

Client Deployment Guide for the Status Receipt Message of the Electronic Freight Management (EFM) Package

Prepared for:

U.S. Department of Transportation

Prepared by:

Battelle

July 17, 2009



TABLE OF CONTENTS

	<u>Page</u>
BACKGROUND	1
INTENDED AUDIENCE	2
OVERVIEW	2
CLIENT LOGIC	3
SECURITY – SSL/TLS	4
Running Keytool.....	4
Running PKCS12Import.....	6
Running the Client	6
WSDLS AND XSDs	6
FULL CLIENT SAMPLE	6

List of Figures

Figure 1. Domain Model.....	3
-----------------------------	---

BACKGROUND

The EFM Initiative is intended to improve the efficiency and productivity of freight movements by evaluating and promoting innovative e-business concepts that support coordination and information sharing among supply chain partners. EFM is an open-architecture, Internet-based solution that promotes the use of standards to allow supply chain partners to efficiently track freight shipments as they move through the supply chain. EFM provides shippers—the supply chain owners—with visibility to meet very tight performance standards and improve operational efficiencies. EFM offers uniform access to existing customized database formats, computing platform independence, and adaptable services. EFM allows each supply chain partner to exchange data with other supply chain partners via Web services using eXtensible Markup Language (XML) data standards in a service oriented architecture (SOA) and open-source software products. The framework employs secure encryption and digital certificates, ensuring that any information exchanged between partners is authorized and secure, that data is not corrupted in-transit, and that the data transmitted is complete. EFM provides a gateway for automated interfaces and software capabilities that are designed to support computer-to-computer interactions over the Internet.

Accurate, visible information provides supply chain partners with the actionable intelligence they need to improve operational efficiency and increase agility in a fast-paced global business environment. Without this information, supply chain partners can face delayed shipments, disrupted assembly lines, congested cargo transfer points, and stressed inventories. Many large firms use logistics software and electronic data exchange to standardize data flows...but small to medium-sized firms often do not because of the high implementation cost and technical expertise required to effectively use the software and standards. The Electronic Freight Management (EFM) framework supports in-transit visibility to all supply chain owners, from the largest to the smallest. EFM can be used by all supply chain partners—from shippers to 3PLs to customs brokers—creating a truly integrated, “shared view” of the status of shipments across the globe and helping to increase the competitiveness and the effectiveness of the supply chain.

All supply chain partners can benefit from using supply chain visibility tools like EFM. While as few as two partners in the supply chain can benefit from using EFM, the value and operational efficiencies grow as more supply chain partner’s link into EFM. As more partners adopt EFM, fewer manual transactions are required; a more complete “shared picture” among partners enables better and more responsive decision-making.

As part of this EFM Initiative, USDOT has sponsored the development of what is known as the EFM Package. The EFM Package is a collection of documents and computer source code which may be downloaded, free of charge, for use by organizations wishing to take advantage of the benefits of EFM. The complete set of documents and the source code is available at the EFM website <http://efm.us.com>.

INTENDED AUDIENCE

This document is one of many in the EFM package and is provided as a guide for partners, such as rail, ocean, truck and air, who themselves are required to provide shipment status information to clients or other parties. When thinking about traditional data flows, this includes events and information similar to what is contained in the EDI 214, EDI 315 and similar messages. Those partners who have this type of information and wish to share that information to a customer are considered EFM clients.

EFM clients do not need to host EFM web services themselves, but instead, would, simply using the guidance provided herein, along with the aid of knowledgeable programmers, to connect the output of their existing systems to the EFM web services hosted by the party interested in their data.

In this particular case, the data of interest is referred to as “Transportation Status” and includes data items such as event (i.e. pickup, drop off, etc.), partner name, transportation IDs (i.e. container #, pallet ID, etc.), date, time and others.

OVERVIEW

Partners can send freight status via the StatusReceiptService web service that is hosted by any partner who wishes to receive such status updates. Typically, the recipient is the buyer, consignee or a value-added service providers (on behalf of the consignee).

The StatusReceiptService has a web operation called ReceiveTransportationStatus that takes TransportationStatusType, TransportEquipmentType, and ShipmentStageType objects as parameters.

The TransportationStatusType and ShipmentStageType objects, as well as the entire EFM domain model, are based on the Universal Business Language (UBL) specification v2.0. The UBL defines the schema for what properties a TransportationStatus object has as well as the relationships to other UBL objects.

Using the herein referenced web service description language (wsdl) and Extensible Markup Language (XML) Schema Definitions(XSD) files, clients wishing to connect to the web service can use various tools and utilities generate all of the classes necessary to call the web service(s) and then write the client code using the generated objects. The example herein uses the wsimport utility, part of the the Java JAX-WS 2.0 toolkit, and used by IDEs such as NetBeans and Eclipse, for adding a new Web Service Client. This same basic ability to import the wsdl and xsd files is available from several other development tools, including .NET, but as EFM focuses on the use of open and open-source tools, it will be the reference tool used.

Figure 1 depicts a class diagram of a subset of the EFM domain model. The classes shown in the diagram are the ones most relevant to TransportationStatus.

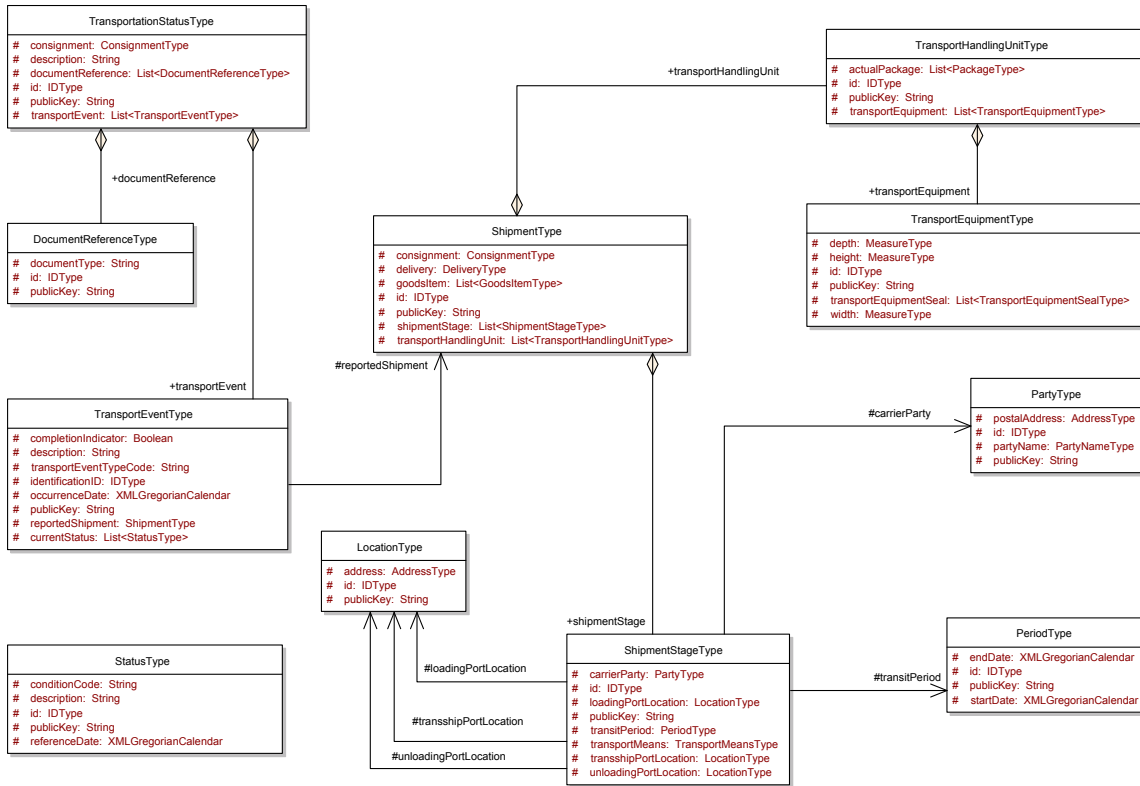


Figure 1. Domain Model

CLIENT LOGIC

When an external client is using the StatusReceiptService to send transportation status messages, three services are relevant:

1. BookingService
2. StatusReceiptService
3. Template Service

After generating the requisite classes using wsimport or similar, the client code can be written. This section describes what the client should do. The basic steps are:

1. Find the correct shipment in the buyer's instance via a web service call (BookingService)
2. Obtain a basic TransportationStatusType template from the buyer's instance via a web service call (TemplateService)
3. Determine ShipmentStage
4. Populate the objects with the correct data
5. Determine correct TransportEquipment object
6. Call the ReceiveTransportationStatus web service

SECURITY – SSL/TLS

EFM will generate PKCS12¹ (PKCS #12: Personal Information Exchange Syntax Standard) files and X.509 certificates that can be used to create/add to java keystores and trust managers. The PKCS#12 standard specifies a portable format for storing or transporting a user's private keys, certificates, miscellaneous secrets, etc.

Once the PKCS12 and X.509 certificate is delivered, the client side administrator must do two things:

1. Create or append to a java keystore
2. Add the new CA to the trust store.

Running Keytool

The widely available tool *keytool* can be used for step 1, as follows. Let's assume the X.509 certificate file is called *CA_org.efm-cert.pem*. Let's also assume that we create a new key store called *efm.jks*.

```
keytool -importcert -v -keystore efm.jks -storepass <pswd> -alias org.efm_ca -file CA_org.efm-cert.pem -trustcacerts
```

You will be prompted for a password for the keystore file. At the end, you will be asked if you want to trust the certificate, to which you should type *yes*.

¹ <http://www.rsa.com/rsalabs/node.asp?id=2138>

Here is what you will see:

```
$ keytool -import -keystore efm.jks -alias alliance -file CA_org.efm-cert.pem
Enter keystore password:
Re-enter new password:
Owner: CN=org.efm - Root CA, OU=Web Hosting, O=org.efm, L=Kansas City, ST=Missouri, C=US
Issuer: CN=org.efm - Root CA, OU=Web Hosting, O=org.efm, L=Kansas City, ST=Missouri, C=US
Serial number: b581fb4b28a5b500
Valid from: Tue Jan 13 17:12:58 EST 2009 until: Fri Jan 11 17:12:58 EST 2019
Certificate fingerprints:
    MD5:  C7:17:F9:28:CE:18:2F:46:E9:99:AF:53:AE:D8:25:BD
    SHA1: BF:AA:1F:57:3D:A7:05:DD:68:F5:B0:76:5A:8F:93:A1:D9:16:65:8A
    Signature algorithm name: SHA1withRSA
    Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: B9 61 E7 9A 81 75 6C E3   87 9A B4 86 09 C8 A9 8D   .a...ul.....
0010: DA 38 D7 AD                               .8..

]
]

#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
    CA:true
    PathLen:2147483647
]

#3: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: B9 61 E7 9A 81 75 6C E3   87 9A B4 86 09 C8 A9 8D   .a...ul.....
0010: DA 38 D7 AD                               .8..

]

[CN=org.efm - Root CA, OU=Web Hosting, O=org.efm, L=Kansas City, ST=Missouri, C=US]
SerialNumber: [    b581fb4b 28a5b500]
]

#4: ObjectId: 2.16.840.1.113730.1.13 Criticality=false

Trust this certificate? [no]: yes
Certificate was added to keystore
```

Running PKCS12Import

Now, add the info from the PKCS#12 file to the java key store. Assume the name of the PKCS file is *efm-test-user-cert.p12*. You will be prompted for the PKCS12 password, the keystore password, and the key password.

```
$ java -jar pkcs12import.jar -file efm-test-user-cert.p12 -alias alliance3 -keystore efm.jks
```

Running the Client

Add the following to your JVM parameters:

- -Djavax.net.ssl.keyStore=C:\dev\efmcert\efm.jks // path to jks file
- -Djavax.net.ssl.trustStore=C:\dev\efmcert\efm.jks // path to jks file
- -Djavax.net.debug=ssl,handshake // just for debugging
- -Djavax.net.ssl.keyStorePassword=<password> // password for key store file
- -Djavax.net.ssl.trustStorePassword=<password> // password for trust store file

WSDLs AND XSDs

The WSDL and XSDs used to generate the stub client are located in the following folder:

- <http://efm.us.com/Files/ClientSampleWSDL.zip>

FULL CLIENT SAMPLE

A full sample client is included in the EFM source distribution under the following folder:

- SourceCode\trunk\efm-platform-sample\src\main\java\org\efm\platform\sample\ws