IT Common Platforms

External Web Services - Quick Start Guide

Version 0.0.3

Document Revisions

Date	Version	Description	Author(s)
2/25/2016	0.0.1	First draft	B. Brandaw
4/13/2016	0.0.2	Added Notifications section	S. Jinright
7/22/2016	0.0.3	Formatting and corrections	B. Brandaw
10/3/2016	0.0.4	Client Services Review	T. Hailu

Sign-Off

Title

Name Date

Title

Name Date

Table of Contents

1. Overview

This document serves as a starting point for Market Participants interested in utilizing ERCOT's External Web Service (EWS) infrastructure. This infrastructure provides a wide variety of functions, including:

- Query and retrieval of Reports/Extracts
- Submissions into ERCOT's Market Management System (MMS)
- Submission into ERCOT's Outage Scheduler

Please note that this document is not intended to be a substitute for any of the existing user guides and ERCOT Interface Specifications. It is an overview of the process to set up a programmatic interface with ERCOT, organized in sequential steps.

- A more comprehensive list of user guides related to market data transparency including ERCOT's "EIP External Interfaces Specification" document that describes machine to machine interfaces for Market Participant applications that need to interact with ERCOT Nodal Market systems is located on ERCOT's website at http://www.ercot.com/services/mdt/userguides.

It is assumed that the entity interested in this implementation has already registered as a Market Participant with ERCOT and has completed any qualification requirements to acquire an ERCOT digital certificate for secure access to the ERCOT Market Information System (MIS).

2. Getting Started

2.1. Preparation for Communicating with the MOTE

- Once your company has completed the registration and qualification process to become an ERCOT Market Participant, a digital certificate is issued to your company's designated User Security Administrator (USA). The following is an overview of the USA registration steps:

- Upon completion of the Market Participant registration process, the designated USA will receive an email with instructions for installing the ERCOT Root Certification Authorities (CA) digital certificate for the MOTE.
- Once the ERCOT Root CA digital certificate is installed, the USA will need to go to the Market Information Systems (MIS) in a browser and log in using the DUNS number and password provided in the email from ERCOT. This will trigger the download of the USA's digital certificate.
- The USA will now have the ability to create digital certificates for end userss or for Application Program Interfaces (APIs) in their company via the MIS.
- For a more detailed explanation of the USA registration process, please see the *MPIM Digital Certificate User Guide* located on ERCOT's website. *http://www.ercot.com/services/mdt/webservices*

- In order to obtain an end user or API digital certificate, contact your company's USA. In order to use ERCOT's API, you will need to request a MOTE API digital certificate from your USA.
- Retrieve additional digital certificates from ercot.com as necessary (http://www.ercot.com/services/mdt/webservices):
  - VeriSign_Class_3_Public_Primary_Certification_Authority_G5.cer (VeriSign's 2048bit Root Cert)
  - VeriSign_Class 3_International_Server_CA_G3.cer (Intermediate Cert)
  - Client Root Certificates for the MOTE: ERCOT_TEST_CA.cer
- Download the API documentation, XSDs and WSDL from ERCOT.com at *http://www.ercot.com/services/mdt/xsds/index.html.*
  - The WSDL and XSDs are located in the *External Web Services XSD vX.XX.zip* (X.XX indicates the latest version posted).
  - The API documentation can be found using the same link: *EIP External Interfaces Specification vX.XX.zip* (X.XX indicates the latest version posted).
  - GetReports Only Documentation to be published.
- Review introductory sections of the documentation
  - Sections 1 and 2 of either document will describe security implementation, message structure, and other overarching concepts.
- Contact ERCOT Client Services and request the MOTE URL from ERCOT.
  - The URL is intentionally left out of the WSDL

2.2. Development for Communicating with the MOTE

ERCOT does not mandate a particular technology for client development. Any modern development language should be able to invoke the services provided that the base technology meets the requirements..

The following is a step-by-step to develop a sample application using the following technologies:

- Java 1.8
- Eclipse
- Maven
- Spring, using the Springboot project functionality

While the steps are specific to this stack, the concepts and sequence of events should remain consistent across client technologies.

2.2.1. Creating a Keystore From Your Client Certificate

1. Make sure you have a JDK (preferably Java 8) installed

2. Copy your client certificate (yourcertificate.pfx) to the bin directory of the JDK. Remove all white spaces from the certificate name before proceeding
3. From the Windows start menu, go to Start > All Programs > Accessories. Right click Command Prompt, then click Run as administrator (requires admin rights)
4. CD to the JDK bin folder where you placed yourcertificate.pfx
5. Run the following command
   keytool –v –list –storetype pkcs12 –keystore yourcertificate.pfx > clientcert.txt
   Enter the password for yourcertificate.pfx when prompted. This will create a text file named clientcert.txt in the working directory with information about your certificate that will be used in subsequent steps
6. Run the following command
   keytool –importkeystore –srckeystore yourcertificate.pfx –srcstoretype pkcs12 –srcstorepass yourcertificatepassword –srcalias yourcertificatealias –destkeystore clientcert.jks –deststoretype jks –deststorepass changeit –destalias clientcert

   Your certificate alias can be found in clientcert.txt created in the previous step. Executing this command will create keystore clientcert.jks in the working directory. The Java client we create in the next section will need this keystore file

2.2.2. Setting Up a Maven Project in Eclipse

1. In Eclipse, go to File > New > Project
2. Select Maven Project and click Next
3. Check “Use default Workspace location” and click Next
4. On the archetype selection screen, choose “maven-archetype-quickstart” and click Next
5. Choose a Group Id suitable to your organization (such as “com.ercot”) and type “ews-client” for Artifact Id. Click Finish
6. Eclipse will generate a Maven quickstart project for you with the appropriate structure

2.2.3. Generating Client Classes with Maven

1. Right click on your maven project and select New > Source Folder. Name the folder “src/main/schemas” and click Finish
2. Unzip “External Web Services XSD Vx.xxB”. Copy all files ending with .xsd and .wsdl to the “src/main/schemas” folder of your Maven project
3. Update your pom.xml file so the <parent>, <properties>, <dependencies>, and <build> sections look like the pom.xml supplied in the Appendix.

4. In the plugins section of pom.xml, you should find two plugins listed. Comment out the spring-boot-maven-plugin around the <plugin> start and end tags. Leave the maven-jaxb2-plugin active.
5. Right click your Maven project, then select Run As > Run Configurations. In the Goals textbox type “package” without quotes, then press Apply, followed by Run
6. The maven-jaxb2-plugin will use “Nodal.wsdl” (as specified in the pom) to create proxy classes in the “src/main/java” source folder

PLEASE NOTE: Do not attempt to generate proxies by using the WSDL provided by the URL. This approach will not be successful. Only generate proxies from the WSDL provided in the XSD bundle downloaded earlier.

2.2.4. Setting Up Springboot Security

1. If your project doesn’t have a source folder named “src/main/resources”, create it by right clicking on your Maven project and choosing New > Source Folder
2. Copy the “clientcert.jks” keystore you created in the first section of this document into the “src/main/resources” folder in your project
3. Copy the “SecurityPolicy.xml” file from the supplied project code into the same folder

2.2.5. Creating Springboot Classes

1. Right click on your project’s “src/main/java” source folder, then choose New > Package. Name the package something relevant. It will contain 3 client classes to configure and run the application as a Springboot app
2. Copy the supplied “Application.java”, “EwsClient.java”, and “EwsConfiguration.java” classes from the code bundle (or create and fill them in yourself).
3. Open “EwsConfiguration.java” and check the following.
   - Make sure the context path of the marshaller() method is set to the package where “RequestMessage.java” lives. If you didn’t specify otherwise, the maven-jaxb2-plugin will have placed it in “com.ercot.schema._2007_06.nodal.ews.message” and no change will be necessary
   - The keyStore() method sets parameters relevant to the keystore you created in the first section. Make sure the filename matches what you actually named the .jks file, and that it is in the “src/main/resources” folder in your project. The password should be whatever you used when you were creating the keystore in the first section (default is changeit)

- The keyStoreHandler() method sets a private key password. This is the password for yourcertificate.pfx. The default alias should match the –destalias used in the first section when generating the keystore. SecurityPolicy.xml in the resources folder should also use this same alias. It's "clientcert" by default if you've been following this guide
- The securityInterceptor() method should configure itself with the "SecurityPolicy.xml" file in your project's resources folder
- The ewsClient() method should set a default URI. Observe the soap:operation elements in Nodal.wsdl for wsdl:operation MarketTransactions, MarketInfo, and Alerts. The default URI should be the portion of the soap:operation string that they have in common (up to "HttpEndPoint" in this example)

4. "EwsClient.java" is a bare bones client class that uses Spring's WebServiceTemplate to create SOAP messages and make web service calls. It doesn't need to be modified but certainly can be to meet your needs
5. Open "Application.java" and observe the following
   - The formRequest() method creates a RequestMessage object and populates it with the required fields (as specified by the wsdl) to make a successful request. It is currently set up to make a "getSystemStatus" web service call. Make sure the source and user id of the HeaderType object are aligned with what's specified in yourcertificate.pfx
   - The CommandLineRunner bean makes a call to the EwsClient. The "soap_address_x" parameter should match the URL of the EWS server you're trying to interact with, the "soap_action_x" parameter should match the endpoint of the type of web service call you're trying to make (in the case of "getSystemStatus" this is the market info endpoint), and the third argument is the RequestMessage for the web service call

2.2.6. Building the Application with Maven

1. Open your pom.xml file. In the build > plugins section of the pom, comment out the maven-jaxb2-plugin around the <plugin> start and end tags, as there is no reason to generate Java proxy classes a second time
2. Uncomment the spring-boot-maven-plugin by removing the comment start and end points around the <plugin> tags that you created earlier in this tutorial
3. Right click your project, then choose Run As > Maven build. It should default to using the Maven run configuration you created earlier
4. A .jar file should be produced in the target folder of your project

2.2.7. Running the Client

1. From the Windows start menu, go to All Programs > Accessories > Cmd to open a command prompt
2. Run the following command
   java –jar jarname.jar
   where jarname.jar is the full path to the jar in the target folder of your project that you created in the previous step (for example C:\MyWorkspace\MyProject\target\jarname.jar)

2.2.8. Next Steps

Once this sample project is implemented, the Market Participant's development resources can modify the sample and integrate the functionality into the Market Participant's existing application(s).

2.3. Business Transactions in MOTE

The first attempt at a transaction should be Get SystemStatus. This transaction is very simple, with no payload on the request or response.

- If this is successful, it is assured that the security configuration and message structures are correct.
- If this fails, it is most likely due to a security configuration problem.

Get SystemStatus is documented in Section 7.2.2 of the Interface Specification. Examples of the request and response messages for this transaction are provided in the Appendix. Contact your ERCOT Account Manager for assistance if you are unable to make this step work

Once Get SystemStatus is working, addition of new transactions will be easier. The only difference will be element values in the Header, presence of well-formed Payloads and, in some cases, changes in Endpoint.

- Error messages will be helpful at this stage to determine root cause of failures.
- Almost all failures at this stage will be message structure or data validation exceptions.

2.4. Business Transactions in Production

After you have completed implementation of all of your needed transactions and successfully tested in the MOTE, consider moving to Production. To do this, you will need to:

- Retrieve Production Certificates and install the certificates following the same procedures used for MOTE Certificates above.
- Contact ERCOT Client Services and request the Production URL from ERCOT.
- Validate installation by using Get SystemStatus. This will verify that all certificates and security settings are working properly before business functions are needed in Production.

3. Notifications Listeners

3.1. Assess the Need

QSEs with Resources and QSEs providing ERS are required to have a Notification Listener set up over the Wide Area Network. Other Market Participants interested in receiving Notifications from ERCOT can set up a Notification Listener as well.

Types of Notifications include:

- Notices and Alerts (Ex: Submission Warnings, DAM Market Messages, AS Deployment Information, Operator Notices)
- Offer/BidSet Submission acceptance or error messages
- Awards (Ex: Energy Offer, Energy-Only Offer, Ancillary Service Awards)
- Obligations (Ex: PTP Obligations, Ancillary Service Obligations)
- Startup/Shutdown Instructions
- Outages

For a complete list of notifications, please see Section 5 of the “External Interfaces Specification” document on the ERCOT website.
http://www.ercot.com/services/mdt/xsds/index.html

3.2. How it Works

- Market Participants will receive Notification messages from ERCOT over HTTPS via the Listener. These messages are signed using ERCOT’s

server certificate. The Notification Listener should validate the signature on the notification message using ERCOT's public key to ensure it was signed by ERCOT, then respond with an Acknowledge message. The Acknowledge message is not signed. See examples in the appendix.

- Once complete, contact ERCOT Client Services to have the ERCOT Notification service configured to send notifications to the new listener and provide the URL(s) of the new listener.

- If there is an excessive delay (more than 15 notifications sent without an Acknowledge response in a 30 minute period) ERCOT may begin placing notification messages in a lower priority queue until the issue is resolved.

- Implement a Notification Listener using the resources listed below.

3.3. Implementation

The Notification Listener is based upon the WS-Notification specification. WSDL and XSD files needed are located on ERCOT.com under *Services->Market Data Transparency->XSDs* (http://www.ercot.com/services/mdt/xsds) - within the "*External Web Services XSD"* zip file.

- Notification.wsdl

- Notification.xsd – defines the structure of the Notify message and the Acknowledge response.

- Message.xsd – defines the ResponseMessage structure which will contain the notification details.

MPs would need to implement the NotificationConsumer Interface.

The Notification Listener should be designed to consume a Notify message and reply with an Acknowledgement. Most listeners will pass along the body of notification messages for further handling to other downstream processes. The listener itself is just the service that a Market Participant sets up to receive notification messages from ERCOT and acknowledges receipt of those messages. Any further handling of the content within the messages is beyond the scope of this document. See the Appendix for examples.

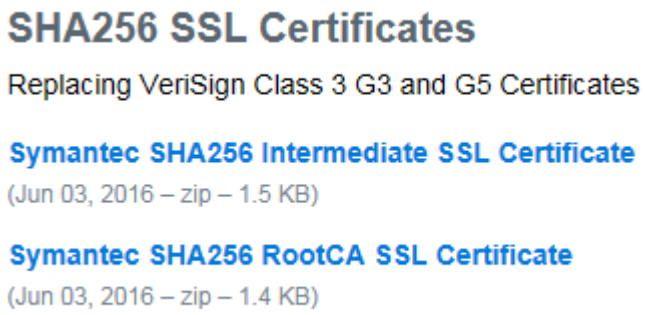
3.4. Certificates

- Purchase a certificate from a Certificate Authority such as GoDaddy, Thawte, VeriSign, etc. This certificate will be used to establish the SSL

tunnel for the HTTPS connection. ERCOT does not accept self-signed certificates.

- Provide ERCOT with the corresponding Intermediate and Root certificates for the CA certificate that you chose.

- Add the SHA256 Intermediate and Root SSL certificates to your keystore from the following link on ERCOT.com (http://www.ercot.com/services/mdt/webservices). These will be used to validate ERCOT's signature on the notification messages. Some market participants use ERCOT's public key instead of the intermediate/root pair. ERCOT's public key is available at the same link mentioned above. ERCOT encourages the use of the intermediate/root pair instead of the public key, but supports both approaches.



4. Where to Go for Help

Any questions regarding your implementation should be directed to your ERCOT Account Manager or ERCOT's Client Services department. That team is equipped with a variety of support resources and has direct contact to the ERCOT IT teams responsible for operating and developing the solution. You may call the general ERCOT Client Services phone number at (512) 248-3900 or contact ERCOT Client Services via email at ClientServices@ercot.com.

5. Appendix

5.1. Get SystemStatus Request Sample

```
<RequestMessage
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
```

```
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
    xmlns="http://www.ercot.com/schema/2007-06/nodal/ews/message">
    <Header>
        <Verb>get</Verb>
        <Noun>SystemStatus</Noun>
        <ReplayDetection>
            <Nonce>1234</Nonce>
            <Created>2015-11-07T11:52:18.881-06:00</Created>
        </ReplayDetection>
        <Revision>001</Revision>
        <Source>QSAMP</Source>
        <UserID>USER1</UserID>
        <MessageID>1234567890</MessageID>
    </Header>
</RequestMessage>
```

5.2. Get SystemStatus Response Sample

```
<ns0:ResponseMessage xmlns:ns0="http://www.ercot.com/schema/2007-
06/nodal/ews/message">
    <ns0:Header>
        <ns0:Verb>reply</ns0:Verb>
        <ns0:Noun>SystemStatus</ns0:Noun>
        <ns0:ReplayDetection>
            <ns0:Nonce>ad1378e7a0c9fff969f893039988e496</ns0:Nonce>
            <ns0:Created>2015-11-07T11:52:06.59-06:00</ns0:Created>
        </ns0:ReplayDetection>
        <ns0:Revision>001</ns0:Revision>
        <ns0:Source>ERCOT</ns0:Source>
        <ns0:UserID>USER1</ns0:UserID>
        <ns0:MessageID>1234567890</ns0:MessageID>
    </ns0:Header>
    <ns0:Reply>
        <ns0:ReplyCode>OK</ns0:ReplyCode>
        <ns0:Timestamp>2012-11-07T11:52:06.59-06:00</ns0:Timestamp>
    </ns0:Reply>
</ns0:ResponseMessage>
```

5.3. Notification Message Sample

```
 <ns0:Notify xmlns:ns0="http://www.ercot.com/schema/2007-
 06/nodal/notification">
   <ns0:NotificationMessage>
     <ns0:Message>
      <ns1:ResponseMessage xmlns:ns1="http://www.ercot.com/schema/2007-
 06/nodal/ews/message">
       <ns1:Header>
        <ns1:Verb>created</ns1:Verb>
        <ns1:Noun>Alert</ns1:Noun>
        <ns1:ReplayDetection>
            <ns1:Nonce>fe77fbeb4bec83d55d49249283ca85ef</ns1:Nonce>
            <ns1:Created>2016-05-26T17:00:05.974-05:00</ns1:Created>
        </ns1:ReplayDetection>
        <ns1:Revision>1.19V</ns1:Revision>
```

```
        <ns1:Source>ERCOT</ns1:Source>
        <ns1:UserID>ERCOT_ADMIN</ns1:UserID>
        <ns1:MessageID>CM-COPVAL-NOTF-QSAMP-2016-05-26T16:00:05-
06:00</ns1:MessageID>
    </ns1:Header>
    <ns1:Reply>
        <ns1:ReplyCode>OK</ns1:ReplyCode>
        <ns1:Timestamp>2016-05-26T17:00:05.974-05:00</ns1:Timestamp>
    </ns1:Reply>
    <ns1:Payload>
        <ns2:Event xmlns:ns2="http://www.ercot.com/schema/2007-
06/nodal/ews">
            <ns2:qse>QSAMP</ns2:qse>
            <ns2:ID>CM-COPVAL-NOTF</ns2:ID>
            <ns2:type>Market Message</ns2:type>
            <ns2:priority>High</ns2:priority>
            <ns2:source>MMS</ns2:source>
            <ns2:issued>2016-05-26T17:00:05-05:00</ns2:issued>
            <ns2:summary>CM-COPVAL-NOTF COP is missing for CC
Configuration: ABC_CC1_1
                belonging to CC Plant: ABC_CC1 for Trade Date:
5/26/2016 for hour(s) [18, 19, 20, 21, 22, 23, 24]</ns2:summary>
        </ns2:Event>
        <ns1:format>XML</ns1:format>
    </ns1:Payload>
  </ns1:ResponseMessage>
 </ns0:Message>
</ns0:NotificationMessage>
</ns0:Notify>
```

5.4. Notification Message Acknowledgement Sample

```
<Acknowledge xmlns="http://www.ercot.com/schema/2007-
06/nodal/notification" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <ReplyCode>OK</ReplyCode>
</Acknowledge>
```

5.5. Pom.xml