<String, String> allEPostmarks = service.getAllEPostmarks();
33
34         for(Entry<String, String> entry : allEPostmarks.entrySet()) {
35             // loop through the results and print the postmarks
36             // normally the postmark would be saved with the original file for future validation
37             String key = entry.getKey();
38
39             String value = entry.getValue();
40
41             System.out.println("For File = [" + key + "]");
42             System.out.println("Postmark = [" + value + "]");
43         }
44     }
45 }

```

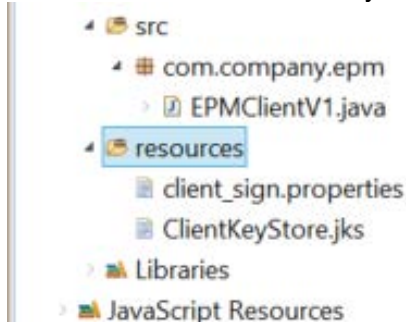
Markers Properties Servers Data Source Explorer Snippets Problems Console Search

You will notice a lot of errors, the next step is to import the EPM SDK

Add the external jar for EMP under the project properties in the java build path screen.



In the resources folder add you java keystore with your public/private key and the config file



Make sure the properties file contains the correct values for your java keystore

For example only...

```
org.apache.ws.security.crypto.provider=org.apache.ws.security.components.crypto.Merlin
org.apache.ws.security.crypto.merlin.keystore.type=jks
org.apache.ws.security.crypto.merlin.keystore.password=client-pass
org.apache.ws.security.crypto.merlin.keystore.private.password=key-pass
org.apache.ws.security.crypto.merlin.keystore.alias=clientx509v1
org.apache.ws.security.crypto.merlin.keystore.file=ClientKeyStore.jks
```

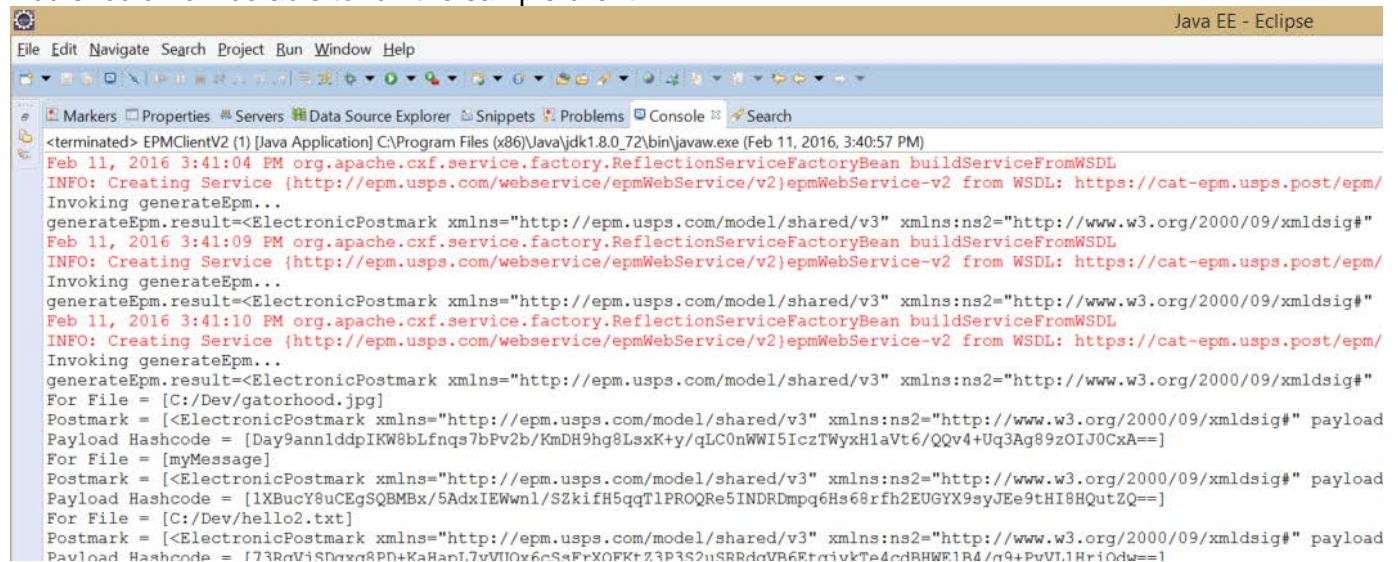

Set your credentials in the sample client

```

49
50= private static CredentialsConfig createCredentialsConfig() {
51     // Convenience method for setting your USPS EPM credentials
52     String accountID = "yourAccountID";
53     String billingID = "yourBillingID";
54     String username = "yourUserName";
55     String password = "yourPassword";
56
57     CredentialsConfig credentials = new CredentialsConfig();
58     credentials.setAccountID(accountID);
59     credentials.setBillingID(billingID);
60     credentials.setUsername(username);
61     credentials.setPassword(password);
62     return credentials;
63 }
64
65= private static KeyConfig createKeyConfig() {
66     // Convenience method for setting your client PKI certificate properties
67     // The properties file below contains more properties required for accessing your java keystore
68     // with your Private/Public Key for your certificate
69     // These are kept in your resource directory
70     String signaturePropertiesFileName = "client_sign.properties";
71     String signatureUN = "clientx509v1";
72
73     KeyConfig config = new KeyConfig();
74     config.setSignaturePropertiesFileName(signaturePropertiesFileName);
75     config.setSignatureUN(signatureUN);
76     return config;
77 }

```

You should now be able to run the sample client



```

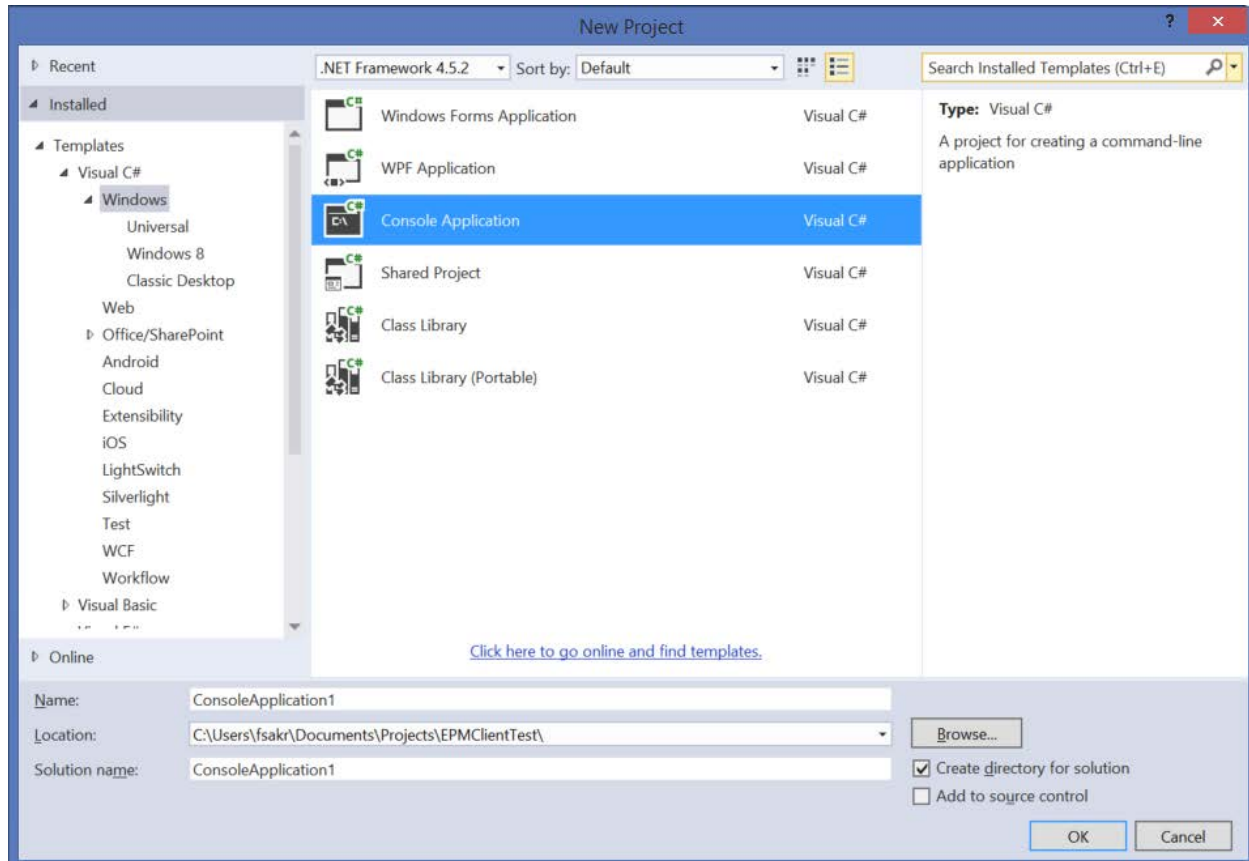
Java EE - Eclipse
File Edit Navigate Search Project Run Window Help
Markers Properties Servers Data Source Explorer Snippets Problems Console Search
<terminated> EPMClientV2 (1) [Java Application] C:\Program Files (x86)\Java\jdk1.8.0_72\bin\javaw.exe (Feb 11, 2016, 3:40:57 PM)
Feb 11, 2016 3:41:04 PM org.apache.cxf.service.factory.ReflectionServiceFactoryBean buildServiceFromWSDL
INFO: Creating Service {http://epm.usps.com/webservice/epmWebService/v2}epmWebService-v2 from WSDL: https://cat-epm.usps.post/epm/
Invoking generateEpm...
generateEpm.result=<ElectronicPostmark xmlns="http://epm.usps.com/model/shared/v3" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"
Feb 11, 2016 3:41:09 PM org.apache.cxf.service.factory.ReflectionServiceFactoryBean buildServiceFromWSDL
INFO: Creating Service {http://epm.usps.com/webservice/epmWebService/v2}epmWebService-v2 from WSDL: https://cat-epm.usps.post/epm/
Invoking generateEpm...
generateEpm.result=<ElectronicPostmark xmlns="http://epm.usps.com/model/shared/v3" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"
Feb 11, 2016 3:41:10 PM org.apache.cxf.service.factory.ReflectionServiceFactoryBean buildServiceFromWSDL
INFO: Creating Service {http://epm.usps.com/webservice/epmWebService/v2}epmWebService-v2 from WSDL: https://cat-epm.usps.post/epm/
Invoking generateEpm...
generateEpm.result=<ElectronicPostmark xmlns="http://epm.usps.com/model/shared/v3" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"
For File = [C:/Dev/gatorhood.jpg]
Postmark = [<ElectronicPostmark xmlns="http://epm.usps.com/model/shared/v3" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" payload
Payload Hashcode = [Day9annlddpIKW8bLfnqs7bPv2b/KmDH9hg8LsxK+y/qLC0nWWI5IczTWyxH1aVt6/QQv4+Uq3Ag89zOIJOcXA==]
For File = [myMessage]
Postmark = [<ElectronicPostmark xmlns="http://epm.usps.com/model/shared/v3" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" payload
Payload Hashcode = [1XBucY8uCEgSQBMBx/5AdxIEWwnl/SZkifH5qqTlPROQRe5INDRDmpq6Hs68rfh2EUGYX9syJEe9tHI8HQtZQ==]
For File = [C:/Dev/hello2.txt]
Postmark = [<ElectronicPostmark xmlns="http://epm.usps.com/model/shared/v3" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" payload
Payload Hashcode = [73RnV4sDxxgRPD+KaHant.7vUUDx6rSsFrXOFKtZ3P3S2uSSRRdcVR6FtciukTa4edRHW1R4/a9+PvVT.1HriOdW==]

```

If you see errors consult the trouble shooting section

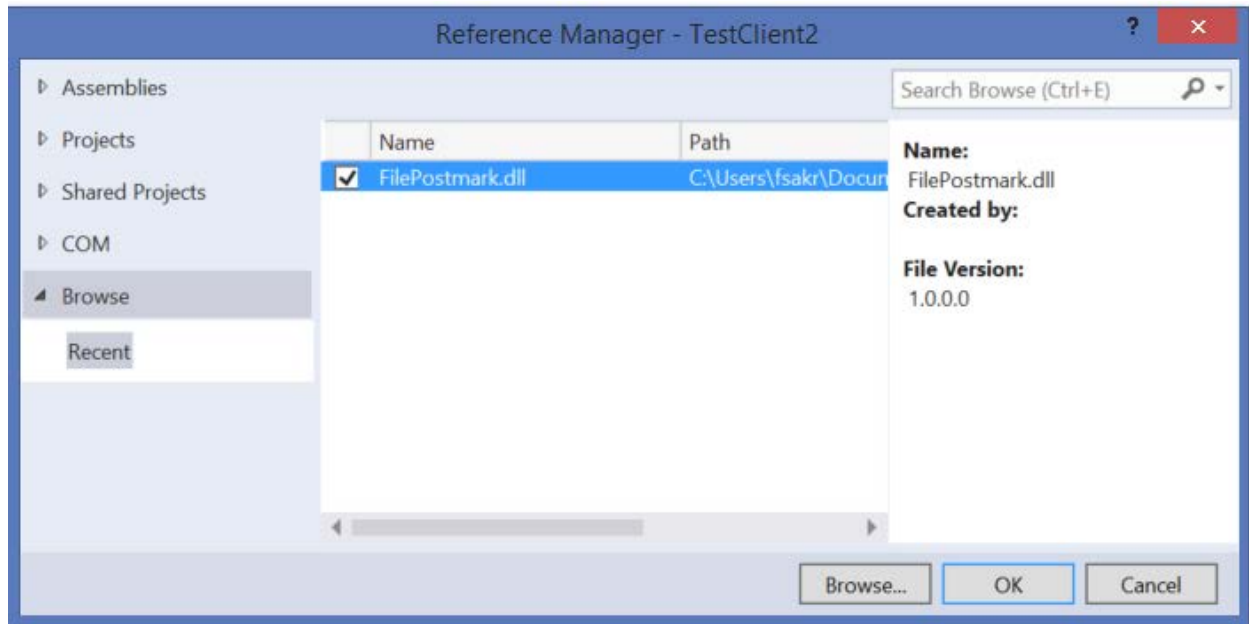
8. Appendix C – Setup the sample .Net client using Visual Studio 2015

8.1 Steps



Create a new console application

Add reference to FilePostmark.dll:



You are now able to add your client.