PJM FTR

External Interface Specification

Date: Feburary 16th, 2018

Revision 17

Transmittal No.:

GE Grid Solutions
Redmond, Washington USA

Revision History

Revision	Transmittal #	Date	Comments
1	461E0325	October 31, 2002	[peh] Original Preliminary Draft. The Halloween Document.
2	461E0336	November 18, 2002	[peh] Updated per comments received from PJM on November 5th and on November 8th via e-mail messages. And, inclusive of comments in discussion with Jeff Bastian on November 12th, 2002 regarding secondary market design.
3	461E0340	November 27, 2002	[peh] Corrections resulting from testing, conversations with Jeff Bastian and Ray Fernandez on November 19th, 2002, and comments from Jeff Bastian via e-mail on November 20th, and changing the root elements to define a SubmitRequest and SubmitResponse root.
4	461E0369	March 5, 2003	[peh] Corrections resulting from testing. Add field to support auction round in the Market Info message. Redesign of the secondary trading messages. Change query for obligation and option price data. Make changes to bring Web U/I more consistent with XML.
5	461Exxxx	March 20, 2003	[peh] Corrections to support new reports requested by Ray on March 11th, 2003.
6	461E0387	April 10th, 2003	[peh] Corrections to modify the QueryInitialARRs request to be by market name rather than interval dates and also to modify the response to include path, bid MW, and cleared MW only. Also, corrections to align the documented behavior with the schema and actual behavior. Also, change the host computer name to ftr.pjm.com.
7	461E0389	April 17th, 2003	[peh] Cleanup, removing some of the lingering statements about ARR trading.
8	461E0390	May 2, 2003	[peh] Minor update to add trade type on cleared FTRs. Note: revision 8 and revision 9 below are released under same transmittal number for same date.
9	461E0390	May 2, 2003	[peh] Updated the bilateral traded FTRs on the secondary market to add new feature of requiring the market/round to

			be selected for each FTR posted to the secondary market.
10	461Exxxx	January 20, 2004	[peh] Updated document to fix two minor textual errors. Section 5.1.3 changed to clarify that bid-in MW and bid-in Price are returned in the query response only. Section 5.6.2 modified to clarify the default handling of the since attribute. Section 5.7.3 modified to add SelfScheduled to the trade type.
11		April 1st, 2006	[dbs] Updated document to allow specification and usage of the Period attribute and element and include new Position and Credit reports.
12		June 1st, 2006	[dbs] Updated document to reflect increases in the allowable size of credit limits and requirements.
13		January 1st, 2009	[dbs] Updated Section 5.6. Added MarketMode and redefined MarketType in the MarketInfo element.
14		January 31st, 2013	[mjg] Added section 6.3 for Submitting ARR bids to annual market.
15		February 20th, 2013	[pmf] Removed QueryInitialArrResults, replaced with QueryArrResults. Added QueryArrConstraints.
16		December 10th, 2014	[mjg] Added section 5.5 for ARR Requests XML Query
17		Februrary 21st, 2018	[dbs] Added section 5.8 for ARR Sinks.

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe the participant's XML-based programmatic interface to the PJM FTR system.

The participant uses this document to guide in the formulation of messages to be sent to the FTR system and the interpretation of messages received from the FTR system. This document describes the addressing, exchange protocol, data, and XML formulation for all defined messages.

The reader of this document is assumed to be a software engineer whose intent is to understand the requirements of the participant's interface to the FTR system and to implement the software necessary to exchange data. Data exchange functions include:

- Query for Initial ARRs, submitted FTRs, cleared FTRs, and other data.
- Submit FTRs to annual or monthly auctions.
- Manage portfolios of option or obligation paths.
- Trade ARRs and FTRs on secondary market.

The business market rules that govern the data exchange for the FTR system are not described in this document. Market rules are published by the PJM separately (see reference [1]).

1.2 Technology Prerequisite Knowledge

In order to design and implement the XML data exchange software interface to the PJM FTR system using this programmatic API, the reader should be familiar with:

- Using XML for data description, XML Schemas, XML Namespaces, and XML parsing and validation.
- Protocols HTTP, HTTPS, and TCP/IP.
- Security and authentication technologies: encryption, authorization, and SSL (secure sockets layer).
- General network communication software methodology.

1.3 Document References

The following documents are pre-requisite reading material supporting this document.

[1] PJM FTR Business Rules Revised, published by PJM. Document identifier # 180329 V15, Revised 12/06/02.

The following references are suggested helps for the technology used by the programmatic API.

[2] Professional XML, 2000 by a myriad of authors, published by Wrox Press.

There is also a Beginning XML by the same publisher but this edition here is more complete.

[3] Professional XML Schemas, 2001 by a countable set of authors, published by Wrox Press.

This book also has sufficient discussion of XML namespaces to provide background for the Programmatic API.

[4] Professional Java Web Services, 2001 by a number of people, published by Wrox Press.

This reference has a good discussion of the SOAP standard giving a number of examples. The inclusion reference is not necessarily an endorsement of the Java language nor does it suggest that the programmatic API is a web services implementation.

[5] SSL and TLS Essentials, 2002 by Stephen Thomas, published by Wrox Press.

A good overview of the SSL protocol. However, it is recommended that SSL capability by acquired from a 3rd party vendor or the OpenSSL software be considered.

[6] RFC 2616: Hypertext Transfer Protocol, HTTP 1.1, 1999 by R. Fielding et' el, published by the Networks Working Group, IETF.

The best description of HTTP is the RFC documenting the standard.

1.4 Terminology

The following terms and acronyms are used by this document.

ACK – acronym meaning acknowledgement and indicates a normal and successful acknowledgement message. Contrast with NAK.

ARR – Auction Revenue Right, the mechanism by which the proceeds from the annual FTR auction are allocated. ARRs are allocated to network transmission customers and to firm point-to-point transmission for the duration of the annual planning period. ARRs can be converted to self-scheduled FTRs in round 1 of the annual FTR auction.

Auction – the method of selling and buying FTRs during the annual and monthly FTR markets. For further information, see entry for *market.*

base64 – a coding scheme used to encode the username and password on the HTTP authorization header. The base64 encoding algorithm is published in several locations and defined by RFCs 1521 and 2045. Public available software (Java API, Visual Basic, and C) is available to code and decode base64 strings.

Class – the class of an FTR is the designation of the daily interval governed by the FTR. The class is designated as *on peak*, *off peak*, or *24 hour*. The 24 hour interval is inclusive of on peak and off peak. The actual hourly intervals specified by on peak or off peak are determined by PJM FTR market rules.

ComplexType – the type descriptor used in this document for XML complex type declarations. A complex type is a type that has sub-elements. Constrast this with a SingletonType.

CRLF – acronym meaning *Carriage Return Line Feed* and is the definition of a line terminator. A line is contiguous set of octets (8-bit characters) that is terminated by a CRLF character. The CRLF character is often platform dependent and may be defined as the hex codes 0x0D, 0x0A or simply as the hex code 0x0A. The usage of 0x0D, 0x0A two-character couplet is standard for Windows platforms (e.g. Windows NT). The usage of 0x0A, single-character, is standard for many Unix platforms. The CRLF can be created using language features for generating a *new-line* character such as the C escape character \n.

Character Data – is the XML term that defines the non-markup data that exists between an XML start and end tag. For example, given a start and end tag of UNITMW, the character data in the following example is the number 400.0: <UNITMW>400.0</UNITMW>.

DTD – acronym meaning *Document Type Definition* and it is defined by the XML specification as the statements that describe the elements, attributes, and entities that comprise an XML document. This specification uses the XML Schema only (see entry for XML Schema). DTDs are not used.

First-Normal-Form – is the specification that character data defined by XML start and end tags (see definition of character data in this terminology section) represents a single data item. Data that does not fit the *First-Normal-Form* definition is data that includes sets of information, repeating groups, or structured elements. For the purposes of this specification, Date and Time data is considered to be First-Normal-Form data even though it defines sets of information (e.g. month, day, year, hour, minute).

FTR – Financial Transmission Right is an instrument that protects firm transmission service customers from increased cost due to transmission congestion in the day-ahead market when

their energy deliveries are consistent with their firm reservations. FTRs are financial entitlements to rebates for congestion charges paid by the firm transmission service customer. FTRs may be offered for sale and purchased in annual and monthly auctions and they can be traded with other PJM members on the secondary bilateral trading market.

Hedge Type – specifies whether an FTR is an obligation or an option. The main difference between an obligation and option FTR is how they are settled on the day-ahead. The hourly economic value of an FTR obligation is negative (a liability) when the FTR path is in a direction opposite to the congested flow; whereas, the value of an FTR option is zero (neither a benefit or a liability) when the FTR path is in the direction opposite to the congested flow. Also, options are defined on a subset of the paths available to obligation type FTRs. ARRs are always considered obligations. Consult the PJM FTR market rules for more information on the differences between obligations and options.

HTTP – acronym meaning Hyper-text Transfer Protocol. HTTP is an application-level protocol for distributed, collaborative, hypermedia, information systems. Or, more simply, HTTP is a network communications protocol used to send and receive data over the Internet. The version of HTTP used by this specification is 1.1 and this is often referred to as HTTP/1.1. The HTTP/1.1 protocol is defined by RFC 2616.

HTTPS – acronym meaning Hyper-text Transfer Protocol Secure. HTTPS is a secure protocol where the security is established by SSL (see terminology entry). HTTPS does not define nor does it add new communications features to HTTP, it is merely a secure version of HTTP. The HTTPS name is used to specify the protocol in the URL declaration to identify it as being secured by SSL.

IETF – acronym meaning Internet Engineering Task Force and it is the body whose members work together to define, specify, and regulate the various standards used for Internet network communications. The RFCs referenced by this specification are defined and maintained by the IETF. The IETF web site can be consulted for more information: www.ietf.org.

Interval – a period of time marked by an explicit start and end date. An interval is used to denote the fractional time period of an FTR offered on the secondary bilateral trading market or the specific period of time of the FTR *period* known by the start and end dates. Contrast this definition with the definitions for *Period* and *Term* appearing in this glossary.

ISO – acronym meaning International Standards Organization and it is the standards body responsible for a number of international standards used implicitly and explicitly by this document. Explicit standards cited by this specification include ISO 8601 used to define Date and Time standards and conventions; and, ISO-8859-1 used to define the XML operative character set. In the deregulated electricity industry, ISO also means Independent System Operator. That definition is not used in this document.

Market – FTR auctions are executed in markets that are held annually and monthly. The annual market takes place on a date chosen by PJM per the FTR market rules. The monthly market occurs on a monthly basis to buy/sell FTRs for the following month. FTRs purchased or sold in the annual market have a *term* of one year. FTRs purchased or sold in the monthly market have a term of one month.

NAK – is the acronym meaning negative acknowledgement and is the opposite in meaning to the ACK acronym. NAK is commonly given when a network communications message is rejected or not processed completely. Contrast with ACK.

Node – See entry for PNode.

Obligation FTR – see Hedge Type.

Option FTR – see Hedge Type.

Path – is defined by two pricing nodes called the source node and the sink node. The path has a direction pointing from the source node to the sink node.

Period – specifies the name of a defined FTR *term*. A specific period may be designated by the *interval* of January 1st to March 31st.

PNode – PNode is a pricing node. FTRs can be defined on paths between two pricing nodes only. All nodes described in this document are pricing nodes so the letter P prefix is dropped usually.

RFC – acronym meaning Request For Comments. The RFC is the documentation method used by the IETF for developing and documenting network communications standards for the Internet community. Several network communications standards cited by this specification are defined by RFCs. Each RFC is identified by a number. For a complete list of RFCs and to obtain copies of RFCs, see the information located at the IETF web site: www.ietf.org.

Request/Reply – is a messaging style where a client sends a request message to a server and the server responds with a reply message. Request/Reply is also sometimes called Client-Server messaging. In the case of the PJM FTR interface, the participant is always the client who initiates the request and PJM is always the server who responds with a reply.

RTO – Regional Transmission Organization.

SingletonType – is the type name used in this document to describe a single element definition that does not have any element sub-structure. It may define attributes that are used on the type and it may or may not define character data (note: the term *character data* in this sense is specific to XML terminology, see glossary entry).

SOAP – means Simple Object Access Protocol and it is the protocol used to *wrap* the various FTR data content messages described by this document. SOAP is used in a number of different messaging patterns. Various profiles of usage have been established and documented. The profile assumed by this FTR specification is to use SOAP as a message wrapper to provide common root context to make routing and handling of messages a little easier. SOAP is not used by this specification for its more popular profile: as a remote procedure call (RPC) mechanism.

SSL – acronym meaning Secure Sockets Layer. SSL is a protocol standard used to establish a secure, encrypted, connection between a given client and server. SSL Version 3.0 is the protocol used for establishing the secure communications between participants and PJM. SSL is an industry de-facto standard developed originally by the Netscape Corporation.

TCP/IP – acronym meaning Transport Control Protocol over Internet Protocol. TCP/IP is actually two separate protocol standards however, as used in this specification (and, in most other applications), TCP/IP are considered a single network communications protocol. TCP/IP is the protocol specified by various RFC documents published by IETF (RFCs in this case are not as useful as other more popular publications of the TCP/IP standard). TCP/IP is used as the carrier protocol for all messages defined by this specification.

Term – the duration of an FTR or an ARR. The term of duration is one-year for an annual FTR or ARR and one-month for a monthly FTR. Other terms of duration may be available in the future. Contrast this definition with the definition for *Period* and *Interval* occurring in this glossary.

Trade Type – is the type of market exchange. A trade is either an offer to **Sell** or a bid to **Buy** at a specified quoted price. A special trade type is defined for self-scheduling FTRs from ARRs. This trade type is called **SelfScheduled** and is allowed in round 1 of the annual auction.

URI – acronym meaning Universal Resource Identifier and is a similar construct to URL (defined next). URL in fact is a kind of URI. The URI is used to name Internet resources that are not necessarily Internet locations. More information on URI can be found in RFC 1630.

URL – acronym meaning Uniform Resource Locator. URL is the standard method for specifying network addresses and network resources used by Internet protocols. The URL defines the protocol, network host address, optional port number, resource path and fragments. URLs are used to specify PJM FTR server addresses that receive messages sent by the participants. URL is defined by RFC 1738.

Valid – as used in this context, *valid* is the measurement of an XML document that is correctly formatted according to its schema. PJM accepts only valid XML documents in messages received from participants. All valid XML documents are implicitly also *well-formed*.

W3C – is the acronym meaning World Wide Web consortium which is a Internet community standards body whose purpose is to develop, publish, and maintain standards for the Internet community. Since W3C is not a government body, its publications are called *recommendations* as the term *standard* is defined as government authorized. The XML and HTML recommendations (aka *standard*) is published by the W3C. More information on the W3C publications and organization can be found at their web site: www.w3c.org.

Well-formed – is used to measure an XML document that is formatted according to the syntax rules set forth by the XML recommendation. A *well-formed* document can be correctly parsed by an XML parser. A document that is not well-formed may fail to parse completely.

XML – is the acronym meaning Extensible Markup Language. XML is a W3C published recommendation. XML is used as the basic format method for all messages defined by this specification. More information on XML can be found at the web site: www.w3c.org/xml.

XML Namespaces – provide a mechanism for qualifying the name of an XML coded entity per a unique identifier defined by a URI. The use of namespaces allows element and other tag names to be derived from more than one vocabulary without conflict or collision. Namespaces, though optional with many XML usage patterns is required for this FTR specification. For more information, consult the web site www.w3c.org.

XML Schema – provided an improved means of declaring structure and content of an XML document. XML Schema is a replacement for the older document type declaration specified by the DTD. Use of XML Schemas are required by this FTR specification. For more information, consult the section on XML Schema at the web site www.w3c.org or references [2] and [3] cited in section 1.3 of this document.

2 External Interface Data Exchange Overview

This chapter provides an overview of the data exchange sets and data descriptions involved in the FTR auction system.

2.1 Data Exchange Synopsis

The FTR data exchange is described in terms of data submitted by the participant and data received as a result of a query in different functional categories as shown in the table below.

Category	Description
Annual Market	The annual market offers the sale the entire FTR capability of the transmission system with a one-year subscription period through a multi-round auction The name of an annual market is determined by PJM.
Monthly Market	The monthly market offers for sale FTRs for any residual transmission capacity remaining after the annual auction subscription. The monthly market also allows participants to sell FTRs that they own. FTRs acquired in the monthly auction have a variable term (BOPP). The name of a monthly market is determined by PJM.
Secondary Market	A secondary trading market is available to participants who wish to trade (buy, sell) existing FTRs bilaterally with other participants.
Portfolio Management	Portfolio management allows users to manage portfolios of paths denoted by the source and sink pricing nodes that are grouped together for the ease of use. Portfolios are primarily a feature that benefits the web browser user but portfolio management is supported via the programmatic API.

Auctions are executed in annual markets and monthly markets. The market and its description is defined by PJM and published via a web page or available by XML query (see QueryMarketInfo).

2.1.1 Annual Market Quote Submittal

The annual market is held once a year at a time designated by PJM according to the FTR Market Rules. The annual market is conducted by rounds numbered from 1 to 4. FTRs purchased in an earlier round can be offered for sale in the subsequent round. FTRs can be self-scheduled from existing ARRs in the first round only.

Self-Scheduling an FTR in round 1 of the annual market

To self-schedule an FTR from an existing ARR, you submit an FTR Quote into round 1 of the annual market using a trade type of *SelfScheduled* rather than *Buy* or *Sell*. This is the only instance where the trade type of SelfScheduled is allowed. This is a price-taker auction buy bid and therefore a price is not specified on the quote. The self-scheduled FTR must have exactly the same path (i.e. source and sink nodes) as the ARR and cannot be for a MW quantity greater than that of the ARR.

The self-scheduled FTR Quote consists of the following data:

- Path designating source node to sink node
- MW quantity (non-zero positive value in tenths of MW)
- Term is fixed at one year, the name of the Period is **All**.
- Hedge type is fixed at Obligation
- Class is fixed at 24H

Submitting FTR offers to sell or bids to purchase in annual market

To submit a Buy bid or Sell offer FTR Quote into a round of the annual auction you must specify the round number and the annual market name.

The annual FTR Quote consists of the following data:

- Path designating source node to sink node
- MW Quantity (non-zero positive value in tenths of MW)
- Term is fixed at one year, the name of the period is **All** (other periods may be defined in the future).
- Hedge type is specified as Obligation or Option
- Class is specified as On Peak, Off Peak, or 24 Hour

- Offer or bid price must be non-zero positive value for Options but can be zero or negative for Obligations. Price specified in dollars per MW.

2.1.2 Monthly Market Quote Submittal

The monthly market is executed on a monthly basis per the PJM FTR Market Rules. A monthly market is a single round auction with a bidding period that extends over several days and a closure leading to market clearing. FTRs can be offered for sale or bid for purchase.

To submit a quote into the monthly market you must specify the market name. You *do not* specify a round. The FTR Quote consists of the following data:

- Path designating source node to sink node
- MW Quantity (non-zero positive value in tenths of MW)
- Term is variable (BOPP).
- Hedge type is specified as Obligation or Option
- Class is specified as On Peak, Off Peak, or 24 Hour
- Offer or bid price must be non-zero positive value for Options but can be zero or negative for Obligations. Price specified in dollars per MW.

2.1.3 Posting FTR Trades To Secondary Market

A secondary bilateral trading market is provided allowing participants to trade FTRs. This secondary market is open each day with settlement closure each day per the FTR Market Rules.

The secondary trading of FTRs provides the following support:

1. The participant may submit an offer to sell or a bid to purchase a given FTR. This is accomplished by submitting the offer or bid to be posted on the secondary market.
2. The participant may query the secondary market and obtain a list of all posted offers and bids.
3. The participant may accept a posted offer or bid at the stated price. By accepting an offer or bid, the poster is made aware of the pending deal. The poster and the acceptor can query the secondary market for pending deals.

4. The participant can approve a pending deal to trade an FTR.

5. The participant can query the secondary market for completed deals that have been confirmed.

6. The participant can query his activity log that describes all his operations executed on the secondary market.

To post an FTR quote to the secondary market, you must specify the trade type as a bid for purchase (**Buy**) or an offer to **Sell**. Also, you may split up an FTR into a specified interval or class or recombine FTRs as long as the total MW quantity owned on a given path is not violated. The start date of a given FTR trade cannot be earlier than the current date.

To post an FTR quote on the secondary market, the following data is specified:

- Path designating source node to sink node
- MW Quantity
- Hedge type is specified as Obligation or Option
- Class is specified as On Peak, Off Peak, or 24 Hour
- Offer or bid price is specified as a positive or negative value in dollars and cents per MW
- Interval of operation for the FTR specified using a start calendar day and an end calendar day.

Note: in order to sell an FTR of a given class, hedge type, an specified interval, you must be the owner of that FTR that covers the class, hedge type, and interval.

More information on trading using the secondary market is described in chapter 7 of this document.

2.1.4 Portfolio Management

A participant (i.e. a company) may create, update, and manage portfolios. A portfolio is a container of paths where each path is specified by its source and sink node. Paths are directional, so if both directions between two nodes are needed, two paths must be defined in the portfolio. All users registered for a given participant company may access all portfolios defined for the company.

Participants name portfolios when they are created and use the name whenever the portfolio is referenced via the XML or the Web browser interface. Portfolios exist primarily as an aid to the Web browser user but they can be used in concert with particular XML query messages.

To create a portfolio, you supply the following data and specify the action of Create:

- Portfolio name (which is unique to the participant)
- List of Path specifiers for the paths to be contained in the Portfolio

To replace an existing portfolio, you specify the action of Replace and supply the same type of data as in the portfolio create. This action will replace the entire contents of the existing portfolio with the contents of the submitted replace.

To remove an existing portfolio, you specify the action of Remove and specify the Portfolio name. You do not specify any Path declarations for the Portfolio. If you do, they will be ignored with no error report given.

To update an existing Portfolio by adding new paths or removing existing paths you specify the action of AddPath or RemovePath. You also specify the paths to be added or removed.

2.1.5 Querying Data

Data that resides in the FTR server may be queried. This includes both public data as well as private data. Also, any submitted data set, such as submitted quotes to annual or monthly markets, or submitted postings to the secondary market can be queried by transaction identifier.

The following data sets can be queried:

- FTRs submitted to annual market (self-scheduled and buy/sell quotes) by transaction identifier.
- FTRs submitted to the annual market by market name (includes all current FTRs). If query occurs prior to market close then the quotes are pending.
- FTRs posted to the secondary market can be queried by transaction ID.
- All current *pending* FTR quotes posted to the secondary market, all pending accepted deals, and all confirmed deals on the secondary market.
- The participant's current pending FTR quotes in a given market.
- Clearing prices can be queried by path (option paths) or by node (does not apply to options).
- Market results can be queried on a per market (and per annual round) basis. Bid and offer prices are only available to the participant that submitted the quote.
- Binding constraints (and the marginal value of the constraints) associated with a given market clearing.

- Total FTR capability can be queried.
- PJM Operator messages.
- Participant portfolio definitions.

2.2 Data Description

The following sections give data field type descriptions for each of the principle data fields involved in FTR transactions.

2.2.1 Common Data Types

The following table defines the common data types that are used throughout the data set descriptions that follow in subsequent sections. The descriptions here are the default behavior, if any override exists for subsequent data sets than it is described as specified for the data set. [Note: in the table below, elements are shown with <> brackets, attributes are shown without].

Data Field	Data Type	Data Description
market	String	Specifies the name of the market as an XML attribute. A market name is a string whose value is determined and published by PJM. PJM publishes the market name via its web interface and it is also available using the QueryMarketInfo request defined later in this document.
round	Integer	Specifies the auction round number for an annual market using an XML attribute on a number of messages. The auction round number values are allowed to be 1, 2, 3, and 4 only.
<Path>	Singleton Type	Specifies the FTR/ARR path from the specified source pricing node to the sink pricing node. Paths have direction from source to sink. The **source** and **sink** node names are specified as strings using the attribute names of source and sink.
<MW>	Number(8.1)	The quantity in MW associated with an FTR/ARR. MW must be greater than 0 and less than 9999999.9.

Data Field	Data Type	Data Description
<Price>	Number(10.2)	The price in $ per MW for an FTR/ARR quote when submitted as an offer to a market or posted to the secondary bilateral trading center. Price can be negative. Zero price is allowed.
<Period>	Character	The name of the term of an FTR
<Class>	Enumeration	An enumerated value designating the hourly interval associated with an FTR. Enumerated values are: **OnPeak**, **OffPeak**, **24H**.
<Hedge>	Enumeration	An enumerated value designating the FTR hedge type. Enumerated values are: **Obligation**, **Option**.
trade	Enumeration	An enumerated value specifying whether an FTR is an offer to sell or a bid for purchase. Enumerated values are: **Buy**, **Sell**, and **SelfScheduled**. The SelfScheduled value is only used in round 1 of the annual auction for converting an ARR into a price-taker FTR.
<Owner> <PostedBy> <AcceptedBy>	String(40)	The participant company identifier of the party trading FTRs and ARRs on the secondary market or the owner of an ARR.
<ID>	Character	The unique identifier of an FTR or an ARR traded on the secondary market. The <ID> is returned as a result of the QueryTrades request and it is used on subsequent submit of TradingAction messages.
<Interval>	Singleton Type	The interval is used when submitting quotes to the secondary market or when retrieving FTR ownership by query for the fractional days of ownership. The interval is specified by two attributes, the **start** calendar day and the **end** calendar day. The interval is inclusive of start and end dates.

Data Field	Data Type	Data Description
<Node>	String(30)	The name of a pricing node as used in various data sets that can be queried such as clearing prices by node.

2.2.2 FTR Quote

An FTR Quote is an offer to sell or a bid to buy entered into an annual or monthly market. The quote consists of the following fields. Note that some of the fields, as described apply only in certain conditions. The FTR layout for the secondary market and market results is slightly different as described later.

An self-scheduled FTR quote can be submitted to round one of the annual auction only. The total MW capacity of the quote is specified even though only 25 percent of that total capacity is cleared for each round. The trade type specified is **SelfScheduled** and the Price field is not used in this case. The selfscheduled FTR is a price taker.

Data Field	Data Description
ID	Unique value identifying a FTR.
Path	The path specifying the source and sink of the FTR.
Class	OnPeak, OffPeak, 24H (must be 24H for selfscheduled FTR quotes).
Period	The name of the term of the FTR .
Hedge	Obligation or Option. (Must be Obligation for selfscheduled FTR quotes).
MW	The quantity in MW associated with this FTR. MW must be greater than 0.
Price	The price in $ per MW for the FTR when submitted as an offer or a bid in a quote. Price can be zero or negative for obligations but options price is non-zero positive only. Price is not used for selfscheduled FTR quotes.

2.2.3 FTR Trading on Secondary Market

The data associated with an FTR traded on the secondary market consists of the following fields.

Data Field	Data Description
list	Name of secondary market list containing FTRs. The different lists include: Available, Accepted, Confirmed, Posted.
rightType	Designator of secondary trade as FTR. Used on the query for trades request.
ID	Secondary market identifier. This identifier is assigned when an FTR is posted for trading on the secondary market. This ID is used on subsequent queries, acceptance, and confirmation messages to uniquely identify a particular secondary traded FTR. Data type is opaque character string (that is, it is typed character but its value of of no importance other than it is unique).
PostedBy	The participant identifier for the original company that posted this FTR to the secondary market.
AcceptedBy	The participant identifier for the company that has accepted the FTR trade.
Confirmation	The date and time of the confirmation of the trade. A non-null value means that the trade has been confirmed.
Interval	The start day and the end day of the FTR trade. Start and End days are inclusive.
Path	The path specifying the source and sink of the FTR.
Class	OnPeak, OffPeak, 24H.
Hedge	Obligation or Option.
Period	The name of the term of the FTR.
MW	The quantity in MW associated with this FTR. MW must be greater than 0.

Data Field	Data Description
Price	The trade offer or bid price in $ per MW for the FTR. Price can be zero or negative.
ClearingPrice	A trade offer to sell must be an FTR cleared from a specific market with a given clearing price. This is the price associated with an FTRs clearing market for the path, hedge, and class type.

2.2.4 FTR Obligation Clearing Prices by Pricing Node

Clearing prices are published when a market clears. Clearing prices are associated with a specified market (annual, including rounds, or monthly). Prices though reported by node are only meaningful as the difference between the source and sink node. Clearing prices for FTR Options are defined on a path basis only.

Data Field	Data Description
Class	OnPeak, OffPeak, 24H
Period	The name of the term.
Node	The name of the pricing node associated with the clearing price.
Price	The Obligation price in $ per MW.

2.2.5 Obligation and Option Clearing Prices by Path

Clearing prices are published when an market clears. Clearing prices are associated with a specified market (annual or monthly). Clearing prices by path are designated using the source and sink pricing nodes. This information is another way of viewing and associating the clearing prices by node. Clearing prices are requested for a specified market. Obligation clearing prices are available by path as computed by subtracting the source LMP from the sink LMP values. Therefore, the information reported by path for Obligation prices is the essentially the same information available using the FTR Obligation clearing prices by pricing node mentioned above.

The option clearing prices are available by path only, there is no pricing node representation. The option clearing prices are computed and made available for a given set of option paths that are determined by PJM.

Data Field	Data Description
Path	The path of the FTR, source to sink.

Data Field	Data Description
Period	The name of the term.
PriceOnPeak	The onpeak clearing price for the given path.
PriceOffPeak	The offpeak clearing price for the given path.
Price24H	The 24 hour clearing price for the given path.

2.2.6 Market Results

The market results lists the FTRs cleared in a given auction. Market results are requested by auction (annual by year and round, monthly by month).

Data Field	Data Description
Path	FTR Path from source to sink.
Class	OnPeak, OffPeak, 24H.
Period	The name of the term.
Hedge	Obligation, Option.
BidMW	The offered/bid quantity in MW of the FTR.
ClearedMW	The cleared quantity in MW of the FTR.
BidPrice	The offered/bid price in $ per MW for the FTR.
ClearedPrice	The clearing price in $ per MW for the FTR.

2.2.7 Constraints

The binding constraint data set specifies the binding constraints associated with various contingencies. The binding constraint is associated with a cleared market (annual on a per round basis, or monthly).

Data Field	Data Description
Period	The name of the term.
Class	OnPeak, OffPeak. Note: constraints reported for OnPeak and OffPeak only.
Monitored	A String(60) value specifying the name of the monitored facility whose constraint is imposed by the contingency.
Contingency	A String(60) value specifying the name of the contingency whose occurrence results in the binding constraint. The base case is designated by a special contingency name.
MarginalValue	A price in dollars per MW of the marginal value of the specified constraint.

2.2.8 FTR Capability

The FTR capability is the estimated MW capacity available on subset of the known paths. The FTR capability is posted for each market (annual or monthly) and for each period (Winter, Summer, AllYear), and for OnPeak and OffPeak class values.

Data Field	Data Description
Path	The path, source to sink, of the FTR.
Period	The name of the term.
OnPeakMW	The On Peak interval's capability in MW. Specified using the MW common data type.
OffPeakMW	The Off Peak interval's capability in MW. Specified using the MW common data type.

2.2.9 Portfolio

The portfolio is offered primarily as an interactive web browser feature; although, the portfolio can be created, modified, and deleted via the XML programmatic API. A portfolio is named by the participant and associates an FTR path specified by source node and sink node. Paths have direction. A path from node A to node B is not the same as a path from node B to node A.

Data Field	Data Description

Data Field	Data Description
Portfolio Name	A participant chosen name for the portfolio. A String(40) value.
Path	One or more paths associated with the portfolio.

2.2.10 Messages

Messages are issued by the FTR market operator as needed.

Data Field	Data Type	Data Description
EffectiveDate	YYYY-MM-DD	The date of the effective start of the message.
TerminationDate	YYYY-MM-DD	The date of the end validity of the message.
Message Text	String(1024)	The message text, up to 1024 characters in free-format.

3 Introduction to the Programmatic API

The programmatic API is an XML based messaging protocol used for the exchange of messages between market participants and PJM. The XML message is formulated as a SOAP wrapped payload carried by the HTTP protocol.

The participant software must implement the messaging protocol according to the specification described in this document. This software implementation relies heavily on standard technologies that can be obtained freely on the Internet (i.e. open software) or that can be obtained from 3rd party vendors.

3.1 Messaging Overview

The programmatic API is a messaging API used by PJM FTR to exchange data with the FTR market participants. In function, the programmatic API mimics the interactive web user interface: all functions supported by the web pages are supported by the programmatic API.

3.1.1 The Role of the Participant

The participant uses the programmatic API to implement an automated interface to the FTR market services provided by PJM. Using the programmatic API, the participant can implement software that integrates efficiently with other applications and processes on the participant side of the network. This allows the necessary message exchange to fit more easily into the participant's business processes and methods.

The participant uses the programmatic API to:

- Submit FTR bids and offers into the annual and monthly markets.
- Query PJM for market results and other public and private data.
- Submit FTR bids and offers into the bilateral secondary market.

3.1.2 The Role of PJM

PJM provides services that support the programmatic API. These services are identified by URLs (Universal Resource Locator) and made available to the participant. When the PJM server receives a request at a given URL, the following processing is performed:

- The request is validated both syntactically and semantically. Any error results in an error response.
- If the request is a data submission then the data is validated per the FTR database constraints and per the business rules. Any violations result in an error response and the

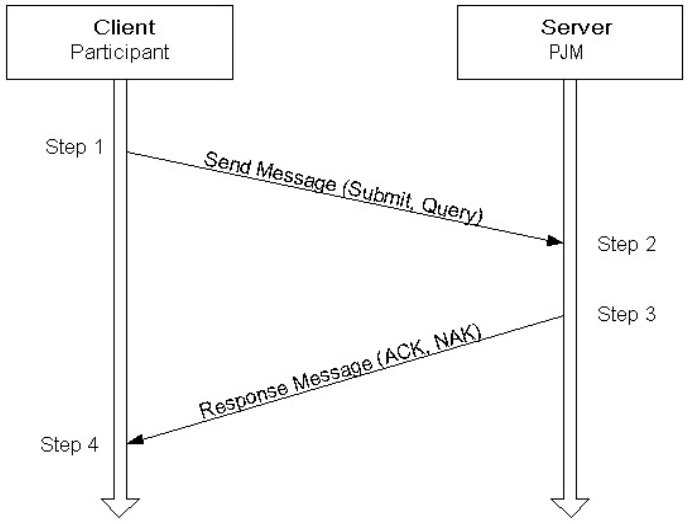
total rejection of all data composing the message. If there are no errors, then the data is entered into the FTR database.

- If the request is a data query then the message is dispatched to the known service handler for the requested packet. If the request is correctly formatted then the data requested is packaged as a SOAP wrapped XML payload and returned to the participant.

3.2 Request/Reply Messaging

The messaging protocol is called request/reply (or, request/response). The client issues a request to the PJM server and a reply (response) is returned by the server to the client. The client initiates all request/reply messages.

The following diagram illustrates the request/reply protocol.



The client is the market participant that is sending a request to the PJM server handling the message. The processing steps are outlined below:

Step 1 – the client software formulates the request message and sends it to the server. Request messages may be to submit FTR bids and offers or query the FTR market database. After sending the request, the client software waits for the response (step 4). Because the client software initiates the request and waits for the response, this is considered a *synchronous* messaging protocol.

Step 2 – the server software is actively listening for new messages and receives the message sent by the client. The server software processes the message which includes validating the message request, processing message action, and formulating the message response (step 3).

Step 3 – the server formulates the response as either an ACK (successful request) or a NAK (unsuccessful request resulting in error response). The successful request may be

a simple indicator of the success of the requested action or it may also include a data packet returned to the client as the response to the requested action (e.g. a data query). The unsuccessful response will always include an error packet that identifies each of the errors encountered in the request.

Step 4 – the client software receives the response message and acts accordingly depending on whether the response indicates success (ACK) or failure (NAK).

The PJM FTR server implements a message listener using a standard HTTP webserver that supports SSL for encryption and authentication[1]. Therefore, submitting a message request is tantamount to submitting an Internet web page request; although, instead of using the HTTP GET command, the HTTP POST command is used.

Therefore, the participant client must implement software that makes an HTTP connection using SSL along with the authentication scheme (username/password credentials). Once the HTTP connection is established the message is sent as a POST command to the server.

3.3 Web Technologies

As evident by the previous discussion, the programmatic API is implemented using standard *web* technologies. In particular, the following technologies are used by this scheme:

- XML, Extensible Markup Language
- SOAP, a web services API
- HTTP/HTTPS
- SSL and Authentication

3.3.1 XML – Extensible Markup Language

XML means eXtensible Markup Language. It is a structured, hierarchical, tag based coding language that is very suitable for describing data that is sent over the public Internet. There are many good references for learning about XML and a few are listed in section 1.3 ([2], [3]) of this document. The specification and details of XML will not be repeated here. However, a few comments on “why” and “how” XML is used is discussed briefly below.

[1] All message exchange between the participant and the PJM server uses HTTPS protocol which is HTTP over an SSL (encrypted) session. The name HTTP names the protocol commands, the suffix of S refers to the secure connection. When discussing the protocol, the name HTTP will be used.

XML Versus CSV

XML is a replacement for data file exchange using CSV format. CSV means *comma separated values* and refers to a commonly used ASCII coded file format where each line is a set of data values separated by a comma (or, some other field separator character such as a TAB). Each line typically refers to a set of associated data and given a particular interpretation by the producers and consumers of the data. The interpretation is usually ruled and defined by an external element that is often cryptic and obscure.

XML has the following advantages over CSV:

- XML is usually easily readable where each data value is described by a name. CSV often does not describe data by name.

- XML supports structure allowing nested or hierarchical relationships that can be validated for correctness. CSV is necessarily flat in structure often leading to the need for repeating groups of data to model data hierarchy.

- XML supports strong data typing (using XMLSchema language) that can be validated as part of the XML parsing process. Since parsers are standard, given the XMLSchema, an XML message can be parsed and validated easily by anyone. Custom validation software is not required (for example, standard tools such as XMLSpy[2] can be used to validate a message against the schema).

- XML supports data transparency where character data can be composed of any character code (assuming a few proper escapes where needed by the language). CSV on the other hand has trouble with embedded commas in character data if the comma is a field separator.

XML Schema

XML Schema is a relatively recent addition to the XML family of standards. XML Schema is a structured language used to describe other XML documents. XML Schema is itself an XML document described by an XML Schema.

XML Schemas are a replacement for the XML DTD (document type definition) which is still commonly used to describe XML. A DTD though lacks many of the features of XML Schema, such as strong data typing, name space management (DTDs do not support name spaces). XML Schema also supports a more powerful relationship and structure specification.

XML Schema is used to describe all FTR data exchange messages. DTDs are not used and they are not supported. There are many references describing XML Schema and how it is used. Those references already listed for XML ([2], [3]).

[2] XMLSpy is a third-party utility product used to create and validate XML schema and XML instance documents. XMLSpy is recommened as a tool for anyone using XML and is available from ALTOVA (see www.xmlspy.com or www.altova.com).

XML Namespaces

XML Namespaces are used to support the association of specific XML markup to a particular vocabulary and schema. Namespaces allow the mixture of XML markup from different vocabularies. In fact, using SOAP and XML Schemas require the use of namespaces and at least two vocabularies.

An XML Namespace is a unique identifier specified using a URI (uniform resource identifier). Thus, a namespace identifier looks like a URL or a web page address. However, an XML Namespace identifier is not a web page address and there is no expectation of finding any resource at the URI that is used. The combination of the unique domain name and the URI path establish the uniqueness that is required of namespace identifiers.

The use of XML Namespaces as shown later in this document are required. Failure to use the proper namespace will result in an XML validation failure and an error response. More information on XML Namespaces can be found in the XML references already cited.

3.3.2 SOAP – A Web Services API

SOAP means Simple Object Access Protocol. It is based on XML and is designed for the exchange of information in a distributed computing environment. The SOAP protocol was originally designed for RPC (remote procedure call) style of computing but it also supports other message exchange profiles. SOAP is not used as an RPC by the PJM FTR system; rather, it is used to wrap message payload in a common, standard, way to facilitate handling, routing, and common processing.

The SOAP protocol is an outside wrapper called the Envelope. This Envelope encloses all other XML data. The Envelope itself contains two distinct structures: Header and Body. The Header element is not used by this specification (it is considered optional by SOAP). They Body element contains the entire payload.

The SOAP standard used by this specification is SOAP 1.1. SOAP is under rapid development and is suffering from volatile change. In fact, the SOAP 1.2 recommendation is already out in draft form and may be a full recommendation by early to mid 2003[3].

There are a growing number of good references for SOAP, the one provided here (reference [4]) is just one recommendation.

Web-Services

Web Services is a defined set of technologies, tools, and standards (W3C recommendations) used to support a general service layer of information exchange. A formal Web Services application is described by a Web Services Description Language (WSDL) configuration file and used by some software tools for supporting a service connection.

[3] The term *recommendation* is the World Wide Web consortium's word for a *standard*. The word *standard* is not used because apparently only government entities can define standards and the W3C is not a government (not yet at least). We will mix the terms standard and recommendation freely in this document without fear or excuse.

SOAP is a protocol that is also defined as part of the Web Services standards. The W3C organization has published standards for the Web Services Description Language that is used in concert with SOAP and other technologies. However, this FTR application *is not* a formal Web Services application. A Web Services Description Language interface is not supported at this time.

3.3.3 HTTP/HTTPS

HTTP means Hyper-text Transport Protocol and HTTPS means Hyper-text Transport Protocol Secure. HTTP is the transport protocol used by the programmatic API (this specification) to carry the FTR XML messages over the Internet. HTTPS is the same protocol as HTTP except that it is a secure, encrypted session established by SSL. All the commands, headers, error responses, and body format for HTTPS is the same as HTTP. Therefore, in this and many documents, HTTP is generally used to refer to both HTTP and HTTPS with regard to the actual protocol rules. The HTTP standard is specified by RFC 2616 (reference [6]). HTTP 1.1 is used by this specification.

Although HTTP is the protocol for message exchange using the programmatic API, it is not the only method supported. File transfer is also supported for download. The File transfer format and naming rules is described later in this document.

3.3.4 SSL and Authentication

SSL means Secure Socket Layer and it is a protocol used to establish a secure, encrypted, TCP/IP session between the client and server computers. SSL is used for all data exchange using this programmatic API. SSL is not the authentication method though -- it merely provides the security for the authentication means.

SSL is an industry defacto standard developed by Netscape Corporation and it is the most commonly used and supported encryption protocol. Other protocols exist. The standard TLS is an IETF endorsed standard to replace SSL and thus is compatible with SSL. TLS means Transport Layer Security. SSL is referenced instead of TLS because it is still the industry dominant protocol used for establishing an encryption channel between client and server.

Authentication is the establishment and verification of the user submitting the requests to the PJM Server. Authentication is based on BASIC realm username and password authentication using the HTTP authorization header (explained in more detail later). The username and password are granted and managed by the PJM eSuite system. All users must be registered with eSuite to be authenticated for PJM FTR operations.

4 Common Principles

This chapter describes common requirements and messaging concepts used by all message types.

4.1 FTR Participant Registration

All FTR participants using the XML programmatic interface must be registered with PJM's eSuite system. Each user must have:

- A username that is specified on the authorization header of each HTTP message.
- A password that is specified on the authorization header of each HTTP message.
- Permission grant to access FTR public and private information. Authorization permissions are granted for: FTR public, FTR private read-only, and FTR private read and write.

4.2 FTR Server Addressing

All functions and data provided by the FTR Server are addressed using a Uniform Resource Locator (URL). Several URLs are defined as shown in the following table. Each message sent to the FTR server must use the appropriate URL. The appropriate URL is defined for each message type defined later in this specification.

Note: the actual host computer name shown below (ftr.pjm.com) is subject to change. Also, there may be more than one host supported by PJM for support of various test and sandbox functions.

Function	URL
Query Public or Private Data	**`https://ftr.pjm.com/ftr/xml/query`**
Submit Private Data	**`https://ftr.pjm.com/ftr/xml/submit`**

4.3 HTTP Command and Headers

All messages submitted to the PJM Server and all messages returned to the participant as responses are coded as HTTP POST command messages.

4.3.1 HTTP Request Message Format

The following example highlights the required format and content of the request message for both data submit and data query (although this example shows a query on private data as shown by the URI).

```
POST /ftr/xml/query HTTP/1.1
Host: ftr.pjm.com
Authorization: BASIC kjfpekmd3kjkj=
Content-Type: text/xml; charset=”UTF-8”
Content-Length: xxx
SOAPAction: “/ftr/xml/query”

<body content>
```

In this HTTP POST command example, the SOAP wrapped XML message (described below) would appear where the <body content> is shown.

The SOAPAction HTTP header describes the SOAP action or method to execute. This is included here but it is not used by the PJM FTR Server software[4]. The SOAPAction header maybe included by the participant but it is ignored by the PJM FTR server.

That gobbledegook string on the Authorization header represents the base64 coded username and password. The authorization realm required by this specification is the BASIC scheme as shown above on the Authorization header. The base64 encoded string is a coding of the username and password in the format of: *username:password*. That is, the username, followed by the : character, followed by the password. This string is then base64 encoded and specified on the Authorization header[5]. This header is required for all requests.

4.3.2 HTTP Response (reply) Message Format

The following example highlights the content of a typical response message.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=”UTF-8”
Content-Length: xxx

<body content>
```

The interpretation of the above lines is standard HTTP. The 200 status code indicates success. All responses returned by the PJM FTR system are considered successful responses using a 200 status code even if they are reporting an error in the content of the message. If an error should be raised by the PJM web server or if there is an authorization failure or bad URL then other HTTP response codes can be returned. Codes that may be returned include:

[4] The SOAPAction header is a requirement of SOAP 1.1, the current version of the SOAP standard but it is expected to be removed or made optional in SOAP 1.2.

[5] The base64 encoding schema simple and well documented by RFC 1421. Utilities for performing base64 encoding and decoding are freely available on the Internet (just do a google search on base64).

400 – bad request is returned if there is a formatting error in the headers or the HTTP command line.

401 – Unauthorized is returned if the request does not include an authorization header or if the authentication fails to authorize the given username and password.

404 – URI not found is returned if the URL specified on the request is unknown.

405 – Method not allowed is returned if any request to the specified URLs given any other command beside POST. Only the POST method is allowed.

500 – Internal Server Error is returned is some part of the server fails to respond or has crashed. See comment below on SOAP and the 500 error code.

It is possible that other error codes would be received but these are the most common likely to be seen during actual operation.

The SOAP standard requires that error code 500 be used in conjunction with any error message returned using the SOAP Fault response message. The main purpose of the SOAP Fault response message is to report failures due to exceptions, argument data type errors, and other RPC oriented problems. The PJM FTR implementation using SOAP is a message based request/reply exchange that does not suffer from the typical RPC type error handling and exception mechanism. Therefore, the SOAP Fault response message is not used and error code 500 is not reported from the PJM FTR server software.

4.4 SOAP Envelope

In the above HTTP request and response examples, the <body content> refers to the SOAP envelope. The SOAP envelope wraps all message content including query requests, query response, submittal request, submittal response, and error response.

The following example shows the canonical SOAP envelope used for all request and response messages. The HTTP header lines are representative of the HTTP structure (for a response message in this example), the body of the SOAP envelope contains the actual message using the following elements:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=”UTF-8”
Content-Length: xxx

<?xml version=”1.0”?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>

   --- actual message body goes here ---

</env:Body>
</env:Envelope>
```

There are other parts to the SOAP standard that are not shown by this example. For example, encoding style can be specified, other name spaces can be referenced.

The SOAP Header which is optional and may appear before the SOAP Body is not used. The Header element can be specified as an empty tag, as shown above and other parts of this document or it can be ignored on submission of data or query requests. The empty Header element is always included on XML response messages. These other parts to the SOAP schema are not used at this time by PJM FTR.

The SOAP Body element contains the XML Query Request or the XML Submit Request elements as documented in later chapters describing individual query and submit messages. When these elements are specified, it is recommended that the namespace required by the PJM FTR be specified as a global namespace operating from that position. For example,

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=”UTF-8”
Content-Length: xxx

<?xml version=”1.0”?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
   <SubmitRequest xmlns="http://eftr.pjm.com/ftr/xml">
     <FTRQuotes market="August2002">
           …other data…
     </FTRQuotes>
   </SubmitRequest>
</env:Body>
</env:Envelope>
```

4.5 XML Namespaces

Two namespaces are assumed and used by this specification. One namespace is used by SOAP and the other name space is used by the FTR XML Schema. These namespaces are:

SOAP:	**http://schemas.xmlsoap.org/soap/envelope/**
PJM XML:	**http://eftr.pjm.com/ftr/xml**

4.6 XML Schema

Two XML Schema references are required to successfully parse and validate the XML messages described by this document. One schema is for the SOAP 1.1 standard and the other schema is for the PJM FTR XML messages. They are:

SOAP:	**http://schemas.xmlsoap.org/soap/envelope/**
PJM XML:	**http://ftr.pjm.com/ftr/xml/schema**

The schema should never be located using the schemaLocation element inside of an XML message. All schemas used by the PJM FTR server are provided externally. The message content schema, if specified, is never used. ***Note that the PJM XML Schema location is subject to change. Please contact PJM for current URL used to locate the PJM XML Schema.***

4.7 Error Response

There are three kinds of response messages returned to the client: success response, success with data response, and error response. The individual message success and success with data responses are described in later sections. The error response is described in this section and is common to all request type messages.

The error response is given for the following types of problems:

- Malformed XML format, that is, unable to parse the XML correctly.
- Invalid XML message, that is, XML content does not validate against the schema. This includes many of the data type, range, relationship errors that can be found.
- Violation of business rules include errors such as market is not open, invalid names or data values, and so on.

When an error response is returned, the transaction request is rejected and not executed and the market database is not modified at all. For example, if a data submit message has an error in just one part of the overall message, the entire message and all of its data is rejected. No partial data submits are allowed or supported. If a query request has an error in any part of the request then no data is returned to the client even though other parts may be error free.

Note that if any of the HTTP errors are returned (as described previously) then there is no body returned with the message that is meaningful to this programmatic API (it might be some other body provided by an intermediary component that operates outside of this specification and therefore may be useful). All error responses described in this section are returned in a successful (`HTTP/1.1 200 OK`) message.

The error response message has the following SOAP wrapped XML format.

```
<?xml version=”1.0”?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
   <SubmitResponse xmlns="http://eftr.pjm.com/ftr/xml">
      <Error>
          <Code>ORA-20034</Code>
```

```
          <Text>Market is not open</Text>
          <Line>342</Line>
       </Error>
       …
   </SubmitResponse>
</env:Body>
</env:Envelope>
```

<Error> is used to specify a single error occurrence, there may be multiple Error elements returned in a single error response message.

<Code> is used to specify the error code (if one exists). The Code element is an optional part of the Error. Error codes may refer to specific software modules used by PJM FTR. The error codes are used for communication between participants and PJM when a particular problem occurs. In general, the error code is an abbreviation of the text message in <Text>.

<Text> is used to contain the error message text describing the problem. This is a required element and always returned.

<Line> is an optional line number that can be returned specifying the line where the problem occurred. Not all submittal methods use the notion of a line so this value is not always available and it may not be useful. It depends on the nature of the submitted data.

4.8 Transaction Semantics

Each submittal is executed using a transaction semantic that governs all of the processing and uniquely identifies the submitted data set for future operations. This is slightly different than a typical database transaction although the database transaction supporting commit and rollback semantics is part of this scheme.

4.8.1 Transaction Semantics and Transaction Identifier

Whenever a submittal request message is processed and is successful, resulting in the modification of the database, a transaction event is created and uniquely identified by a transaction identifier. This transaction identifier is returned as part of the Success response message. This transaction identifier is used only for data submittal, data queries do not produce transaction identifiers and none are returned.

Every database row modified as a result of the submit transaction is marked as associated with the transaction event. These transaction events and the associated database row identifiers are used by analysts for administration and analysis purposes. These transaction events and identifiers are also used by participants to perform certain operations as described below.

4.8.2 Transaction Log

The resulting transaction event *log* maintains a complete history of all changes to the database, whether initiated by the web user or an XML message submit request. This log is available for review via the web interface. The fields supported include the following:

- Transaction ID
- Participant ID of the owner of the data set
- User name of the submitter (or, web user)
- Time and Date of Transaction Event
- Type of Event: Web Submit or XML Submit
- Type of data associated with the transaction (FTR Quote, Secondary Market Quote, etc.)
- Number of rows of data associated with the Transaction (may be zero).
- File that contains the message data of the original submit (optional).
- Market owning the transaction event.

4.8.3 Query By Transaction

Data can be queried by transaction. Using the QueryByTransaction message (as described in chapter 8 of this document), a query request can be executed to return all parts of the message submitted by a previous SubmitRequest message. You can query by transaction at any time as long as the data is still available in the database[6]. Query by transaction is supported by the XML interface only.

4.8.4 Delete By Transaction

Data can be deleted by transaction but only if the data has not yet been used by market operations. Delete by transaction is supported by the XML interface only. For the PJM FTR application, you can delete by transaction only under the following conditions:

- If transaction identifies quotes submitted to an annual market (any given round) or a monthly market, you can delete the transaction only if the market bidding is still open. Once bidding for the market closes, you cannot delete the transaction.
- Submitted data that is part of the secondary trading system cannot be deleted by transaction. You can query this data by transaction but you cannot delete it. The reason is that secondary trading data is processed immediately upon validation. There is no window of opportunity to delete such a transaction.
- You cannot delete any transaction that itself includes a DeleteByTransaction request.

[6] PJM System Administration may roll out or archive portions of the database after a given retention period. Data so archived is no longer available via QueryByTransaction.

- You may not delete portfolio create, update, or delete operations by transaction.

The delete by transaction operation should be considered a yank function. That is, you submitted a quote or set of quotes into the database and you want to yank back those submitted quotes before they are processed.

4.8.5 Replacing By Transaction

There is no ReplaceByTransaction message. Instead, if you want to replace the data submitted and identified by a transaction identifier you do the following steps prior to market close:

1. Use QueryByTransaction to obtain the data submitted by the transaction you want to replace.

2. Delete the data identified by the transaction identifier.

3. Make changes to the data set returned in response to the query in step 1 and resubmit as a normal quote SubmitRequest.

4.9 XML Data Type and Specification Semantics

In general, data specified using the XML Schema rules is bound by very tight data type and constraining facets. However, some fields require special care in how they are specified since the user has complete freedom to specify any characters he wishes and some of these cause problems for XML or have dubious interpretations.

4.9.1 Character Case

Character case, upper, lower, and camel case must be honored by all XML specified in this document. Within the body of the XML, there are no case insensitive strings. Case must be honored at all times. For example:

- The values of the Trade enumeration are **Buy** and **Sell**. Case must be honored, spellings such as *BUY, buy, sell, SELL,* will be rejected.

- The values of portfolio names, owner names, node names are case sensitive.

- Attribute names, such as **source** and **sink** on the **Path** element or **start** and **end** on the **Interval** element must be spelled honoring character case.

4.9.2 XML Boolean Data

Boolean data is a recognized type defined by the XML Schema. Boolean is binary data represented as true or false. The Boolean values acceptable for true and false are shown in the table below:

State	XML Representations
True	true 1
False	false 0

For example, if a Boolean element called <Veracity> were specified as input, each of the examples below would represent a true value being submitted:

```
<Veracity>true</Veracity>

<Veracity>1</Veracity>
```

4.9.3 XML Date and Time

All date and time values must follow the XML Schema data type rules for date and time. The formats used by this specification include:

date	YYYY-MM-DD (example: 2003-02-03)
dateTime	YYYY-MM-DDTHH:MM:SS (example: 2003-02-03T12:30:00)
dateTime with Zone	YYYY-MM-DDTHH:MM:SS.sss+HH:MM (example: 2003-02-03T12:30:00.000-05:00)

The first DateTime format shown above is the default format used for any input operations. Currently, there are no requirements to input a date and time field. The second format, showing the time zone is used on output, such as the confirmation date and time of a secondary bilateral trade or the Market interval date and times on the MarketInfo element.

The XML Schema data type description as documented by the W3C standards group should be consulted for the exact representation and handling of the dateTime data type. Briefly, the above fields are interpreted in the following manner:

YYYY	The year as a 4 digit number such as 1999, 2000, 2001, and so on. Abbreviations using 2 digits are not allowed.
MM	The month as a 2 digit number ranging from 01 through 12. January is represented by 01, February by 02, and so on. Leading 0 for

	months 01 through 09 must be specified.
DD	The day of the month as a 2 digit number ranging from 01 through 31. The leading 0 for days 01 through 09 must be specified.
T	A literal "T" value that acts as the separator between calendar day and the clock time values.
HH	The hour of the day as a 24 hour clock using 2-digits from 00 through 23. Leading 0 must be specified for hours 00 through 09.
MM	The minute of the hour as a 2-digit value ranging from 00 through 59. Leading 0 must be specified for minutes 00 through 09.
SS SS.sss	The second of the minute as a 2-digit value ranging from 00 through 60. Leading 0 must be specified for seconds 00 through 09. The second form includes the fractions of a second shown as SS.sss where the .sss fractional part is any fractional value of arbitrary precision. The value used with this specification on output is .000. Note that the meaning of 60 seconds or more is useful for special leap second days of the year and this feature is not used by this specification.
+	This field represents a + (plus) or a – (minus) sign to separate the time zone offset from GMT. A minus sign is used for those times west of the prime meridian and therefore lag GMT time.
HH:MM	This field to the right of the + or – sign of the time zone offset indicates the time zone offset in hours. The value is represented as whole number of hours in this specification (that is, the MM field is zero).

The date and time format rules above are part of the XML standard. However, this specification will use .000 for the sub-second offset and for time zone only whole hours are used.

All time zones default to Eastern Prevailing Time.

4.9.4 XML Character Data and Markup and Escaped Characters

XML text consists of intermingled character data and markup. Markup takes the form of start-tags, end-tags, empty-element tags, entity references, character references, comments, and CDATA section delimiters. Anything that is not markup is called *character data* of the XML document.

The ampersand character (&) and the left angle bracket (<) may appear in their literal form only when used as markup. These characters cannot appear in their literal form within *character data*.

If these characters are needed within character data, they must be escaped using either numeric character references or the strings "&" and "<" respectively for & and <. The right angle bracket may appear in character data in some circumstances but it is best to escape this character using ">" or the numeric escape equivalent.

For example, if an element calls for the specification of character data for a name for instance, that name cannot include any of the characters mentioned above. Consider the following:

`<name>MW&MVAR</name>`	This is illegal because the & sign must be escaped. The result is that the parser will see the & as a escape which means that the next few characters are sucked in and interpreted as special characters. The result is that the < and the / are eaten and the resulting XML document is not well-formed.
`<name>MW&MVAR</name>`	This is the correct specification when the & sign is needed.
`<name>MW and MVAR</name>`	This is better yet, avoid the & character altogether.

The XML specification should be consulted for a more detailed explanation of the differences in handling character data, markup, entities, entity references and so on. Our motivation though is to create an XML Schema that will not result in any of these situations however that assumes that the user always does the right thing. This explanation is for those users who get caught doing the wrong thing.

4.9.5 XML Null Data

Sometimes it might be useful to submit a null value into the database[7]. This is allowed for only certain data fields defined by the XML Schema but the method of submitting a null is to submit an empty element. For example, the following two lines specify an empty element:

```
<EmergencyLimit/>

<EmergencyLimit></EmergencyLimit>
```

These two examples are considered equivalent, the result is the same. They specify an element by name, EmergencyLimit, but do not specify any character data. If the XML Schema defines

[7] At this time, the message format do not define any data fields where a null value can be entered.

non-null character data than this representation is interpreted to submit a null value. This null value overwrites any existing value in the database for that field with a null.

Constrast this handling of null data with an absent element described next.

4.9.6 XML Absent Elements

Some elements are optional. They may appear in the XML document or they may not. If they do not appear, the default handling is usually governed either by the XML Schema definition or the application. If the XML Schema specifies a default value than that value is substituted just as if the element were specified with that value.

If there is no default value associated with the element than the application treats this as if the element did not exist. That is, if the operation is some kind of replace of existing data and the database field reflecting the missing element is left alone. It is not modified in any way. It is not set to null. If there is an existing value, it is retained.

4.10 File Upload and Download Format

The file upload and download XML format is identical to the message body of the HTTP query request, submit request, or response. The only missing lines are those specifying HTTP headers. The following example shows a query response as an HTTP message and the same query response formatted as a file.

First, the HTTP message query response[8]:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=”UTF-8”
Content-Length: xxx

<?xml version=”1.0”?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
   <QueryResponse xmlns:mkt="http://eftr.pjm.com/ftr/xml">
      <ClearingNodePrices market=”August2003”>
          <NodePrice>
            <Node>xxx</Node>
            <Class>24H</Class>
            <Price>8.50</Price>
          </NodePrice>
          <NodePrice>
            <Node>yyy</Node>
            <Class>OnPeak</Class>
            <Price>20.52</Price>
          </NodePrice>
      </ClearingNodePrices>
   </QueryResponse>
</env:Body>
</env:Envelope>
```

[8] XML example is shown for illustrative purposes only, it does not necessarily refer to an actual message with realistic data.

The equivalent file download format appears as shown below:

```
<?xml version=”1.0”?>
<env:Envelope xmlns:env=”http://schemas.xmlsoap.org/soap/envelope/”>
<env:Header/>
<env:Body>
   <QueryResponse xmlns:mkt="http://eftr.pjm.com/ftr/xml">
      <ClearingNodePrices market=”August2003”>
          <NodePrice>
            <Node>xxx</Node>
            <Class>24H</Class>
            <Price>8.50</Price>
          </NodePrice>
          <NodePrice>
            <Node>yyy</Node>
            <Class>OnPeak</Class>
            <Price>20.52</Price>
          </NodePrice>
      </ClearingNodePrices>
   </QueryResponse>
</env:Body>
</env:Envelope>
```

Line breaks may be used to format the response message for readability and processing but are not part of the message content. There is no meaning to a line break in the XML message. Also, all line breaks are issued as single byte LF codes (the typical Unix style of line break, the Microsoft Windows style of CRLF are not used).

The file is named according to the type of data but this name can easily be changed by the user as part of the download process therefore not much substance is given to the file names.

Therefore, to submit data for upload, the very first line in your message is the <?xml version="1.0"?> line following by the SOAP envelope and the content of the Body.

5 FTR Data Query

The Data Query is the request method used to receive predefined data sets from the PJM FTR system. Data query is equivalent to displaying a web page.

There are two types of data that can be queried: public and private. Public data is available to all registered participants, private data is only available to the participant that owns the data. Ownership is determined by participant identification which is obtained via the authentication credentials of the request message. Each user specified by the username operates as an agent for the participant who owns the data and may access such private data.

Queries for secondary trading data is covered in chapter 7 of this document.

When making a request for private data, you do not explicitly give your participant identity – that would be a serious security risk. Instead, the participant identity is derived from the username. However, other parameters affecting the data request can be specified in the query. These parameters may include the date, auction type, portfolio, and so on.

The data query is initiated using a query request element as shown below. This query request is the payload that is enclosed by the SOAP Body element. Only one such query request element can be specified. As shown, the query request encloses individual query elements that are unique to the query type. There can be multiple query elements given in one query request message.

Example of query request for clearing option prices for the three given auctions (aka markets):

```
<env:Envelope xmlns:env=”http://schemas.xmlsoap.org/soap/envelope/”>
<env:Header/>
<env:Body>
   <QueryRequest xmlns="http://eftr.pjm.com/ftr/xml">
      <QueryOptionPrices market="August2002"/>
      <QueryOptionPrices market="September2002"/>
      <QueryOptionPrices market="October2002"/>
   </QueryRequest>
</env:Body>
</env:Envelope>
```

The elements that are children of <QueryRequest> can appear in any order. You may repeat a given element using different qualifier values. The response elements returned under <QueryResponse> listed below are returned in the same order as requested.

The response message is always returned as a SOAP wrapped payload and each individual query request response data is returned in the same order that the requests were made. Some requests may result in a null data set on return, in those situations, the containing element is always returned as documented in for each message.

For example, the response packet for the above example may appear as shown below.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<env:Header/>
<env:Body>
   <QueryResponse xmlns:mkt="http://eftr.pjm.com/ftr/xml">
      <OptionPrices market="August2002">
          -- Option clearing prices data in response to request --
      </OptionPrices>
      <OptionPrices market="September2002">
          -- Option clearing prices data in response to request --
      </OptionPrices>
      <OptionPrices market="October2002"/>
   </QueryResponse>
</env:Body>
</env:Envelope>
```

In the above example, the market for October2002 is not cleared so there are no prices available and the null set is returned.

The following example shows an error response when the query goes bad. In this example, we illustrate an error response when the market name on a query is not correct:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
   <QueryResponse xmlns:mkt="http://eftr.pjm.com/ftr/xml">
      <Error>
         <Code>Blue</Code>
         <Text>Market name is unknown: ziklag2002</Text>
      </Error>
   </QueryResponse>
</env:Body>
</env:Envelope>
```

All query requests must be issued to the following URL (Note: actual host computer name is subject to change, consult PJM for actual name):

```
https://ftr.pjm.com/ftr/xml/query
```

If you choose to include the SOAPAction HTTP header, it must be specified using the query action URI which is:

```
/ftr/xml/query
```

Each of the following sections describes the query request and the query response for each type of query defined.

5.1 Query for FTR Quotes (Private)

5.1.1 Purpose

This message format is used to query for FTR quotes submitted to a specified market.

To query for the FTR quotes, you must specify the market name.

5.1.2 Message Format

The FTR Quotes query is shown below:

```
<QueryRequest>
  <QueryFTRQuotes market=”xx” round=”xx”>
    <All/>
    <Path source=”xxx” sink=”yyy”/>
    <PorfolioName>aaa</PortfolioName>
    <ID>xxx</ID>
  </QueryFTRQuotes>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryFTRQuotes>	Complex Type	Specifies the query for the FTR quotes submitted to the named market. This market may be open or it can be a past market that has cleared and closed.
market	Character	Required field specifying the name of the market for the desired FTR Quotes. Market names are like Year2002, August2004.
round	Numeric(1)	The numeric signature of the round as 1, 2, 3, 4. This is required for annual markets only. If specified for a monthly market, it is ignored.
<All/>	Null	Optional field specifying that all quotes owned by the participant for the market are to be queried. At least one and only one element of the set <All/>, <Path>, and <Portfolio> can be specified.

Element or Attribute	Data Type	Description
<Path>	Singleton Type	Optional field to specify a desired path as a parameter for the query. At least one and only one element of the set <All/>, <Path>, and <Portfolio> can be specified.
source	Character	Source node name.
sink	Character	Sink node name.
<PortfolioName>	Character	Optional field to specify the query to be of paths defined by this named portfolio. At least one and only one element of the set <All/>, <Path>, and <Portfolio> can be specified.
<ID>	Numeric(15)	Unique identifier for the FTR.
<Period>	Character	Optional element specifying the period. Occurs 0 or 1 times.

5.1.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <FTRQuotes market=”xxx” [ round=”xx” ] >
    <FTRQuote trade=”xxx”>
      <ID>xxx</ID>
         <Path source=”xxx” sink=”yyy”/>
      <Class>xxx</Class>
      <Period>xxx</Period>
      <Hedge>xxxx</Hedge>
      <MW>999.9</MW>
      <Price>999.99</Price>
    </FTRQuote>
  </FTRQuotes>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<FTRQuotes>	Complex Type	The root element containing all FTR Quotes for this market. Occurs 0 to many times.
market	Character	Specifies the name of the market.
round	Numeric(1)	Specifies the round signature number, used for annual markets only. Does not appear in monthly markets.
<FTRQuote>	Complex Type	Used to specify a single quote on a given path. Occurs 0 to many times.
trade	(Buy, Sell, SelfScheduled)	Specifies whether this quote is an bid to buy or an offer to sell or self-scheduled.
<ID>	Numeric(15)	Uniquly identifies the FTR.
<Path>	Singleton Type	Specifies the FTR path as a source and sink.
source	Character	Specifies the path source pricing node.
sink	Character	Specifies the path sink pricing node.
<Class>	(OnPeak, OffPeak, 24H)	Specifies the class of the FTR.
<Period>	Character	Specifies the period of the FTR.
<Hedge>	(Obligation, Option)	Specifies the hedge type of the FTR.
<MW>	Numeric (8.1)	Specifies the MW quantity (bid-in) of the FTR. Occurs just once.
<Price>	Numeric(10.2)	Specifies the offer price if selling or the bid price if buying. Occurs just once.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.
- Transaction Identifier does not exist.

5.2 Query for Obligation Clearing Prices By Node (Public)

5.2.1 Purpose

This message format is used to query a market obligation clearing prices. Clearing prices are queried for a particular market and by price node (clearing prices can also be requested by path using another query request. Prices are not available for options using this query method. Query prices can also be requested by node using a portfolio where the nodes included in the report is the union of the nodes specifying paths in the portfolio.

5.2.2 Message Format

The Obligation Clearing Prices by Node Query request is shown below:

```
<QueryRequest>
  <QueryNodePrices market=”xxx” [round=”xx”] >
    <All/>
    <Node>xxx</Node>
    <PortfolioName>xxx</PortfolioName>
  </QueryNodePrices>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryNodePrices>	Complex Type	Specifies the query for a particular market. Occurs 0 to many times.
market	Character	Required field specifying the name of the market.
round	Numeric(1)	Specifies the round signature number. Only used for annual markets.

Element or Attribute	Data Type	Description
<All/>	Null	Optional element specifying that all nodes with clearing prices are to be returned. At least one and only one element of the set <All/>, <Node>, and <Portfolio> can be specified.
<Node>	Character	Optional element specifying particular pricing nodes. If not specified (and, if no portfolio is specified) then the nodes defined by PJM portfolios are included in the report. At least one and only one element of the set <All/>, <Node>, and <Portfolio> can be specified.
<PortfolioName>	Character	Optional element specifying the nodes as the union of all paths defined for the Portfolio. At least one and only one element of the set <All/>, <Node>, and <Portfolio> can be specified.
<Period>	Character	Optional element specifying the period. Occurs 0 or 1 times.

5.2.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <ClearingNodePrices market=”xxx” [round=”xx”] >
    <NodePrice>
      <Node>xxx</Node>
      <Class>xxx</Class>
      <Period>xxx</Period>
      <Price>99999.99</Price>
    </NodePrice>
  </ClearingNodePrices>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<ClearingNodePrices>	Complex Type	Element that contains the clearing prices by node for the specified market. Occurs 0 to many times.
market	Character	Specifies the name of the market.
round	Numeric(1)	Specifies the annual auction round. Used only with annual market queries.
<NodePrice>	Complex Type	Specifies the clearing price at a given node, for a given class and term. Occurs 0 to many times.
<Node>	Character	Specifies the pricing node name.
<Class>	(OnPeak, OffPeak, 24H)	Specifies the class of the clearing result prices.
<Period>	Character	Specifies the period for the computed clearing price.
<Price>	Numeric(10.2)	Specifies the FTR market clearing price in $ per MW.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.
- Error reported if the market has not cleared.
- Error reported if a specified node does not exist or if a specified portfolio does not exist.

5.3 Query for Obligation Clearing Prices (Public)

5.3.1 Purpose

This message format is used to query a market obligation clearing prices by path. Clearing prices are queried for a particular market and by path or portfolio.

5.3.2 Message Format

The Clearing Prices by Path Query request is shown below:

```
<QueryRequest>
  <QueryObligationPrices market=”xxx” [round=”xx”]>
    <Path source=”xxx” sink=”xxx” />
    <PortfolioName>xxx</PortfolioName>
  </QueryObligationPrices>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryObligationPrices>	Complex Type	Specifies the query for a particular market.
market	Character	Required field specifying the name of the market.
round	Numeric(1)	Specifies the annual auction round signature (1,2,3,4).
<Path>	Singleton Type	Optional element that specifies a path to receive the net clearing price. Net clearing price is sink node price minus source node price. Must choose query using <Path> or query using <PortfolioName> but not both.
source	Character	Required field specifies the path

Element or Attribute	Data Type	Description
		source node.
sink	Character	Required field specifies the path sink node.
<PortfolioName>	Character	Optional element specifying all paths defined for the Portfolio. Must choose query using <Path> or query using <PortfolioName> but not both.
<Period>	Character	Optional element specifying the period. Occurs 0 or 1 times.

5.3.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <ObligationPrices market=”xxx” [round=”xxx”]>
    <ObligationPrice>
      <Path source=”xxx” sink=”xxx”/>
         <Period>xxx</Period>
      <PriceOnPeak>999.99</PriceOnPeak>
      <PriceOffPeak>999.99</PriceOffPeak>
      <Price24H>999.99</Price24H>
    </ObligationPrice>
  </ObligationPrices>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<ObligationPrices>	Complex Type	Element that contains the obligation clearing prices by path for the specified market. Occurs 0 to many times.
market	Character	Specifies the market name.
round	Numeric(1)	Specifies the annual auction round signature (1,2,3,4). Not used for

Element or Attribute	Data Type	Description
		monthly auction markets.
<ObligationPrice>	Complex Type	Specifies the obligation clearing prices on the specified path.
<Path>	Complex Type	Specifies the path between source and sink pricing nodes.
source	Character	Specifies the path source node name.
sink	Character	Specifies the path sink node name.
<Period>	Character	Specifies the period name.
<PriceOnPeak>	Numeric(10.2)	Specifies obligation clearing price for OnPeak class.
<PriceOffPeak>	Numeric(10.2)	Specifies obligation clearing price for OffPeak class.
<Price24H>	Numeric(10.2)	Specifies obligation clearing price for 24H class.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.
- Error reported if the market has not cleared.
- Error reported if a specified path does not exist or if a specified portfolio does not exist.

5.4 Query for Option Clearing Prices (Public)

5.4.1 Purpose

This message format is used to query a market option clearing prices. Option clearing prices are queried for a particular market.

5.4.2 Message Format

The Clearing Prices by Path Query request is shown below:

```
<QueryRequest>
  <QueryOptionPrices market=”xxx” [round=”xx”]/>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryOptionPrices>	Complex Type	Specifies the query for a particular market.
market	Character	Required field specifying the name of the market.
round	Numeric(1)	Specifies the annual auction round signature (1,2,3,4).
<Period>	Character	Optional element specifying the period. Occurs 0 or 1 times.

5.4.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <OptionPrices market=”xxx” [round=”xxx”]>
    <OptionPrice>
      <Path source=”xxx” sink=”xxx” />
      <Period>xxx</Period>
      <PriceOnPeak>999.99</PriceOnPeak>
      <PriceOffPeak>999.99</PriceOffPeak>
      <Price24H>999.99</Price24H>
    </OptionPrice>
  </OptionPrices>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<OptionPrices>	Complex Type	Element that contains the option clearing prices by option path for the specified market. Occurs 0 to many times.
market	Character	Specifies the market name.
round	Numeric(1)	Specifies the annual auction round signature (1,2,3,4). Not used for monthly auction markets.
<OptionPrice>	Complex Type	Specifies the option clearing prices on the specified path.
<Path>	Complex Type	Specifies the option path.
source	Character	Specifies the path source node name.
<Period>	Character	Specifies the period name.
sink	Character	Specifies the path sink node name.
<PriceOnPeak>	Numeric(10.2)	Specifies option clearing price for OnPeak class.
<PriceOffPeak>	Numeric(10.2)	Specifies option clearing price for OffPeak class.
<Price24H>	Numeric(10.2)	Specifies option clearing price for 24H class.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.
- Error reported if the market has not cleared.

- Error reported if a specified path does not exist or if a specified portfolio does not exist.

5.5 Query for ARR Requests

5.5.1 Purpose

This message format is used to query for the ARR Requests. The query request is by ARR market. The query response is the set of ARR Requests for the participant. Each ARR request is specified by sink, source, (bid) MW, and capability MW value.

5.5.2 Message Format

The ARR Requests Query is shown below:

```
<QueryRequest>
  <QueryARRRequests>
     <MarketName>xxxx</MarketName>
     <MarketRoundName>yyyyy</MarketRoundName>
     <SinkZoneName>qqqqq</SinkZoneName>
     <SinkName>ssssss</SinkName>
  </QueryARRRequests
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryARRRequests>	Singleton Type	Specifies the query for the ARR requests.
<MarketName>	Character	Required field specifying the name of the market.
<MarketRoundName>	Character	Required field specifying market round.
<SinkZoneName>	Character	Required field, to filter requests based on the specified sink zone.
<SinkName>	Character	Required field for stage-1 and optional for stage-2, to filter requests based on the specified sinkname.

5.5.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <ARRRequests>
    <ARRRequest>
      <SinkName>xxxx</SinkName>
      <SourceName>xxxx</SourceName>
      <BidMW>999</BidMW>
      <CapabilityMW>999.9</CapabilityMW>
    </ARRRequest>
  </ARRRequests >
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<ARRRequests>	Complex Type	Element that contains the ARR Requests. Occurs 0 to 1.
<ARRRequest>	Complex Type	Specifies the ARR Request for the specified sink. Occurs 0 to many times.
<SinkName>	Character	Specifies the path sink node name.
<SourceName>	Character	Specifies the path source node name.
<BidMW>	(MWType)	Specifies the requested (bid) ARR allocation for the sinkzone in MW.
<CapabilityMW>	(MWType)	Specifies the capability ARR for the sinkzone in MW.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified ARR request market does not exist.

5.6 Query for ARR Results

5.6.1 Purpose

This message format is used to query for the ARR Results. The query request is by ARR market. The query response is the set of ARR Results for the participant. Each allocated ARR is specified by sink, source, (bid) MW, and allocated (cleared) MW value.

5.6.2 Message Format

The ARR Results Query request is shown below:

```
<QueryRequest>
  <QueryARRResults>
     <MarketName>xxxx</MarketName>
     <MarketRoundName>yyyyy</MarketRoundName>
     <PortfolioName>qqqqq</PortfolioName>
  </QueryARRResults
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryARRResults>	Singleton Type	Specifies the query for the ARR results.
<MarketName>	Character	Required field specifying the name of the market.
<MarketRoundName>	Character	Required field specifying market round.
<PortfolioName>	Character	Optional field, to filter results based on the specified portfolio name.

5.6.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <ARRResults>
```

```
    <ARRResult>
      <ID>99999</ID>
      <SinkName>xxxx</SinkName>
      <SourceName>xxxx</SourceName>
      <BidMW>999</BidMW>
      <ClearedMW>999.99</ClearedMW>
    </ARRResult>
  </ARRResults >
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<ARRResults>	Complex Type	Element that contains the ARR Results. Occurs 0 to 1.
<ARRResult>	Complex Type	Specifies the ARR Results for the specified sink and source. Occurs 0 to many times.
<ID>	Number	Specifies id for the results
<SinkName>	Character	Specifies the path sink node name.
<SourceName>	Character	Specifies the path source node name.
<BidMW>	(MWType)	Specifies the requested (bid) ARR allocation for the sink and source in MW.
<ClearedMW>	(MWType)	Specifies the allocated (cleared) ARR allocation for the sink and source in MW.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified ARR results market does not exist.

5.7 Query for ARR Constraints

5.7.1 Purpose

This message format is used to query for the ARR Constraints. The query request is by ARR market. The query response is the set of ARR Constraints for the participant.

5.7.2 Message Format

The ARR Constraints Query request is shown below:

```
<QueryRequest>
  <QueryARRConstraints>
     <MarketName>xxxx</MarketName>
     <MarketRoundName>yyyyy</MarketRoundName>
  </QueryARRConstraints>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryARRConstraints>	Singleton Type	Specifies the query for the ARR Constraints.
<MarketName>	Character	Required field specifying the name of the market.
<MarketRoundName>	Character	Required field specifying market round.

5.7.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <ARRConstraints>
    <ARRConstraint>
      <ConstraintName>xxxxx</ConstraintName>
      <ContingencyName>xxxx</ContingencyName>
    </ARRConstraint>
  </ARRConstraints>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<ARRConstraints>	Complex Type	Element that contains the ARR Constraints. Occurs 0 to 1.
<ARRConstraint>	Complex Type	Specifies the ARR Constraint for the specified path. Occurs 0 to many times.
<ConstraintName>	Character	Specifies the constraint name
<ContingencyName>	Character	Specifies the contingency name.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified ARR market or round does not exist.

5.8 Query for ARR Sinks

5.8.1 Purpose

This message format is used to query for the ARR Sinks. The query request is by participant, ARR market and round.

5.8.2 Message Format

The ARR Sinks Query request is shown below:

```
<QueryRequest>
  <QueryARRSinks>
    <MarketName>xxxx</MarketName>
    <MarketRoundName>xxxx</MarketRoundName>
  </QueryARRSinks>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryARRSinks>	Singleton Type	Specifies the query for the ARR Sinks.
<MarketName>	Character	Required field specifying the name of the market.
<MarketRoundName>	Character	Required field specifying market round.

5.8.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<ARRSinks>
  <ARRSink marketName="xx" marketRoundName="xx"
sinkZoneName="xx"sinkName="xx">
    <BidMW>9999999.999</BidMW>
    <CapabilityMW>9999999.999</CapabilityMW>
    <NetworkServicePeakLoad>9999999.999</NetworkServicePeakLoad>
  </ARRSink>
</ARRSinks>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<ARRSinks>	Complex Type	Element that contains the ARR Sinks. Occurs 0 to 1.
<ARRSink>	Complex Type	Specifies the ARR Constraint for the specified path. Occurs 0 to many times.
<BidMW>	MWType	Specifies the total Bid MW for the sink.
<CapabilityMW>	MWType	Specifies the Capability MW for the sink.
<NetworkServicePeakLoad>	MWType	Specifies the Network Service Peak Load for the sink.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified ARR market or round does not exist.

5.9 Query for Market Period

5.9.1 Purpose

This message format is used to query market period market period and market interval for the period.

5.9.2 Message Format

The Market Info Query request is shown below:

```
<QueryRequest>
  <QueryMarketPeriodType since=”xxx” [round=”xx”]>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryMarketPeriod>	Singleton Type	Specifies the query is for market information.
market	Character	Required field specifying the name of the initial allocation market.
round	Numeric(1)	Specifies the annual auction round signature (1,2,3,4). Not used for monthly auction markets.

5.9.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <MarketPeriod>
    <PeriodType>xxx</PeriodType>
    <MarketInterval> start=”ddd” end=”ddd” </MarketInterval>
  </MarketPeriod>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<MarketPeriod>	Complex Type	Element that contains the market Period. Occurs just once per query requested item.
market	Character	Specifies the market name.
round	Numeric(1)	Specifies the annual auction round signature (1,2,3,4). Not used for

Element or Attribute	Data Type	Description
		monthly auction markets.
<MarketPeriod>	Complex Type	Specifies the Market Period for the specified market and market round. Occurs 0 to many times.
<PeriodType>	Complex Type	Specifies the type of period i.e 'All', 'JUN', 'JUL'
<MarketInterval>	Singleton Type	Specifies the operational interval of the FTRs purchased or sold in the market as a start and end date. Interval is inclusive of dates. For example, annual markets may be June 1st, 2002 through May 31st, 2003. Or, a monthly market may be July 1st, through July 31st. Other periods, such as Winter or Summer would be defined with particular intervals.
start	dateTime with Zone	The market period's start date-time.
end	dateTime with Zone	The market period's end date –time.

5.10 Query for Cleared FTRs

5.10.1 Purpose

This message format is used to query market results for cleared FTRs of all participants.

5.10.2 Message Format

The Market Results Query request is shown below:

```
<QueryRequest>
  <QueryClearedFTRs market=″xxx″ round=″xxx″/>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message.

Element or Attribute	Data Type	Description
		Query request elements can be intermixed within the message. Occurs just once.
<QueryClearedFTRs>	Complex Type	Specifies the query for the cleared FTR results from a particular market. Occurs 0 to many times.
market	Character	Required field specifying the name of the market.
round	Numeric(1)	Optional field specifying the annual auction round signature (1,2,3,4). Not used for monthly auction markets.
<Period>	Character	Optional element specifying the period. Occurs 0 or 1 times.

5.10.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <ClearedFTRs market=”xxx” [round=”xx”] >
    <ClearedFTR trade="xxx">
      <ID>xxx</ID>
      <Owner>xxx</Owner>
      <Path source=”xxx” sink=”xxx” />
      <Class>xxx</Class>
      <Period>xxx</Period>
      <Hedge>xxx</Hedge>
      <ClearedMW>999.9</ClearedMW>
      <ClearedPrice>999.99</ClearedPrice>
    </ClearedFTR>
  </ClearedFTRs>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<ClearedFTRs>	Complex Type	Element that contains the cleared FTRs for the specified market. Occurs 0 to many times.

Element or Attribute	Data Type	Description
Market	Character	Name of the market.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly markets.
<ClearedFTR>	Complex Type	Used to specify a single FTR cleared in the market. Occurs 0 to many times.
trade	(Buy, Sell, SelfScheduled)	Specifies the trade type of the cleared FTR.
<ID>	Numeric(15)	Uniquly identifies the FTR
<Owner>	Character	Specifies the owner of the cleared FTR.
<Path>	Singleton Type	Specifies the FTR path. Occurs just once.
source	Character	Specifies the FTR path source node.
sink	Character	Specifies the FTR path sink node.
<Class>	(OnPeak,OffPeak, 24H)	Specifies the class of the FTR.
<Period>	Character	Specifies the period of the FTR.
<Hedge>	(Obligation, Option)	Specifies the hedge type.
<ClearedMW>	Numeric (8.1)	Specifies the cleared MW quantity of the FTR. Occurs just once.
<ClearedPrice>	Numeric(10.2)	Specifies the cleared price of the FTR. Occurs just once.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.

Error reported if the market has not cleared.

5.11 Query for Market Results (Private)

5.11.1 Purpose

This message format is used to query market results. Market results are queried by specified market and optionally by portfolio or path. If no portfolio or path is specified then all results for the participant are returned spanning all submitted FTR quotes.

5.11.2 Message Format

The Market Results Query request is shown below:

```
<QueryRequest>
  <QueryMarketResults market=”xxx” round=”xxx”>
     <All/>
     <Path source="xxx" sink="yyy"/>
     <PortfolioName>xxx</PortfolioName>
     <ID>xxx</ID>
  </QueryMarketResults>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryMarketResults>	Complex Type	Specifies the query for the results from a particular market. Occurs 0 to many times.
market	Character	Required field specifying the name of the market.
round	Numeric(1)	Optional field specifying the annual auction round signature (1,2,3,4). Not used for monthly auction markets.
<All/>	Null	Optional element to specify all FTRs submitted by the participant for the given market.

Element or Attribute	Data Type	Description
		Must choose one of: <All/>, <Path/>, or <PortfolioName>. Can specify choice at most one time.
<Path>	Singleton Type	Optional element to specify the path of FTRs. Must choose one of: <All/>, <Path/>, or <PortfolioName>. Can specify choice at most one time.
source	Character	Path source name.
sink	Character	Path sink name.
<PortfolioName>	Character	Optional element to specify portfolio defining paths of FTRs. Must choose one of: <All/>, <Path/>, or <PortfolioName>. Can specify choice at most one time.
<Period>	Character	Optional element specifying the period. Occurs 0 or 1 times.
<ID>	Numeric(15)	Uniquly identifies the FTR.

5.11.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <MarketResults market=”xxx” [round=”xx”] >
    <FTRCleared trade=”xxx”>
      <ID>xxx</ID>
      <Path source=”xxx” sink=”xxx” />
      <Class>xxx</Class>
      <Period>xxx</Period>
      <Hedge>xxx</Hedge>
      <BidMW>999.9</BidMW>
      <ClearedMW>999.9</ClearedMW>
      <BidPrice>999.99</BidPrice>
      <ClearedPrice>999.99</ClearedPrice>
    </FTRCleared>
  </MarketResults>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<MarketResults>	Complex Type	Element that contains the cleared FTRs for the specified market. Occurs 0 to many times.
market	Character	Name of the market.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly markets.
<FTRCleared>	Complex Type	Used to specify a single FTR cleared in the market. Occurs 0 to many times.
trade	(Buy, Sell,SelfScheduled)	Specifies whether the FTR cleared as an sell offer or as a purchase bid or self-scheduled.
<ID>	Numeric(15)	Uniquely identifies the FTR.
<Path>	Singleton Type	Specifies the FTR path. Occurs just once.
source	Character	Specifies the FTR path source node.
sink	Character	Specifies the FTR path sink node.
<Class>	(OnPeak,OffPeak, 24H)	Specifies the class of the FTR.
<Period>	Character	Specifies the period of the FTR.
<Hedge>	(Obligation, Option)	Specifies the hedge type.
<BidMW>	Numeric (8.1)	Specifies the bid in MW quantity of the FTR. Occurs just once.
<ClearedMW>	Numeric (8.1)	Specifies the cleared MW quantity of the FTR. Occurs just once.

Element or Attribute	Data Type	Description
<BidPrice>	Numeric(10.2)	Specifies the bid in price of the FTR. Occurs just once.
<ClearedPrice>	Numeric(10.2)	Specifies the cleared price of the FTR. Occurs just once.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.
- Error reported if the market has not cleared.

5.12 Query for Constraints (Public)

5.12.1 Purpose

This message format is used query a market for binding constraints. Binding constraints are published as a result of market clearing. Therefore, this report is available only for cleared markets.

To query for the binding constraints, you must specify the market and round (for annual markets).

5.12.2 Message Format

The binding Constraints Query request is shown below:

```
<QueryRequest>
  <QueryConstraints market=”xxx” [round=”xx”]/>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.

Element or Attribute	Data Type	Description
<QueryConstraints>	Complex Type	Specifies the query of a single report of the market interval's binding constraints. Occurs 0 to many times.
market	Character	Required field specifying the name of the market for the desired binding constraints.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly markets. Default value is 1.
<Period>	Character	Optional element specifying the period. Occurs 0 or 1 times.

5.12.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <Constraints market=”xxx” round=”xx”>
    <Constraint>
          <Period>xxx</Period>
       <Class>xxx</Class>
       <Monitored>xxx</Monitored>
       <Contingency>xxx</Contingency>
       <MarginalValue>99.99</MarginalValue>
    </Constraint>
  </Constraints>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<Constraints>	Complex Type	Element that contains the binding constraints that exist for the given market interval. Occurs 0 to many times. If the specified market interval does not include any binding constraints then this element will appear but there will

Element or Attribute	Data Type	Description
		be no <Constraint> elements.
market	Character	Specifies the name of the market.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly markets.
<Constraint>	Complex Type	Element that describes a single constraint of the class type designated. Occurs zero to many times. If the query resulted in no constraints for a given market interval then no <Constraint> elements will appear.
<Period>	Character	Specifies the period name.
<Class>	(OnPeak, OffPeak, 24H)	Element specifies the class of the constraint as OnPeak or OffPeak.
<Monitored>	Character	Element specifies the monitored facility name.
<Contingency>	Character	Element specifies the contingency causing the violation.
<MarginalValue>	Numeric(10.2)	The marginal value of the constraint in dollars per MW.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.
- Error reported if the specified interval is not defined.

5.13 Query for FTR Capability (Public)

5.13.1 Purpose

This message format is used query a market to obtain the total FTR capability on designated paths. Query can specify by path or by portfolio.

5.13.2 Message Format

The FTR Capability Query request is shown below:

```
<QueryRequest>
<QueryFTRCapability market=”xxx” round=”xx”>
  <All/>
  <Path source=”xxx” sink=”xxx” />
  <PortfolioName>xxx</PortfolioName>
</QueryFTRCapability>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryFTRCapability>	Complex Type	Specifies the query for a particular market.
market	Character	Required field specifying the name of the market.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly markets. Default value is 1.
<All/>	Null	Optional element that specifies that all paths with FTR Capability values be returned. Must choose one of: <All/>, <Path/>, or <PortfolioName>. Can specify choice at most one time.
<Path>	Singleton Type	Optional element that specifies a path to receive the FTR capability. Must choose one of: <All/>, <Path/>, or <PortfolioName>. Can specify choice at most one

Element or Attribute	Data Type	Description
		time.
source	Character	Required field specifies the path source node.
sink	Character	Required field specifies the path sink node.
<PortfolioName>	Character	Optional element specifying all paths defined for the Portfolio. Must choose one of: <All/>, <Path/>, or <PortfolioName>. Can specify choice at most one time.
<Period>	Character	Optional element specifying the period. Occurs 0 or 1 times.

5.13.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <FTRCapability market=”xxx” round=”xx”>
       <Capability>
      <Path source=”xxx” sink=”xxx”>
      <Period>xxx</Period>
      <OnPeakMW>9999.9</OnPeakMW>
      <OffPeakMW>9999.9</OffPeakMW>
    </Capability>
  </FTRCapability>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<FTRCapability>	Complex Type	Element that contains the individual FTRPath elements specifying the capability for a given market.
market	Character	Specifies the market.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly

Element or Attribute	Data Type	Description
		markets.
<Capability>	Complex Type	Specifies the FTR path to define the capability in MW. Occurs 0 to many times.
<Path>	Singleton Type	Specifies the FTR Path.
source	Character	Specifies the path source node name.
sink	Character	Specifies the path sink node name.
<Period>	Character	Specifies the period name.
<OnPeakMW>	Number(8.1)	Specifies the total on peak MW capability on the FTR path. Occurs just once.
<OffPeakMW>	Number(8.1)	Specifies the total off peak MW capability on the FTR path. Occurs just once.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.
- Error reported if a specified path does not exist or if a specified portfolio does not exist.

5.14 Query for FTR Nodes (Private)

5.14.1 Purpose

This message request is used to query system and obtain the pricing nodes available to the participant for FTRs.

5.14.2 Message Format

The FTR Nodes Query request is shown below:

```
<QueryRequest>
  <QueryFTRNodes market=”xxx” round=”xx”/>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryFTRNodes>	Complex Type	Specifies the query for a particular market.
market	Character	Required field specifying the name of the market.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly markets. Default value is 1.
<Period>	Character	Optional element specifying the period. Occurs 0 or 1 times.

5.14.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <FTRNodes market=”xxx” round=”xx”>
    <Node>xxx</Node>
  </FTRNodes>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<FTRNodes>	Complex Type	Element that contains the nodes available for FTRs for the specified market. Occurs 0 to many times.

Element or Attribute	Data Type	Description
market	Character	Specifies the market name.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly markets. Default value is 1.
period	Character	Specifies the period name.
<Node>	Character	Specifies the name of the pricing node. Occurs 0 to many times.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.
- Error reported if a specified path does not exist or if a specified portfolio does not exist.

5.15 Query for Option Paths (Public)

5.15.1 Purpose

This message format is used query the system for the available FTR option paths.

5.15.2 Message Format

The Option Paths Query request is shown below:

```
<QueryRequest>
  <QueryOptionPaths market=”xxx” round=”xx”/>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs

Element or Attribute	Data Type	Description
		just once.
<QueryOptionPaths>	Singleton Type	Specifies the query for a particular market.
market	Character	Required field specifying the name of the market.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly markets. Default value is 1.

5.15.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <OptionPaths market=”xxx” round=”xx”>
    <Path source=”xxx” sink=”xxx” />
  </OptionPaths>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<OptionPaths>	Complex Type	Element that contains the FTR option paths for the specified market. Occurs 0 to many times.
market	Character	Specifies the market name.
round	Numeric(1)	Optional field. Annual auction round (1,2,3,4). Not used for monthly markets. Default value is 1.
<Path>	Singleton Type	Specifies the FTR option path. Occurs 0 to many times.
source	Character	Specifies the path source node name. Occurs just once.

Element or Attribute	Data Type	Description
sink	Character	Specifies the path sink node name. Occurs just once.

The Unsuccessful (NAK) response is an error report as described in chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified market does not exist.

5.16 Query for Messages (Public)

5.16.1 Purpose

This message format is used query a market for current messages.

5.16.2 Message Format

The Messages Query request is shown below:

```
<QueryRequest>
  <QueryMessages>
    <EffectiveDate>YYYY-MM-DD</EffectiveDate>
  </QueryMessages>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryMessages>	Complex Type	Specifies the request to query messages. Both public and private messages are returned. There are no request parameters. Occurs just once.
<EffectiveDate>	YYYY-MM-DD	Optional element. Specifies an alternative effective current date to use

Element or Attribute	Data Type	Description
		to filter the messages. This allows the messages to be queried that were active on the specified date. The default is to use the current date. May be specified at most once.

5.16.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <Messages>
    <Message effectiveDate=”x” terminationDate=”x”>xx</Message>
  </Messages>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<Messages>	Complex Type	Element that contains all public and private messages. Occurs just once. If there are no messages then this container will not include any <Message> elements.
<Message>	Character	Specifies a single message. Occurs 0 to many times. The Character data is the message text.
effectiveDate	YYYY-MM-DD	Specifies the effective date of the message.
terminationDate	YYYY-MM-DD	Specifies the termination date of the message.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.

5.17 Query for Portfolios (Private)

5.17.1 Purpose

This message format is used query a for the participant's portfolios or any PJM public portfolios.

5.17.2 Message Format

The Messages Query request is shown below:

```
<QueryRequest>
  <QueryPortfolios>
     <All/>
     <PortfolioName>xxx</PortfolioName>
  </QueryPortfolios>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryPortfolios>	Complex Type	Specifies the portfolio query request. Occurs 0 to many times.
<All/>	Null	Optional field. Specifies that all of the participant's portfolios are returned. Must choose one of: <All/>, or <PortfolioName>. Can specify choice at most one time.
<PortfolioName>	Character	Optional field. Specifies the name of the portfolio. Must choose one of: <All/> or <PortfolioName>. Can specify choice at most one time.

5.17.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <Portfolios>
    <Portfolio name=”xxx”>
      <Path source=”xxx” sink=”xxx” />
    </Portfolio>
  </Portfolios>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<Portfolios>	Complex Type	Element that contains all portfolios returned as a result of the query. Occurs just once. If there are no portfolios that match the query then this element will not contain any <Portfolio> elements.
<Portfolio>	Complex Type	Specifies the named portfolio in response to the query. Occurs 0 to many times.
name	Character	Specifies the name of the portfolio. Occurs just once.
<Path>	Singleton Type	Specifies the path composed of a source and sink node. Occurs 0 to many times. Note that paths are directional.
source	Character	Required field specifies the path source node.
sink	Character	Required field specifies the path sink node.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.
- Error reported if the specified portfolio does not exist.

5.18 Query for Position (Private)

5.18.1 Purpose

This message format is used query for the participant's position on a specified date.

5.18.2 Message Format

The Position Query request is shown below:

```
<QueryRequest>
  <QueryPosition>
    <Date>YYYY-MM-DD</Date>
  </QueryPosition>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryPosition>	Complex Type	Specifies the request to query the FTR position. Occurs just once.
<Date>	YYYY-MM-DD	Required element. Specifies the date to use to calculate the position.
<Path>	Singleton Type	Specifies a path composed of a source and sink node. May be specified at most once.

5.18.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <Positions>
```

```
    <Position>
        <ID>xxx</ID>
        <MarketName>July Auction</MarketName>
      <MarketRound>1</MarketRound>
        <Path source=”xxx” sink=”yyy”/>
      <Trade>Sell</Trade>
      <Class>24H</Class>
      <Period>xxx</Period>
      <Hedge>Obligation</Hedge>
      <MW>999.9</MW>
      <Price>999.99</Price>
    </Position>
  </Positions>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<Positions>	Complex Type	Element that contains all positions for the specified date and path. Occurs just once. If there are no positions then this container will not include any <Position> elements.
<Position>	Complex Type	Specifies a single position. Occurs 0 to many times.
<ID>	Numeric(15)	Uniquly identifies the FTR
<MarketName>	Character	Name of the market.
<MarketRound>	Numeric(1)	Optional field. Annual auction round (1, 2, 3, 4). Not used for monthly markets.
<Path>	Singleton Type	Specifies the FTR path. Occurs just once.
source	Character	Specifies the FTR path source node.
sink	Character	Specifies the FTR path sink node.
<Trade>	(Buy, Sell, SelfScheduled)	Specifies the trade type of the FTR.

Element or Attribute	Data Type	Description
<Class>	(OnPeak,OffPeak, 24H)	Specifies the class of the FTR.
<Period>	Character	Specifies the period of the FTR.
<Hedge>	(Obligation, Option)	Specifies the hedge type.
<MW>	Numeric (8.1)	Specifies the MW quantity of the FTR. Occurs just once.
<Price>	Numeric(10.2)	Specifies the price of the FTR. Occurs just once.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.

5.19 Query for Credit Summary (Private)

5.19.1 Purpose

This message format is used query for the participant's credit summary on a specified date.

5.19.2 Message Format

The Position Query request is shown below:

```
<QueryRequest>
  <QueryCreditSummary>
    <Date>YYYY-MM-DD</Date>
  </QueryCreditSummary>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QuerySummary>	Complex Type	Specifies the request to query the credit summary. Occurs just once.
<Date>	YYYY-MM-DD	Required element. Specifies the date to use to calculate the credit summary.

5.19.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
  <QueryResponse>
     <CreditSummary date="2006-05-03" creditLimit="5000"
availableCredit="370.2" creditRequirement="4629.8" credit="0">
       <AuctionCreditDetail>
         <MarketName>BOPP JUN 2006-12PF</MarketName>
         <MarketRound>1</MarketRound>
         <RightType>FTR</RightType>
         <Credit>2777.88</Credit>
       </AuctionCreditDetail>
    </CreditSummary>
  </QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<CreditSummary>	Complex Type	Element that contains all credit summaries for each of the markets existing for the specified date. Occurs just once. If there are no markets then this container will not include any <AuctionCreditDetail> elements.
date	YYYY-MM-DD	Specifies the date of the credit summary.

Element or Attribute	Data Type	Description
creditLimit	Numeric(13.2)	The participant's total credit limit on the specified date.
availableCredit	Numeric(13.2)	The calculated amount of available credit on the specified date.
creditRequirement	Numeric(13.2)	The calculated amount of credit requirement on the specified date.
credit	Numeric(13.2)	The calculated amount of credit on the specified date.
<AuctionCreditDetail>	Complex Type	Specifies the credit details of the given market on the specified date. Occurs 0 to many times.
<MarketName>	Character	Name of the market.
<MarketRound>	Numeric(1)	Optional field. Annual auction round (1, 2, 3, 4). Not used for monthly markets.
<RightType>	(FTR,ARR)	Specifies the trade type of the FTR.
<Credit>	Numeric(13.2)	Optional field that specifies the amount of credit requirement/credit (depending on RightType). Occurs none or just once.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.

5.20 Query for Credit Details (Private)

5.20.1 Purpose

This message format is used query for the participant's position on a specified date.

5.20.2 Message Format

The Position Query request is shown below:

```
<QueryRequest>
  <QueryCreditDetails>
    <Date>YYYY-MM-DD</Date>
  </QueryCreditDetails>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs just once.
<QueryCreditDetails>	Complex Type	Specifies the request to query credit detail. Occurs just once.
<Date>	YYYY-MM-DD	Required element. Specifies the date used to calculate the credit detail.

5.20.3 Response Message

The Success (ACK) response message contains the data requested in the order requested.

```
<QueryResponse>
  <CreditDetails date="2006-05-03">
    <CreditDetail>
      <RightType>FTR</RightType>
      <MarketName>BOPP JUN 2006-12PF</MarketName>
      <MarketRound>1</MarketRound>
      <Path source="JCPL" sink="DPL" />
      <Period>AUG</Period>
      <Class>OffPeak</Class>
      <Hedge>Obligation</Hedge>
      <Trade>Buy</Trade>
      <BidMW>3</BidMW>
      <BidPrice>1</BidPrice>
      <ClearedMW>3</ClearedMW>
      <ClearedPrice>1</ClearedPrice>
      <BidCreditReq>2777.88</BidCreditReq>
      <ClearedCreditReq>2777.88</ClearedCreditReq>
      <CurrentCreditReq>2777.88</CurrentCreditReq>
    </CreditDetail>
  </CreditDetails>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element containing all response elements. Occurs just once.
<CreditDetails>	Complex Type	Element that contains all credit details for the specified date. Occurs just once. If there are no credit details then this container will not include any <CreditDetail> elements.
date	YYYY-MM-DD	The date when the credit detail where calculated.
<RightType>	(FTR,ARR)	Specifies the right type of the FTR
<CreditDetail>	Complex Type	Specifies a single detail. Occurs 0 to many times.
<MarketName>	Character	Name of the market.
<MarketRound>	Numeric(1)	Annual auction round (1, 2, 3, 4). Not used for monthly markets.
<Path>	Singleton Type	Specifies the FTR path. Occurs just once.
source	Character	Specifies the FTR path source node.
sink	Character	Specifies the FTR path sink node.
<Period>	Character	Specifies the period of the FTR.
<Class>	(OnPeak,OffPeak, 24H)	Specifies the class of the FTR.
<Trade>	(Buy, Sell, SelfScheduled)	Specifies the trade type of the FTR.
<Hedge>	(Obligation, Option)	Specifies the hedge type.
<BidMW>	Numeric (8.1)	Specifies the bid MW quantity of the FTR. Occurs just once.

Element or Attribute	Data Type	Description
<BidPrice>	Numeric (10.2)	Specifies the bid price of the FTR. Occurs just once.
<ClearedMW>	Numeric (8.1)	Specifies the cleared MW quantity of the FTR. Occurs just once.
<ClearedPrice>	Numeric (10.2)	Specifies the cleared price of the FTR. Occurs just once.
<BidCreditReq>	Numeric(13.2)	Optional field that specifies the bid credit requirement of the FTR. Occurs none or just once.
<ClearedCreditReq>	Numeric(13.2)	Optional field that specifies the cleared credit requirement of the FTR. Occurs none or just once.
<CurrentCreditReq>	Numeric(13.2)	Optional field that specifies the current credit requirement of the FTR. Occurs none or just once.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be encountered by this query include:

- Invalid or improper XML request.

6 FTR Private Data Submit

The Data Submit request is used to send data to the PJM FTR server and into the FTR market database for further processing. Data submitted is defined for the following purposes:

- Submitting, updating, or removing (deleting) FTR offers to sell and bids to buy in an annual or monthly auction or other defined market.

- Creating, Updating, Removing Portfolios.

Submits (and Queries) for the secondary trading market are covered in Chapter 7 of this document.

All data submit actions are executed under transaction control and a transaction identifier is associated with each submittal request. This transaction identifier has limited lifetime and is provided so that data submitted by an earlier message can be updated or deleted. The transaction identifier is invalid after the auction or market closes. Its only purpose is to update or remove data submitted to a market.

All data submitted is considered private – there is no notion of submitting public data. However, some of the private data, such as cleared quotes, is made available in public reports that can be queried or displayed on the web. At this point, the data is no longer considered private.

The data submit is initiated using a data submit request element as shown below. This submit request is the payload that is enclosed by the SOAP Body element. Only one such submit request element can be specified. The SubmitRequest may contain one or more instances of individual data elements containing the submitted data. By definition, only one kind of data submit is possible for each message.

Example of submit of FTR quotes to market:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Body>
   <SubmitRequest xmlns="http://eftr.pjm.com/ftr/xml">
      <FTRQuote>
          -- elements of FTR quote --
      </FTRQuote>
      <FTRQuote>
          -- elements of FTR quote --
      </FTRQuote>
      …
   </SubmitRequest>
</env:Body>
</env:Envelope>
```

The response message is always returned as a SOAP wrapped payload indicating either success or error. The success indicator is the same for all types of data submit. The successful submit always has the same format as shown below:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<env:Body>
   <SubmitResponse xmlns="http://eftr.pjm.com/ftr/xml">
      <Success>
         <TransactionID>Abee3433</TransactionID>
      </Success>
   </SubmitResponse>
</env:Body>
</env:Envelope>
```

The format for each of the submit requests follows the same naming conventions.

The following example shows an error response as a result of some problems in the submit:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Body>
   <SubmitResponse xmlns="http://eftr.pjm.com/ftr/xml">
      <Error>
         <Code>BlueGreen</Code>
         <Text>Market does not exist</Text>
      </Error>
   </SubmitResponse>
</env:Body>
</env:Envelope>
```

All private submit requests must be issued to the following URL:

```
https://ftr.pjm.com/ftr/xml/submit
```

If you choose to specify a SOAPAction HTTP header, then it must have the following value:

```
/ftr/xml/submit
```

Each of the following sections describes the defined submit requests.

6.1 Submitting FTR Quote to an Annual or Monthly Market

6.1.1 Purpose

This message format is used to submit FTR quotes (offers to sell and bids to buy) into annual and monthly (and other) markets. In order to submit an FTR quote the market must be open. Prior to market close and clearing, FTR quotes can be modified via replace or removed from the market.

The PJM FTR Business rules dictate the values that may be submitted. For example:

- Selfscheduled FTRs can be submitted into round 1 of the annual auction only. This is the only time that the trade type of SelfScheduled is valid.

- For monthly auctions, the Period element is not used and the default value is simply "All" meaning the entire month.

- When submitting quotes into a monthly auction, the round attribute on the FTRQuotes element is not used.

To modify quotes that have been submitted, you must delete the transaction that entered the quote originally and then resubmit the data with your modifications. If you fail to delete the previous set of quotes then you are adding to those quotes instead of replacing them.

6.1.2 Message Format

The full FTR Quote message format is described below.

```
<SubmitRequest>
<FTRQuotes market=”xxx” round=”xx”>
  <FTRQuote trade=”xxx”>
    <Path source=”xxx” sink=”xxx” />
    <Class>xxx</Class>
    <Period>xxx</Period>
    <Hedge>xxx</Hedge>
    <MW>999.9</MW>
    <Price>999.99</Price>
  </FTRQuote>
</FTRQuotes>
</SubmitRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitRequest>	ComplexType	The root element for all submit requests. Occurs just once and placed within the SOAP envelope Body element.
<FTRQuotes>	Complex Type	The element containing all FTR Quotes for this market. Occurs zero to many times.
market	Character	Required field. Specifies the name of the market to receive the quotes.
round	Numeric(1)	The auction round number 1, 2, 3, or 4 specified for annual markets only. Not used and not specified for monthly markets.
<FTRQuote>	Complex Type	Used to specify a single quote on a

Element or Attribute	Data Type	Description
		given path. Occurs 0 to many times.
trade	(Buy, Sell, SelfScheduled)	Required field specifies whether this quote is an bid to buy or an offer to sell. SelfScheduled is valid for round 1 of annual auction only.
<Path>	Singleton Type	Specifies the FTR Path.
source	Character	Required field specifies the FTR path source node.
sink	Character	Required field specifies the FTR path sink node and cannot be the same as source.
<Class>	(OnPeak,OffPeak, 24H)	Required field specifies the class of the FTR.
<Period>	(Character Name of the period.	
<Hedge>	(Obligation, Option)	Optional field specifies the hedge type, the default value is Obligation. Occurs none or just once.
<MW>	Numeric (8.1)	Required field specifies the MW quantity of the FTR. MW values are greater than zero. Occurs just once.
<Price>	Numeric(10.2)	Required field specifies the offer price if selling or the bid price if buying. Occurs just once. Price is allowed to be positive, negative, or zero for FTR Obligations and strictly positive only for FTR options.

6.1.3 Response Message

The Success (ACK) response message is described below.

```
<SubmitResponse>
  <Success>
     <TransactionID>xxx</TransactionID>
  </Success>
```

```
</SubmitResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitResponse>	Complex Type	The root element of all submit response elements. Occurs just once.
<Success>	Complex Type	The element indicating a successful submit operation. Occurs just once.
<TransactionID>	Character	Specifies the transaction identifier associated with the successful submit operation. Always returned.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be reported include:

- Invalid or improper XML
- Market does not exist.
- Market is not open.
- Violation of FTR Market Rules.

6.2 Creating or Updating Portfolio

6.2.1 Purpose

This message format is used to create or update a participant's portfolio. Each portfolio is a named association of paths designated by a source and sink node.

The action field in the message below specifies the update action to perform. The following values are supported:

Create	Means to create a new portfolio.
Replace	Means to replace an existing portfolio.
Remove	Means to remove an existing portfolio (the Path elements are

ignored for this action).

AddPath	Means to add new paths to an existing portfolio.
RemovePath	Means to remove paths from an existing portfolio.

Note that paths have direction. If you want to include both directions between two nodes, two paths must be specified where the nodes are reversed as source and sink.

6.2.2 Message Format

The full PortfolioSubmit message is described below:

```
<SubmitRequest>
   <Portfolio name=”xxx” action=”xxx”>
      <Path source=”xxx” sink=”xxx”/>
   </Portfolio>
</SubmitRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitRequest>	Complex Type	The root element containing all Portfolio manipulation commands. Occurs just once.
<Portfolio>	Complex Type	Used to create, replace, update, or remove a portfolio. Occurs 0 to many times.
name	Character	Required field specifies the name of the portfolio. Occurs just once. This same name field is required for each of the other elements: <AddToPortfolio>, <RemoveFromPortfolio>, and <RemovePortfolio>.
action	(Create, Replace, Remove, AddPath, RemovePath)	Optional field specifies the action to be performed as one of the enumerated values. The default action is Create.
<Path>	Singleton Type	Specifies the path to composed of a source and sink node. Occurs 0 to

Element or Attribute	Data Type	Description
		many times. This same <Path> element is used with the <AddToPortfolio> and <RemoveFromPortfolio> described below.
source	Character	Required field specifies the path source node.
sink	Character	Required field specifies the path sink node and cannot be the same as source.

6.2.3 Response Message

The Success (ACK) response message is described below.

```
<SubmitResponse>
  <Success>
    <TransactionID>xxx</TransactionID>
  </Success>
</SubmitResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitResponse>	Complex Type	The root element for all submit responses. Occurs just once.
<Success>	Complex Type	The element indicating a successful submit operation.
<TransactionID>	Character	Specifies the transaction identifier associated with the successful submit operation. Always returned.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. The following errors may be reported:

- Invalid or improper XML
- Attempt to duplicate an existing portfolio without the *replace* override.

- Portfolio does not exist.

6.3 Submitting ARR bids to an Annual Market

6.3.1 Purpose

This message format is used to submit ARR bids into annual markets. In order to submit an ARR bids the market must be open. Prior to market close and clearing, ARR bids can be modified via replace or removed from the market.

To modify bids that have been already submitted, you can just resubmit with your modifications in bid mw.

To delete bids that have been already submitted, you need to resubmit with zero or null bid mw. Null means no value for bid mw. Do not put the string 'null' or 'NULL' for bid mw otherwise it will cause an error.

6.3.2 Message Format

The full FTR Quote message format is described below.

```
<SubmitRequest>
<ARRQuotes MarketName=″xxx″ MarketRoundName=″xxx″>
  <ARRQuote>
    <Path source=″xxx″ sink=″xxx″ />
    <MW>999.9</MW>
  </ARRQuote>
</ARRQuotes>
</SubmitRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitRequest>	ComplexType	The root element for all submit requests. Occurs just once and placed within the SOAP envelope Body element.
<ARRQuotes>	Complex Type	The element containing all FTR Quotes for this market. Occurs zero to many times.
MarketName	Character	Required field. It specifies the name of the market to receive the quotes.
MarketRoundName	Character	Required field. It specifies the auction

Element or Attribute	Data Type	Description
		round name. Valid values are: “Stage 1A” or “Stage 1B” or “Stage 2 Round 1” or “Stage 2 Round 2” or “Stage 2 Round 3”.
<ARRQuote>	Complex Type	Used to specify a single quote on a given path. Occurs 0 to many times.
<Path>	Singleton Type	Specifies the FTR Path.
source	Character	Required field. It specifies the FTR path source node.
sink	Character	Required field. It specifies the FTR path sink node and cannot be the same as source.
<MW>	Numeric (8.1)	Required field. It specifies the MW quantity of the FTR. MW values are greater than zero. Occurs just once.

6.3.3 Response Message

The Success (ACK) response message is described below.

```
<SubmitResponse>
  <Success>
     <TransactionID>xxx</TransactionID>
  </Success>
</SubmitResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitResponse>	Complex Type	The root element of all submits

Element or Attribute	Data Type	Description
		response elements. Occurs just once.
<Success>	Complex Type	The element indicates a successful submit operation. Occurs just once.
<TransactionID>	Character	Specifies the transaction identifier associated with the successful submit operation. Always returned.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be reported include:

- Invalid or improper XML
- Market does not exist.
- Market is not open.
- Market round name is invalid.
- Bid MW value is negative
- Invalid source
- Invalid sink
- Violation of ARR Market Rules.

7 Secondary Trading

The secondary trading service allows FTRs to be bilaterally traded with other participants. FTR trading is open all year after the annual auction.

The following functions are supported by the secondary trading service:

- Post FTRs to the secondary trading bulletin board service. FTRs can be posted as offers to sell or bids to purchase (buy). Once posted, the FTRs are publicly available for review by other other participants. Posting to the bulletin board is accomplished by submitting the TradingPost message. The participant can also query for those trades he posted to the bulletin board. This is accomplished using the QueryTrades specifying the list of "Posted".

- Query the secondary trading bulletin board for posted available FTR trades. Querying the bulletin board is accomplished by the QueryTrades specifying the list of "Available".

- The submitter of a posted FTR may delete the post which removes it from the bulletin board. This can only be done if it has not been accepted by some other participant. To delete a posted FTR, you submit the TradingAction message with the action of DeletePosted.

- A participant may accept a trade that has been posted on the bulletin board. This is accomplished using the submit TradingAction message with the action of AcceptTrade. To view those trades that have been accepted but not yet confirmed, the participant executes the QueryRequest message QueryTrades specifying the "Accepted" list. Only one participant may accept a trade.

- A participant may cancel any of the trades that have been accepted but not yet confirmed. This is accomplished using the submit TradingAction with the action of CancelAccepted. Once the trade is cancelled, it reverts back to being available on the bulletin board.

- The poster of an FTR may also query to determine which posted deals have been accepted by other participants. This is also accomplished by issueing the Query Request for QueryTrades using the "Accepted" list. The PostedBy and AcceptedBy fields determine if the participant is the poster of an accepted trade or the acceptor of a trade.

- The poster may reject an accepted trade merely by submitting the message TradingAction with the action of RejectTrade. When a trade is rejected, it is deleted from the bulletin board and no longer available. To be available, the poster must resubmit the trade.

- The poster may confirm a trade by submitting the message TradingAction with the action of ConfirmTrade. Those trades that have been confirmed are available for query by either party to the trade using the QueryTrades specifying the list "Confirmed". When a party to a trade queries the confirmed trades, all information is returned. However, any

participant may query the confirmed trades. If a participant is not a party to the trade itself, the price information of the trade is not returned.

- An activity log is maintained for all secondary trading activity of a given participant. This activity log is queried using the QueryTradingLog message.

Secondary trading introduces several data elements that are part of the data exchange messages. These data elements are described here:

<ID>	This element is used to specify the trading deal unique identifier. These identifiers are obtained by querying the trades posted to the bulletin board (using QueryTradingAvailable). They are used on subsequent TradingAction messages.
<PostedBy>	This element identifies the participant who posted the trade to the secondary trading bulletin board.
<AcceptedBy>	This element identifies the participant who accepted a trade posted by another participant.
action	This attribute used with the TradingAction message describes the actions to be performed on the trades identified using the <ID> element. The actions defined are: DeletePosted, RejectTrade, CancelAcceptance, ConfirmTrade, AcceptTrade.
list	This attribute is used with the QueryTrades message and it describes the list being queried. The lists are: Available, Posted, Confirmed, Accepted.

All data submit actions are executed under transaction control and a transaction identifier is associated with each submittal request. However, unlike the submittal of FTR Quotes, the secondary trading submits cannot be revoked using the transanction identifier. The DeleteByTransaction message does not apply. The reason for this is that the submittals to the secondary trading service are executed as soon as they are validated.

The TradingAction and TradingPost submits are executed using the SubmitRequest as described earlier in chapter 6. The query requests are executed using a QueryRequest as described in chapter 5.

Error responses are returned in a manner as described in chapters 5 and 6 of this document.

All submit requests to the secondary trading service must be issued to the following URL:

```
https://ftr.pjm.com/ftr/xml/submit
```

All query requests to the secondary trading service are issued to the following URL:

```
https://ftr.pjm.com/ftr/xml/query
```

If you choose to specify a SOAPAction HTTP header, then it must have the following value:

```
/ftr/xml/submit
```

Each of the following sections describes the submit and query requests defined for secondary trading.

7.1 Submitting TradingPost Message

7.1.1 Purpose

This message format is used to submit FTRs to the secondary trading bulletin board service. Once submitted and validated, the trades are publicly available to all participants.

All FTRs offered for sale on the secondary market must specify the market (and round if the auction is an annual market) that cleared the FTR. The markets specified can include annual markets, monthly markets, and special study markets used by PJM for transmission reservations.

7.1.2 Message Format

The TradingPost message format is described below.

```
<SubmitRequest>
<TradingPost>
  <FTR trade=”xxx” market="xxx" round="x" >
    <Interval start="xxx" end="xxx"/>
    <Path source=”xxx” sink=”xxx” />
    <Class>xxx</Class>
    <Period>xxx</Period>
    <Hedge>xxx</Hedge>
    <MW>999.9</MW>
    <Price>999.99</Price>
  </FTR>
</TradingPost>
</SubmitRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitRequest>	ComplexType	The root element for all submit requests. Occurs just once and placed within the SOAP envelope Body element.
<TradingPost>	Complex Type	The element containing all posts to the secondary trading bulletin board. Occurs zero to many times.

Element or Attribute	Data Type	Description
<FTR>	Complex Type	Used to specify a single FTR to be posted to the bulletin board. Occurs 0 to many times.
trade	(Buy, Sell)	Required field specifies whether this trade is an bid to buy or an offer to sell.
market	Character	Required field specifying the market name of the market that cleared the FTR being posted for sale. Requests to buy do not require a market name. Thus, it is not specified for a trade type of Buy.
round	Numeric(1)	Required field if the market specified above is an annual market. Specified as an integer value 1, 2, 3, or 4. Not required for trade type of Buy.
<Interval>	Singleton Type	Specifies the interval of the FTR trade inclusively from the start to the end date.
start	Date	Required field specifying the start date.
end	Date	Required field specifying the end date.
<Path>	Singleton Type	Specifies the FTR Path.
source	Character	Required field specifies the FTR path source node.
sink	Character	Required field specifies the FTR path sink node and cannot be the same as source.
<Class>	(OnPeak,OffPeak, 24H)	Required field specifies the class of the FTR.
<Period>	Character	Required field specifies the name of the period.
<Hedge>	(Obligation, Option)	Optional field specifies the hedge type, the default value is Obligation. Occurs

Element or Attribute	Data Type	Description
		none or just once.
<MW>	Numeric (8.1)	Required field specifies the MW quantity of the FTR. MW values are greater than zero. Occurs just once.
<Price>	Numeric(10.2)	Required field specifies the offer price if selling or the bid price if buying. Occurs just once. Price is allowed to be positive, negative, or zero for FTR

7.1.3 Response Message

The Success (ACK) response message is described below.

```
<SubmitResponse>
  <Success>
     <TransactionID>xxx</TransactionID>
  </Success>
</SubmitResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitResponse>	Complex Type	The root element of all submit response elements. Occurs just once.
<Success>	Complex Type	The element indicating a successful submit operation. Occurs just once.
<TransactionID>	Character	Specifies the transaction identifier associated with the successful submit operation. Always returned.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be reported include:

- Invalid or improper XML
- Violation of Secondary Trading Rules.

7.2 Submit TradingAction

7.2.1 Purpose

This message format is used to submit trading actions on the secondary market. The following trading actions are defined. These are specified by the action attribute on the TradingAction element.

AcceptTrade	Executed to accept the trade from the available trades posted. Completion of this action moves the trade item from the available trades bulletin board to both the PostedBy and the AcceptedBy parties Accepted List.
CancelAccepted	Executed by the accepting party to delete the accepted trade prior to confirmation by the posting party. This command action is executed by the party who accepted the deal but now is no longer interested. Completion of this action results in the Accepted Trade being removed from the parties accepted list and placed back in the posted available list.
ConfirmTrade	Executed by the posting party to confirm an accepted trade. Completion of this action results in the trade being moved from the accepted list to the confirmed trades list.
DeletePosted	Executed to delete a trade posted to the available trades bulletin board. This action can be executed only by the participant who posted the trade.
RejectTrade	Executed by the posting party to reject a trade accepted by the counter party (the AcceptedBy party). Completion of this command results in the trade being deleted from the secondary trading service. The secondary trading log of both parties (posted by and accepted by) is updated to reflect the rejection.

7.2.2 Message Format

The TradingAction message is described below:

```
<SubmitRequest>
   <TradingAction action="xxx">
      <ID>xxx</ID>
   </TradingAction>
```

```
</SubmitRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitRequest>	Complex Type	The root element containing all Portfolio manipulation commands. Occurs just once.
<TradingAction>	Complex Type	Used to submit a Trading action command on the secondary trading service.
action	(AcceptTrade, CancelAccepted, ConfirmTrade, DeletePosted, RejectTrade)	Required field action to be performed:
<ID>	Character	Specifies the trade unique identifier associated with the trade element. This trade identifier is obtained first by querying the available trades from the bulletin board service. Can be specified 1 to many times. Each trade identified is affected by the action specified.

7.2.3 Response Message

The Success (ACK) response message is described below.

```
<SubmitResponse>
  <Success>
    <TransactionID>xxx</TransactionID>
  </Success>
</SubmitResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<SubmitResponse>	Complex Type	The root element for all submit responses. Occurs just once.
<Success>	Complex Type	The element indicating a successful

Element or Attribute	Data Type	Description
		submit operation.
<TransactionID>	Character	Specifies the transaction identifier associated with the successful submit operation. Always returned.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. The following errors may be reported:

- Invalid or improper XML
- Violation of Secondary Trading Business rules.

7.3 QueryTrades: Request for Trades Posted, Available, Accepted, and Confirmed

7.3.1 Purpose

These query request messages are used to query the various secondary trading lists that are provided for FTRs. These lists are:

Posted are those FTR trades submitted by the participant.

Available are those FTR trades available on the bulletin board.

Accepted are those trades that have been accepted by a participant.

Confirmed are those trades that have been confirmed and are completed deals.

The query can specify filters using <ID>, <PostedBy>, <AcceptedBy>, and <Interval>. These filters are accumulated into a single query action. If no filters are applied then all contents of the list are returned. If you are not a participant to a confirmed trade then you will not receive the <Price> element on the result.

7.3.2 Message Format

The QueryTrades message format is described below.

```
<QueryRequest>
<QueryTrades list="xxx">
  <ID>xxx</ID>
  <PostedBy>xxx</PostedBy>
  <AcceptedBy>xxx</AcceptedBy>
  <Interval start="xxx" end="xxx"/>
```

```
</QueryTrades>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	ComplexType	The root element for all query requests. Occurs just once and placed within the SOAP envelope Body element.
<QueryTrades>	Complex Type	The element that specifies a given trade list and a right type (ARR or FTR) to query. Occurs zero to many times.
list	(Accepted, Available, Confirmed, Posted)	Specifies the list to query.
righttype	Character	Specifies the right type i.e. 'FTR', 'ARR'
<ID>	Character	Optional element to specify an individual FTR by unique identifier. This unique identifier must mach up with the corresponding FTR and list specified. The query can be further restricted by other filters below. Can be specified at most once.
<PostedBy>	Character	Optional element to specify a filter to apply to the query. Only those trades posted by the specified participant identifier are queried. Occurs zero or one time. Can be specified at most once.
<AcceptedBy>	Character	Optional element to specify a filter to apply to the query. Only those trades accepted by the specified participant identifier are queried. Occurs zero or one time. Can be specified at most once.
<Interval>	Singleton Type	Specifies the interval of the available trades to consider in the query. Any trade active within the inclusive start and end dates is is included. A trade can span outside this interval range but some part of the trades interval must

Element or Attribute	Data Type	Description
		qualify per this filter. Occurs zero or one time. Can be specified at most once.
start	Date	Required field specifying the start date.
end	Date	Required field specifying the end date.

7.3.3 Response Message

The Query response message is shown below.

```
<QueryResponse>
  <Trades list="xxx">
     <FTR market="xxx" round="xx" trade="xx" >
       <ID>
       <PostedBy>xxx</PostedBy>
       <AcceptedBy>xxx</AcceptedBy>
       <Confirmation>xxx</Confirmation>
       <Interval start="xxx" end="xxx"/>
       <Path source="xxx" sink="xxx"/>
       <Class>xxx</Class>
       <Hedge>xxx</Hedge>
       <Period>xxx</Period>
       <MW>xxx</MW>
       <Price>xxx</Price>
       <ClearingPrice>xxx</ClearingPrice>
     </FTR>
  </Trades>
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element of all query response elements. Occurs just once.
<Trades>	Complex Type	The element specifies the trades resulting from the query. Occurs zero to many times.
list	(Available, Accepted, Confirmed, Posted)	Specifies the secondary trading list.
<FTR>	Complex Type	Specifies an FTR trade item. Occurs

Element or Attribute	Data Type	Description
		zero to many times.
<ARR>	Complex Type	Specifies an ARR trade item. Occurs zero to many times.
trade	(Buy, Sell)	Specifies whether the FTR was posted to the secondary market as a offer to Sell or a bid to purchase (Buy).
market	Character	The name of the market that cleared the FTR. Only specified for posted offers to Sell an FTR.
round	Numeric(1)	Market round specified (1, 2, 3, or 4) only if the market specified above is an annual market.
<ID>	Character	Specifies the trade items unique identifier. This identifier is used on subsequent TradeAction requests.
<PostedBy>	Character	Specifies the identifier of the participant that posted the FTR to the bulletin board. Occurs once.
<AcceptedBy>	Character	Specifies the identifier of the participant that accepted the FTR trade. This value may be null (specified as an empty tag <AcceptedBy/>. Occurs once.
<Confirmation>	DateTime	Specifies the date and time of the confirmation of a trade. If null, the trade has not yet been confirmed. Occurs once.
<Interval>	Singleton Type	Specifies the interval range from start to end dates of the FTR trade. Start and end dates are inclusive. Occurs once.
start	Date	The interval start date.
end	Date	The interval end date.

Element or Attribute	Data Type	Description
Path	Singleton Type	Specifies the path of the FTR. Occurs once.
source	Character	The path source node.
sink	Character	The path sink node.
Class	(OnPeak, OffPeak, 24H)	The class of the FTR. Occurs zero or one time.
Hedge	(Obligation, Option)	The hedge for the FTR. Occurs zero or one time.
Period	Character	The period of the FTR. Occurs zero or one time.
MW	MWType	The FTR capacity.
Price	Numeric(10.2)	The agreed upon sale price of the FTR in $ per MW.
ClearingPrice	Numeric(10.2)	The clearing price of the FTR posted as an offer to sell. This field is optional and only appears for those FTRs posted for sale.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be reported include:

- Invalid or improper XML

7.4 QueryTradingLog

7.4.1 Purpose

This query request message is used to query the participant's secondary trading activity log. You can query for today's activities (by default) or specify any past date to bound the query.

7.4.2 Message Format

The QueryTrades message format is described below.

```
<QueryRequest>
<QueryTradingLog since="date"/>
</QueryRequest>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryRequest>	ComplexType	The root element for all query requests. Occurs just once and placed within the SOAP envelope Body element.
<QueryTradingLog>	Singleton Type	The element that specifies to query for the trading log. Occurs zero to many times.
since	Date	Optional value. Specifies the date for the start of the query activity events. By default, the query is for today.
<PostedBy>	Character	Specifies the identifier of the participant that posted the. Occurs once.
<AcceptedBy>	Character	Specifies the identifier of the participant that accepted the FTR trade. This value may be null (specified as an empty tag <AcceptedBy/>. Occurs once.

7.4.3 Response Message

The Query response message is shown below.

```
<QueryResponse>
  <TradingLog>
    <TradeEvent event="xxx" timestamp="YYYY-MM-DDTHH:MM:SS.000-HH:00">
     <FTR market="xxx" round="x" trade="xxx" >
       <ID>
       <PostedBy>xxx</PostedBy>
       <AcceptedBy>xxx</AcceptedBy>
       <Confirmation>xxx</Confirmation>
       <Interval start="xxx" end="xxx"/>
       <Path source="xxx" sink="xxx"/>
       <Class>xxx</Class>
       <Hedge>xxx</Hedge>
       <Period>xxx</Period>
       <MW>xxx</MW>
       <Price>xxx</Price>
       <ClearingPrice>xxx</ClearingPrice>
     </FTR>
    </TradeEvent>
  </TradingLog>
```

```
</QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<QueryResponse>	Complex Type	The root element of all query response elements. Occurs just once.
<TradingLog>	Complex Type	The element specifies the Trading Log structure and other TradeEvent elements.
<TradeEvent>	Complex Type	Specifies one or more trade events. Occurs zero to many times.
event	(AcceptTrade, CancelAccepted, ConfirmTrade, DeletePosted, PostTrade, RejectTrade)	Specifies trade event type.
timestamp	dateTime with Zone	Specifies the time stamp of the event.
<FTR>	Complex Type	Specifies the FTR involved in the event. Each event records the action on a single FTR. If a single message involves more than on FTR then they are recorded as separate events. The description of the FTR fields can be found in the previous description of the response for QueryTrades.

The Unsuccessful (NAK) response is an error report as described in Chapter 4 {Error Response} of this specification. Errors that may be reported include:

- Invalid or improper XML

8 Querying, Deleting, And Replacing By Transaction Identifier

8.1 Description of Transactions: Query, Delete, and Replace

Every successful submit is identified by a unique transaction ID that is returned in response to the SubmitRequest as part of the SubmitResponse <Success> element. The transaction governs all data submitted as part of the SubmitRequest.

In general, with XML, the most affective way to replace a given submittal or to update part of the data is to first remove the entire submittal using the DeleteByTransaction message and then resubmitting the data in full after modifying the message content. Of course, this replacement operation must take place prior to market (aka auction) close. This method is generally easier to accomplish than creating a special replace or update message that only does a partial change to existing data. If a partial change though is more convenient for some reason, the interactive web browser interface offers that capability.

Query, Delete, or Replace by Transaction is effective for changes made to the FTR Auction markets only. Query, Delete, or Replace cannot be used with the Secondary Trading Service.

8.1.1 Querying By Transaction Identifier

To query by transaction identifier you must have the TransactionID that was returned by some previous data submittal. You may query for any previous submittal as long as the data is available. Data is historically retained for a given amount of time (consult the RTO for the actual length of time supported) and query by transaction only operates on data still available in the market database.

Data can be queried at any time. If you are desiring to query data prior to market close with the intent of resubmitting the data, the query operation and resubmittal must of course take place prior to market close.

To query by transaction, you submit the following QueryRequest message:

```
<QueryRequest>
<QueryByTransaction>
  <TransactionID>xxx</TransactionID>
</QueryByTransaction>
</QueryRequest>
```

The message fields and elements are described in the table below:

Element or Attribute	Data Type	Description
<QueryRequest>	Complex Type	The root element containing all query request elements in this message. Query request elements can be intermixed within the message. Occurs

Element or Attribute	Data Type	Description
		just once and specified in the SOAP Body element.
<QueryByTransaction>	Complex Type	Specifies that the query request is a query by transaction. May be repeated as desired.
<TransactionID>	Character String	Specifies the transaction identifier for the query by transaction. May be repeated as desired.

The response to a successful query by transaction is a QueryResponse element that includes the individual message types that can be returned. The following example shows the query by transaction response of the FTRQuotes submitted to a particular auction market.

```
<QueryResponse>
   <FTRQuotes market="August2002">
      -- data of FTRQuotes --
   </FTRQuotes>
       …

</QueryResponse>
```

If the response include errors, such as invalid transaction identifier, then the <Error> element will be returned as part of the <QueryResponse>.

8.1.2 Deleting By Transaction Identifier

To delete a previously submitted message by transaction identifier the market must not be closed for the message type. If the market is closed, then the data cannot be deleted. Otherwise, previously submitted data can be deleted at any time.

To delete a submittal by transaction identifier, you submit the following SubmitRequest:

```
<SubmitRequest>
   <DeleteByTransaction>
      <TransactionID>xxx</TransactionID>
   </DeleteByTransaction>
</SubmitRequest>
```

The message fields and elements are described in the table below:

Element or Attribute	Data Type	Description
<SubmitRequest>	Complex Type	The root element containing the submit request elements in this message. Occurs just once and specified in the

Element or Attribute	Data Type	Description
		SOAP Body element.
<DeleteByTransaction>	Complex Type	Specifies that the request is a delete by transaction. May be repeated as desired.
<TransactionID>	Character String	Specifies the transaction identifier for the delete by transaction. May be repeated as desired.

The successful response to a delete by transaction is a regular SubmitResponse success element as shown below:

```
<SubmitResponse>
   <Success>
      <TransactionID>xxx</TransactionID>
   </Success>
</SubmitResponse>
```

The message fields are interpreted just like any other successful SubmitResponse documented elsewhere in this specification. Note however that a transaction identifier returned as a result of a delete by transaction is special. It cannot be deleted by a subsequent DeleteByTransaction message. You cannot undo any delete by transaction requests.

The error response is returned if there are any errors uncovered during the delete by transaction message submitted. Errors are returned using the <Error> element returned as part of the <SubmitResponse> as described for other data submittals. The errors that can be returned include:

- Invalid or malformed XML.
- Market is closed, delete by transaction rejected.
- Transaction cannot be deleted (for example, it is a delete transaction).

8.1.3 Replacing Data

To replace data submitted by earlier Submit Request, it is usually easiest to resubmit the entire data set with the necessary modifications and edits already made. This can be done as long as the market is still open. To perform such a replace, you do the following:

1. Query by transaction identifier to obtain the data set to be updated or replaced (this is only necessary if your system does not already have the necessary data).
2. Make appropriate changes to data and formulate as a SubmitRequest.

3. Delete the previous data set by transaction identifier (same identifier as used for the query in step 1 above).

4. Submit the modified data (from step 2).

It is important to make sure that steps 3 and 4 are separate submit messages. The reason is that any submittal that includes a delete operation cannot itself by deleted by transaction identifier. Therefore, by keeping them separate transactions, you avoid the problem of linking a delete operation into the same transaction as a data submittal.

9 Error Response

The error response message format is described below:

```
<SubmitResponse>  or  <QueryResponse>
   <Error>
      <Code>xxx</Code>
      <Text>xxx</Text>
      <Line>xxx</Line>
   </Error>
</SubmitReponse>  or  </QueryResponse>
```

The following table describes each of the elements and attributes and how they are used:

Element or Attribute	Data Type	Description
<Error>	Complex Type	Specifies a single error report. Occurs 1 to many times (hopefully not too many times).
<Code>	Character	Optional error code. Error codes may be associated with various software modules. Error codes have meaning only when reporting problems to PJM or requesting help on a given error type. Occurs at most once. Only occurs if vendor specific error code exists (e.g. Oracle, Weblogic, etc.).
<Text>	Character	Specifies the text of the error message. Occurs 1 to many times. Normally, occurs just once, sometimes two or three <Text> elements are used per error.
<Line>	Character	Optional line number indicator provided only when it is available and when it has meaning. Occurs 0 or 1 times.