TELVENT

Detailed Design Document

Urban Partnership Program (UPP)
Parking Vendor Interface
/Master Database Interface
1.0 (Final)

December 7, 2010

Revision History			
Date	**Author**	**Version**	**Notes**
3/19/2010	Reuben Bergman		Initial Draft
3/29/2010	Karl Barnes		Added formatting changes
3/31/2010	Reuben Bergman/ Karl Barnes		Updated database table definitions
4/12/2010	Reuben Bergman		Added vendorID, stored procedures
4/23/2010	Reuben Bergman		Added new tables and stored procs for AssociatedFiles, Entrances, ParkingAccess, SpacesByType
7/14/2010	Reuben Bergman	1.4	Changes to pricing and additional static fields
9/21/2010	Reuben Bergman/ Karl Barnes	1.5	Addressing MTC comments
12/7/2010	MTC	1.0 (Final)	Version 1.0 Final Relaese

TABLE OF CONTENTS

1 INTRODUCTION 5
1.1 Purpose 5
1.2 Objectives 5
1.3 Scope of Design 5
1.4 Acronyms 5
1.5 References 6
1.6 Design Process 7
1.7 Design Tools 7
1.8 Development Tools 7
1.9 Work Products 8

2 SOFTWARE ARCHITECTURE 9
2.1 Real-Time Parking System 9
2.2 Communications 11
2.3 Logging 11
2.4 Error/Exception Processing 12
2.5 Database Objects 13

3 MODELS 24
3.1 Use Cases 24
3.2 Class Diagrams 26
3.3 Sequence Diagrams 31

4 DEPLOYMENT 33
4.1 Deployment (Deployment Diagram) 33

LIST OF FIGURES

FIGURE 1. RT PARKING LOGICAL DIAGRAM 10
FIGURE 2. JMS CONNECTION DIAGRAM 10
FIGURE 3. PARKINGUSECASES (USE CASE DIAGRAM) 25
FIGURE 4. UPPPARKINGCLASSES (CLASS DIAGRAM) 27
FIGURE 5. MASTERDATABASEINTERFACECLASSES (CLASS DIAGRAM) 29
FIGURE 6. UPPPARKINGSEQUENCE (SEQUENCE DIAGRAM) 31
FIGURE 7. MDBISEQUENCE (SEQUENCE DIAGRAM) 32
FIGURE 8. DEPLOYMENT (DEPLOYMENT DIAGRAM) 33

1 Introduction

1.1 Purpose

The following report describes the detailed design of the Urban Partnership Program (UPP) Real-Time Parking component of the 511 real-time system. In particular, the master database for the UPP Parking Information System including elements of the data transfer design between the various 511 Systems and the parking vendor systems. The purpose of the system is to allow 511 users to access parking information on 511 phone and web, and to set preferences for parking information through the personalized MY 511 service.

1.2 Objectives

The main objective of this document is to provide software developers with details of the implementation of the UPP Parking system. The Metropolitan Transportation Commission (MTC) is currently implementing several new enhancements associated with the 511 Real-Time Transit System. Among those features is the ability for users to request parking space availability, garage location, pricing information, etc. for municipal parking lots in downtown San Francisco.

1.3 Scope of Design

This document describes the design for System Requirements Interface I-001 through I-010. (Allocation PI511). The design described is limited to the services interfaces. Information concerning the IVR design is not included in this document. In addition, requirements I-011 through I-029 pertain to the parking vendors systems.

1.4 Acronyms

The acronyms that appear throughout this document are defined in the table below.

Acronym Definitions

UML	Unified Modeling Language
ATIS	Advanced Traveler Information System
ITIS	Integrated Transport Information System
JDOM	Java Document Object Model

HW	Hardware
URL	Uniform Resource Locator
LRMS	Location Referencing Messaging Specification
IVR	Interactive Voice Response
SQL	Structured Query Language
LAN	Local Area Network
MTC	Metropolitan Transportation Commission
NDA	Non-Disclosure Agreement
RT	Real-time
SLA	Service Level Agreement
MDBI	Master Database Interface
MMTP	Multi Modal Trip Planner
PVI	UPP Parking Vendor Interface
SW	Software
UPP	Urban Partnership Program
XML	Extensible Markup Language
SMTP	Simple Mail Transfer Protocol
IP	Internet Protocol
XSD	XMLSchema Definitions

1.5 References

- *Urban Partnership Program Coordinated Concept of Operations, Kimley-Horn and Associates, June 4, 2009*
- *System Requirements for UPP Parking, Telvent and SAIC, August 2010*
- *DRAFT SAE J2354 Message Sets for Advanced Traveler Information System (ATIS), February 1, 2005*

1.6 Design Process

Object oriented analysis and design techniques were used in creating this design. The design is documented using diagrams that conform to the Unified Modeling Language (UML) specification. Listed below is the process that was used to create the work products contained in this document:

- Use case diagrams were created to reflect the requirements.
- Class diagrams were created that show possible new classes that will be developed for the system, pre-existing classes that are expected to be used, and the relationships between both new and existing classes.
- Sequence diagrams were created from the use case and class diagrams that show how the classes and their methods will be used to perform each use case.

The design was partitioned into packages that group classes that have a high degree of dependency. This partitioning is documented in a package diagram.

A deployment diagram was created to show which network server(s) would contain each application required by the software module described in this document.

1.7 Design Tools

The work products contained within this document are extracted from the Telelogic Tau UML Suite design tool. This design tool maintains a repository of all project design data stored in a database management system (DBMS). The diagram notation used within this tool is based on the Unified Modeling Language (UML) for Object-Oriented Development, developed by Grady Booch, Ivar Jacobson, and James Rumbaugh. Maintaining the project design data within this tool allows the storage of all documentation within the Models section in a database with the corresponding UML diagram. DocExpress is a tool used to extract the data from the Telelogic Tau system into a Microsoft Word document.

Within the Telelogic Tau tool, the design is contained in the pbfi_repos database in the UPPParking and UPPParkingMDBI subsections of the TravInfo®\Analysis\WCDemo section.

1.8 Development Tools

The following tools will be used to develop the source code for this module:

- Java Development Environment v1.6.
- Jakarta Ant v1.7.0.
- Telelogic Synergy/CM Release 6.4 Configuration Management System
- Microsoft SQL Server v2005
- SonicMQ JMS Server v7.6 or later

1.9 Work Products

This design contains the following work products:

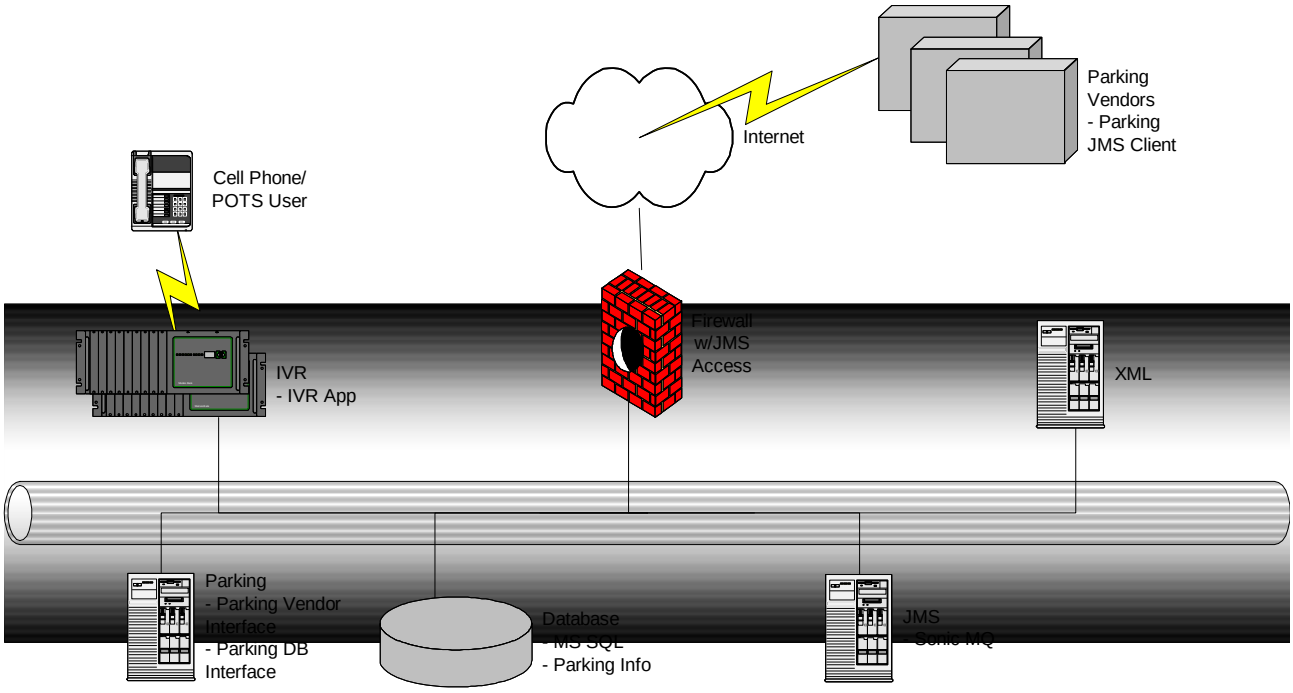
- UML Use Case diagrams that show the operations that are performed by the system.
- UML Class diagrams that show the software objects that will allow the system to be used as described in the Use Case diagrams and implement the system requirements.
- UML Sequence diagrams that show how the classes interact via their methods to accomplish usages of the system.
- A UML Deployment diagram showing the servers where each software component will be deployed.

2 Software Architecture

This section covers the architecture chosen for the UPP Real-time Parking system.

2.1 Real-Time Parking System

Initially, the 511 Parking Information feature will provide parking space availability, garage and on-street metered parking locations, and pricing information for municipal parking lots in downtown San Francisco on 511 phone and web systems, the MY 511 system, and in the 511 traffic data feed to private and public sector Information Service Providers (ISPs). The system will allow 511 users to set preferences for parking information through the personalized MY 511 service. The parking information will be received from San Francisco Municipal Transportation Authority (SFMTA)'s UPP SFpark project for one (1) parking lot, fourteen (14) garages, and six thousand (6000) on-street metered spaces. [1]



[1] The parking lots at four Caltrain stations that are being instrumented as part of the Department of Transportation's Safe Trip-21 initiative are not included in this phase of the UPP Parking Information System.

Figure 1. RT Parking Logical Diagram

Figure 1 shows the logical relationships between the entities of the UPP Parking System. Parking vendors will provide information in a pre-determined XML format to the parking system via a single or cluster of one or more JMS servers. Static information about parking facilities will be provided to the RT parking system on a request basis through pre-configured queues (one per parking vendor) on the JMS server. Dynamic information concerning parking space availability and pricing will be published by the vendor to the JMS server when status changes are detected. The static and dynamic data sent to the parking interfaces will be stored in the 511 IVR and Master Parking Database. Data published by the parking vendors will also be received by the Multi Modal Trip Planner and a future parking web site through the Master Parking Database. The Master Database Interface is responsible for reading or writing to the Master Parking Database.

The Parking Vendor Interface receives static and dynamic data from parking vendors.. The dynamic data is published to a JMS topic to make it available to the 511 database interface and the Multimodal Trip Planner. Static data is sent to the Master database interface. The MDBI responds to requests for reading and writing to the parking master database. The responses are sent back to the request originator, such as the PVI. Parking vendors will be required to register their intent to become a data publisher on the UPP Parking system prior to sending data. The following diagram illustrates the sequences and interfaces for data exchange under the UPP Parking System:

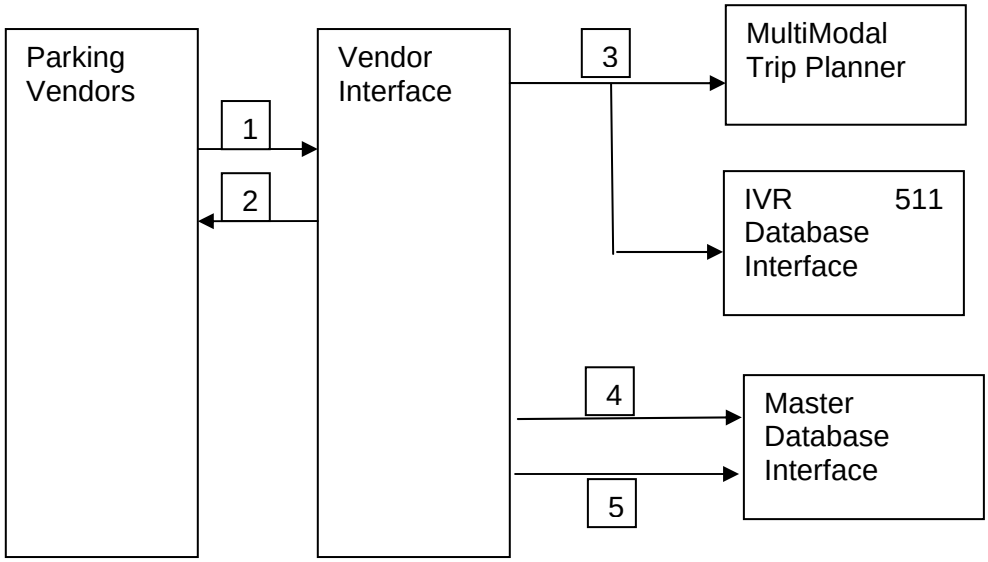


Figure 2. JMS Connection Diagram

1. JMS Topic Parking Vendor XYZ publish to PVI
2. JMS Queue PVI sending to Parking Vendor XYZ
   These are requests of the vendors.
3. JMS Topic PVI publishing to the Multi Modal Trip Planner and other subscribers.

4. JMS Read Queue to Master MDBI. Number four (4) is used for Read request/response. PVI, MMTP, and other authorized 511 subsystems may send requests to the MDBI for reading.

5. JMS Write Queue to Master MDBI. Number five (5) is used for Write request/response to save data in the MDBI. PVI, MMTP and other authorized 511 subsystems may send requests to the MDBI that modify the database.

Prior to connecting to the automated parking system, participating parking vendors will receive the following information offline:

1. The URL of the JMS broker.
2. The JMS user name.
3. The JMS password.
4. The JMS publish topic name (for vendor sends to the PVI)
5. The vendor receive queue name (will be FROMPVI2XXXXX plus 5 digits of their vendorID).

The PVI is a Java program that runs on a Windows server. The PVI receives vendor data, stores the static data in the master 511 database through the MDBI. It also publishes dynamic data to make available to the 511 IVR database interface and the MMTP.

The queue name is FROMPVI2XXXXX where XXXXX is a 5 digit vendor ID that will be assigned to each vendor. For example, if vendor SF Park has vendorID of 01234, SF Park will receive requests on queue name FROMPVI201234.

2.2 Communications

The parking vendors must have TCP connectivity to the JMS server port through any interposed firewall/proxy servers. The parking application server(s) will reside behind a firewall in the 511.org domain preferably within the same subnet as the JMS and parking database servers.

2.3 Logging

Error/exception messages, system messages, and debug information will be logged to time-stamped files. The severity level of the information logging will be controlled by a properties file. XML data will be logged in a separate file from system message data.

2.4 Error/Exception Processing

The software handles all expected exceptions. Expected exceptions are problems from which the system can recover from such as problems with input data. The default action when an exception occurs is to either handle the exception where it occurs or to throw a system-specific exception. Unexpected exceptions or errors will typically cause a stack trace to be written to the log file to aid in analyzing the problem. Errors with requested input data will be returned to the vendor via an XML return message. Errors from which the system cannot recover (e.g. heap space/memory exhaustion, full hard drive, loss of internet connectivity, etc.) will cause termination of the system.

2.5 Database Objects

The following sections contain the definitions for the database objects that will be used for this module. Two databases are used: an IVR parking database, and a parking master database. The following tables will be created in both databases.

2.5.1 Database Table: tblVendors

This table contains parking vendors.

Table: tblVendors					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
VendorID	int	YES	NO	(none)	Primary key
VendorName	nvarchar(128)	NO	NO	(none)	Name of vendor
VendorStreet	nvarchar(128)	NO	NO	(none)	Street address of vendor
VendorCity	nvarchar(128)	NO	NO	(none)	City of vendor
VendorState	nvarchar(2)	NO	NO	(none)	State Abbreviation
VendorZip	nvarchar(10)	NO	NO	(none)	ZIP of vendor
VendorPhoneOnSite	nvarchar(20)	NO	NO	(none)	Vendor on-site phone (nnn) nnn-nnnn
VendorPhone2	nvarchar(20)	NO	NO	(none)	Vendor on-site phone (nnn) nnn-nnnn
VendorEmail	nvarchar(128)	NO	NO	(none)	Vendor email
VendorWebsite1	nvarchar(128)	NO	NO	(none)	Vendor website
VendorWebsite2	nvarchar(128)	NO	NO	(none)	Vendor website
JMSQToVendor	nvarchar(128)	NO	NO	(none)	JMS queue name assigned to the vendor

2.5.2 Database Table: tblParkingFacility

This table contains information about parking facilities.

Table: tblParkingFacility					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
ParkingFacID	int	Yes (1)	No	(none)	0-65535
VendorID	int	No	No		0-65535
Lotname	nvarchar(30)	No	Yes		
City	nvarchar(64)	No	Yes		
Neighborhood	nvarchar(64)	No	Yes		
ParkingType	tinyint	No	No		Open Lot (1), garage (2), permit (3), contract (4), free (5), pay (6), other (7) 8-149 reserved 150-255 local use
PaymentOptions	nvarchar(128)	No	Yes		unknown cash personal check visa master card american express diners club other
SpecialServices	nvarchar(128)	No	Yes		
MaxTimeLimit	int	No	Yes	0	Time interval in minutes
TotalSpaces	int	No	Yes	0	Total number of spaces in the facility
dttmLastUpdate	datetime	No	NO	Current time	Timestamp given by vendor

2.5.3 Database Table: tblBusinessHours

This table contains stop definitions for parking facility business hours.

Table: tblBusinessHours					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
parkingFacID	int	Yes	No	(none)	Foreign key to tblParkingFacilities
DayTypeID	tinyint	Yes	No	(none)	Key to tblDayTypeLookup
OpenTime	nvarchar(10)	No	No	(none)	Time in 24 hr format (i.e. HH:MM)
CloseTime	nvarchar(10)	No	No	(none)	Time in 24 hr format (i.e. HH:MM)

2.5.4 Database Table: tblPriceSchedule

This table contains a list of the prices for parking facilities.

Table: tblPriceSchedule					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
ParkingFacID	int	Yes	No	(none)	Foreign key to tblParkingFacilities
Sequence	tinyint	Yes	No	(none)	order
Schedule	int	No	No		Schedule number
ScheduleEntry	int	No	No		Schedule entry number
Priority	tinyint	No	Yes		Schedule priority. Higher priorites can be used to override lower priority schedules.

Earlybirdspecial	Bit	No	Yes		Indicates whether schedule is earlybird
DayTypeID	tinyint	No	Yes	(none)	Key into tblDayTypeLookup
SpaceTypeID	tinyint	No	Yes		Key to tblSpaceType
Starttime	datetime	No	Yes		Start time of schedule
StarttimeOfst	nchar(5)	No	Yes		Time zone offset e.g. -8000
Endtime	datetime	No	Yes		End time of schedule
EndtimeOfst	nchar(5)	No	Yes		Ent time time zone offset
Sttimemaxmin	nchar(6)	No	Yes		Start hour of day for min,max
EndTimeMaxMin	nchar(6)	No	Yes		End hour of day for min, max
priceFirstpaymnt	money	No	Yes		Price of first payment (minimum)
PriceMax	money	No	Yes		Max price (min, max may be the same)
Description	nvarchar(128)	No	Yes		Description of Max, Min costs
TimeInvl	nvarchar(16)	No	Yes		Time interval in minutes
Rate	money	No	Yes		Cost per time interval
RateDescription	nvarchar(128)	No	Yes		Any text e.g. "$5.00 for first hour"
PriceFirstpaymentI	nvarchar(8)	No	Yes		Either usDollar or usPenny
PriceMaxi	nvarchar(8)	No	Yes		Either usDollar or usPenny
RateI	nvarchar(8)	No	Yes		Either usDollar or usPenny
DiscountDollar	money	No	Yes		Discount in dollars
DiscountPct	float	No	Yes		Discount in percent

2.5.5 Database Table: tblMail

This table contains the email address to which reports will be sent.

Table: tblMail					
Column Name	**Data Type**	**Primary**	**Nullable**	**Default**	**Description**

		Key			
MailAddress	nvarchar(128)	Yes	No	(none)	Email address used for reporting

2.5.6 Database Table: tblFacilityInfo

This table contains information about each parking facility.

Table: tblFacilityInfo					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
parkingFacID	nvarchar(32)	Yes	No	(none)	Foreign key to tblParkingFacilities
Parkingsitename	nvarchar(128)	No	No	(none)	Name of parking facility site
Ownershipagencytype	nvarchar(8)	No	Yes	(none)	Public, Private
Structuretype	nvarchar(16)	No	Yes	(none)	Garage, Streetlevel Lot, Business parking
onsiteSecurityInfo	Bit	No	Yes	No	Security available onsite
Peakusagetime	nvarchar(16)	No	Yes	(none)	Time of day of peak number of unavailable spaces in HH:MM AM/PM
contactphoneOnsite	nvarchar(16)	No	Yes	(none)	Phone number of onsite contact (nnn) nnn-nnnn
Contactphone2	nvarchar(16)	No	Yes	(none)	Phone number of onsite contact (nnn) nnn-nnnn
contactphoneCustSvc	nvarchar(16)	No	Yes	(none)	Phone number of customer service (nnn) nnn-nnnn
contactFax	nvarchar(16)	No	Yes	(none)	Fax number of onsite contact (nnn) nnn-nnnn

contactphoneSecurity	nvarchar(16)	No	Yes	(none)	Phone number of onsite security (nnn) nnn-nnnn
contactphoneTTY	nvarchar(16)	No	Yes	(none)	Phone number for hearing disabled (nnn) nnn-nnnn
contactEmail	nvarchar(16)	No	Yes	(none)	Email address of onsite contact
contactWebsite1	nvarchar(32)	No	Yes		Web site address of facility/site
contactWebsite2	nvarchar(32)	No	Yes		Web site address of facility/site
Transitparkingallowed	nvarchar(4000)	No	Yes		Agencies delimited by semicolon
Realtimeinfoavailable	bit	No	Yes		
InAndOut	bit	No	Yes		Allow leave and return
HasOvernightParking	bit	No	Yes		
HasATM	bit	No	Yes		
validationAvail	bit	No	Yes		Discounts for nearby shops
validationDescription	nvarchar(128)	No	Yes		
Haschargingport	nvarchar(8)	No	Yes		“True” or “False”
Hasemergencycallbox	nvarchar(8)	No	Yes		
paymentdiscountAAA	money	No	Yes		
paymentdiscountSenior	money	No	Yes		
paymentdiscountCorporate	money	No	Yes		
paymentdiscountGroup	money	No	Yes		
paymentdiscountHybrid	money	No	Yes		
paymentdiscountStudent	money	No	Yes		
paymentdiscountTaxExempt	bit	No	Yes		
paymentdiscountResident	money	No	Yes		
paymentdiscountCarpool	money	No	Yes		
Paycashieratexit	nvarchar(8)	No	Yes		
Kioskpaystations	nvarchar(8)	No	Yes		
Paycashinsertslots	nvarchar(8)	No	Yes		
Epay	nvarchar(8)	No	Yes		
Payonline	nvarchar(8)	No	Yes		
Servicetypes	nvarchar(4000)	No	Yes		

ReservAvailable	nvarchar(16)	No	Yes		
reservMethod	nvarchar(16)	No	Yes		
reservPhone	nvarchar(16)	No	Yes	(none)	Phone number for reservations (nnn) nnn-nnnn
reservEmail	nvarchar(16)	No	Yes	(none)	Email for reservations
reservOnline	nvarchar(128)	No	Yes	(none)	
reservAddInfo	nvarchar(128)	No			
Lockersize	float	No	No	0.0	
Lockercost	money	No	No	0	
Lockerquantity	int	No	No	0	
Lockermaxoccupancy	int	No	No	0	
AdditionalDesc	nvarchar(4000)	No	Yes	(none)	Additional info

2.5.7 Database Table: tblCrossStreetProfile

This table contains concerning links/cross streets.

Table: tblCrossStreetProfile					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
parkingFacID	int	Yes	No	(none)	Foreign key to tblParkingFacilities
PointOrLink	nvarchar(8)	Yes	No	(none)	“Point” or “Link”
onStreet	nvarchar(128)	No	No		Point
atStreet	nvarchar(128)	No	No		Point
toStreet	nvarchar(128)	No	No		link
fromStreet	nvarchar(128)	No	No		link
pointLatitude	float	No	No	0.0	Latitude of the point (i.e. cross street) -90.0 <= n <= 90.0

pointLongitude	float	No	No	0.0	Longitude of the point (i.e. cross street) -180.0 <= n <= 180.0
startLatitude	float	No	No	0.0	Latitude of the starting point of the link -90.0 <= n <= 90.0
startLongitude	float	No	No	0.0	Longitude of the starting point of the link -180.0 <= n <= 180.0
endLatitude	float	No	No	0.0	Latitude of the ending point of the link -90.0 <= n <= 90.0
endLongitude	float	No	No	0.0	Longitude of the ending point of the link -180.0 <= n <= 180.0

2.5.8 Database Table: tblGridProfile

This table contains parking grid information.

Table: tblGridProfile					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
parkingFacID	int	Yes	No	(none)	Foreign key to tblParkingFacilities
Sequence	int	Yes	No	(none)	Counter used to maintain ordering
gridID	int	No	No	0	Numeric 0 to 255
gridLatitude	int	No	No	0	degrees *100 and rounded to nearest int
gridLongitude	int	No	No	0	degrees*100 and rounded to nearest int
gridZoom	int	No	No	0	

Table: tblAssociatedFiles

Column Name	Data Type	Primary Key	Nullable	Default	Description
parkingFacID	int	Yes	No	(none)	Foreign key to tblParkingFacilities
Filename	nvarchar(128)	Yes	No		
Description	nvarchar(512)	No	Yes		
FileDataBase64	varchar(max)	No	Yes		File data in base 64

Table: tblEntrances					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
parkingFacID	int	Yes	No	(none)	Foreign key to tblParkingFacilities
EntranceLocation	nvarchar(256)	Yes	No		
EntranceHeight	int	No	Yes		Height in centimeters

Table: tblParkingAccess					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
parkingFacID	int	Yes	No	(none)	Foreign key to tblParkingFacilities
AccessType	nchar(12)	Yes	No		
AccessFeature	nvarchar(128)	Yes	No		
AdditionalInfo	nvarchar(128)	No	Yes		

Table: tblSpacesByType					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
parkingFacID	int	Yes	No	(none)	Foreign key to tblParkingFacilities
SpaceType	nvarchar(16)	Yes	No		
Spaces	int	No	No		Available spaces

Table: tblSpaceTypeLookup					
Column Name	**Data Type**	**Primary**	**Nullable**	**Default**	**Description**

		Key			
SpaceTypeId	int	Yes	No		Primary key
SpaceType	nvarchar(50)	No	No		Space Type (e.g. unknown, auto, compact, handicapped, etc.)

Table: tblDayTypeLookup					
Column Name	**Data Type**	**Primary Key**	**Nullable**	**Default**	**Description**
DayTypeId	int	Yes	No		Primary key
DayType	nvarchar(50)	No	No		Day Type (e.g. holiday, Sunday, weekend, etc.)

3 Models

The following sections provide models and diagrams that show the detailed design of the UPP Real Time Parking system software. These sections contains one or more use case diagrams, class diagrams, and sequence diagrams. The use case diagrams serve to show the types of ways the users and external systems may interact with the system. The class diagrams show software objects that comprise the system and their relationships to each other. The sequence diagrams show how the objects defined in the class diagram(s) interact to accomplish specific uses of the system.

3.1 Use Cases

3.1.1 ParkingUseCases (Use Case Diagram)

This use case diagram describes how the texting services receive data from the parking vendors and store the data in the parking and IVR databases.

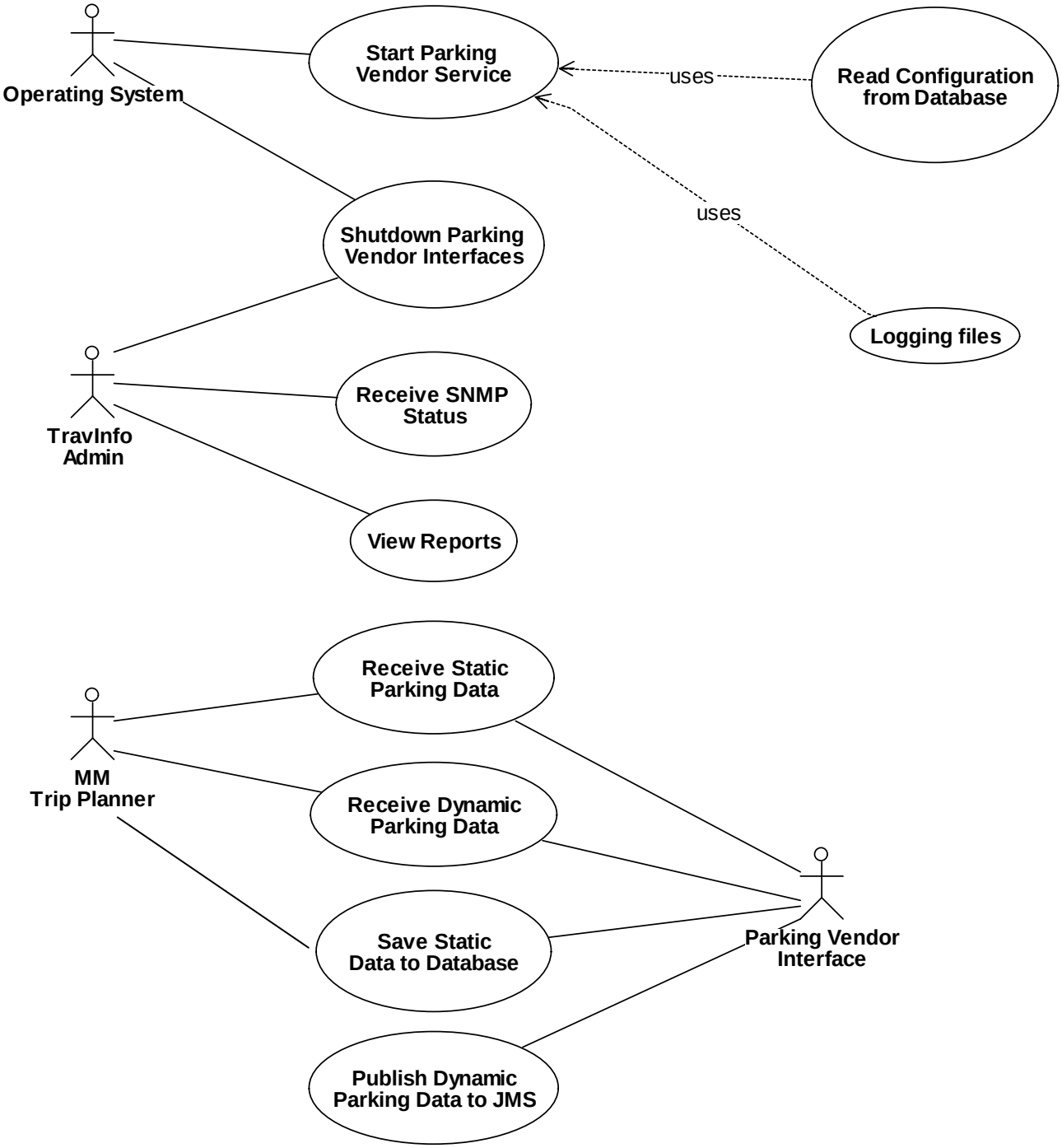


Figure 3. ParkingUseCases (Use Case Diagram)

3.1.1.1 Logging files (Use Case)

Writes status and error information to a log file.

3.1.1.2 MM Trip Planner (Actor)

The MM Trip Planner receives static and dynamic parking information (i.e. total spaces and parking facility occupancy) and may request this static and dynamic information from the

master database. Also, the MM Trip Planner writes to the master database.

3.1.1.3 Operating System (Actor)

The operating system starts the service at system startup.

3.1.1.4 Parking Vendor Interface (Actor)

The parking interface will maintain a Java Message Server (JMS) that will contain updates to the parking vendor configuration, and parking facility occupancy updates. The parking service will subscribe to updates to the JMS.

3.1.1.5 Read Configuration from Database (Use Case)

The parking service will read persisted parking configuration information from the database.

3.1.1.6 Shutdown Parking Vendor Interfaces (Use Case)

When the parking service is shutdown, the service will un-subscribe to vendor JMS updates and shutdown all internal processes.

3.1.1.7 Receive SNMP Status (Use Case)

All changes in the status of the application including errors are sent as alerts to an SNMP manager application or the network to respond to queries for program status.

3.1.1.8 Start Parking Vendor Service (Use Case)

3.1.1.9 Publish Dynamic Parking Data to JMS (Use Case)

Publishes parking data to the JMS service in order to be used by external systems.

3.1.1.10 Receive Dynamic Parking Data (Use Case)

Receives parking data from the JMS server in response to data sent by a parking vendor.

3.1.1.11 Receive Static Parking Data (Use Case)

Receives parking data from the JMS server in response to a request.

3.1.1.12 Save Static Data to Database (Use Case)

Writes parking data to the system database.

3.1.1.13 TravInfo Admin (Actor)

The administrator is responsible for the operation of the service. The administrator may re-start the service after a shutdown, perform a forced shutdown on the system, manually request an parking configuration update, view reports prepared by the system, or view the status of the service via SNMP.

3.1.1.14 View Reports (Use Case)

Reports, including a parking vendor configuration report, will be generated.

3.2 Class Diagrams

3.2.1 UPPParkingClasses (Class Diagram)

This class diagram shows the relationship between the classes comprising the main class of the

Parking service application. This diagram shows the hierarchical and containment/composition relationships between the classes.

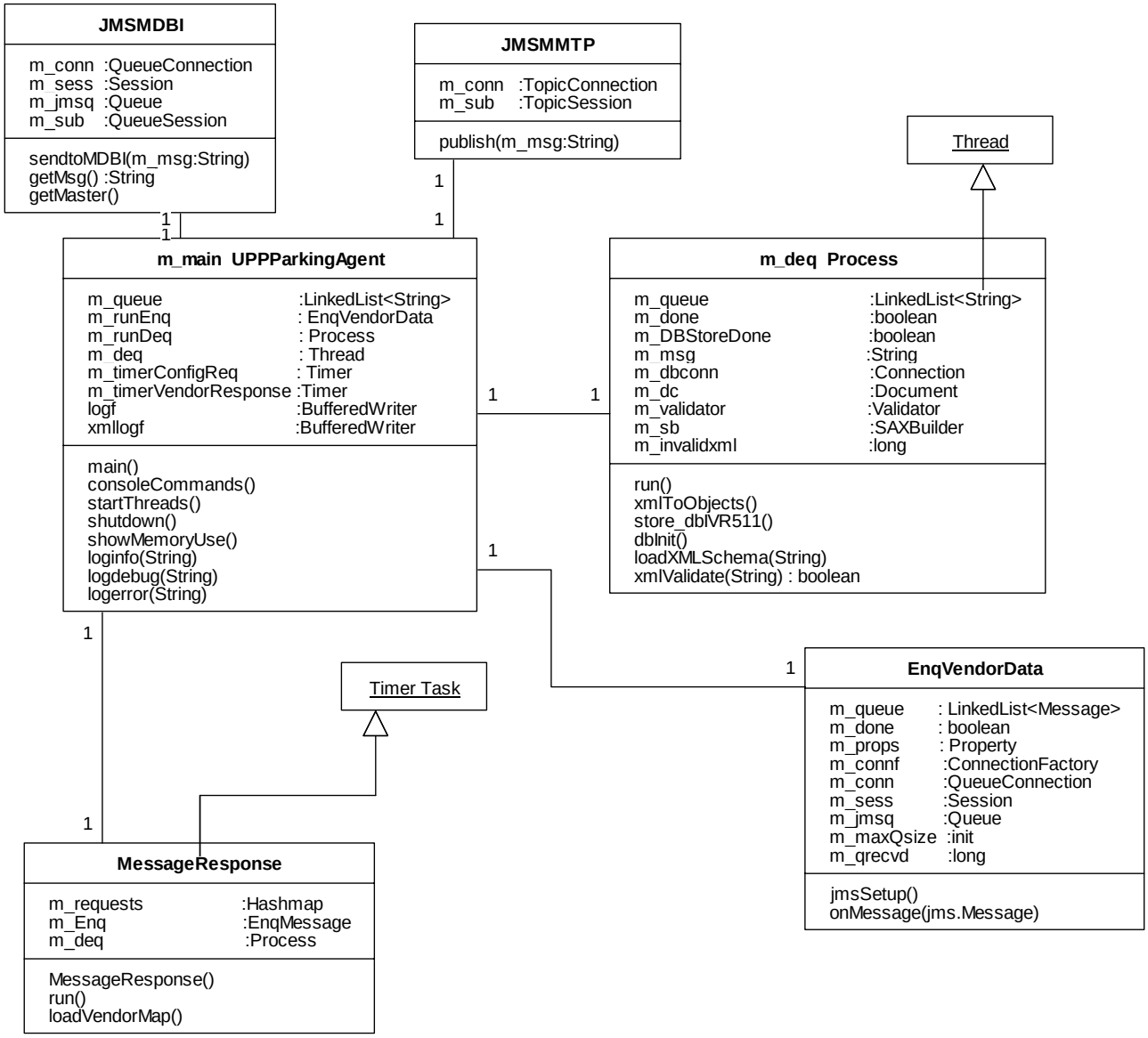


Figure 4. UPPParkingClasses (Class Diagram)

3.2.1.1 EnqVendorData (Class)

This class provides two (2) methods: jmsSetup() initializtion of JMS for receiving vendor messages. onMessage() is the handler that fires for each received message.

3.2.1.2 JMSMDBI (Class)

Write to IVR db. Either parses XML and calls stored procedures, or sends XML to another interface. There are three (3) methods: dbInit() initialize database connection loadXMLSchema() loads the schema xsd, XMLValidate(String) called for each incoming message to validate.

3.2.1.3 JMSMMTP (Class)

Sends messages to JMS to be published to the topic for the MMTP.

3.2.1.4 MessageResponse (Class)

Checks whether responses have timed out and, if so, throws trap and logs message.

3.2.1.5 m_deq Process (Class)

Processes all incoming messages from vendors and from the MDBI.

3.2.1.6 m_main UPPParkingAgent (Class)

Main class of application. Initializes structures and logging. This class has five (5) methods: main() start of PVI application consoleCommands() waits for command from console, startThreads() initializes threads, shutdown() shutdown procedure, loginfo() used for logging

3.2.2 MasterDatabaseInterfaceClasses (Class Diagram)

This class diagram shows the relationship between the classes comprising the main class of the Parking Master Database interface. This diagram shows the hierarchical and containment/composition relationships between the classes. The valid requests that are sent to the MDBI are the following:

1. Write Vendor inventory to master db
2. Write Vendor name, id to master db
3. Read and return vendor inventory from master
4. Return list of vendor name, ids

~~The Multimodal Trip Planner will only send requests 3 and 4~~. ~~The PVI can issue all four~~. Unsolicited data sent by parking vendors will be forwarded to the MDBI as requests to store in the master db when they are of the <staticfmt>.

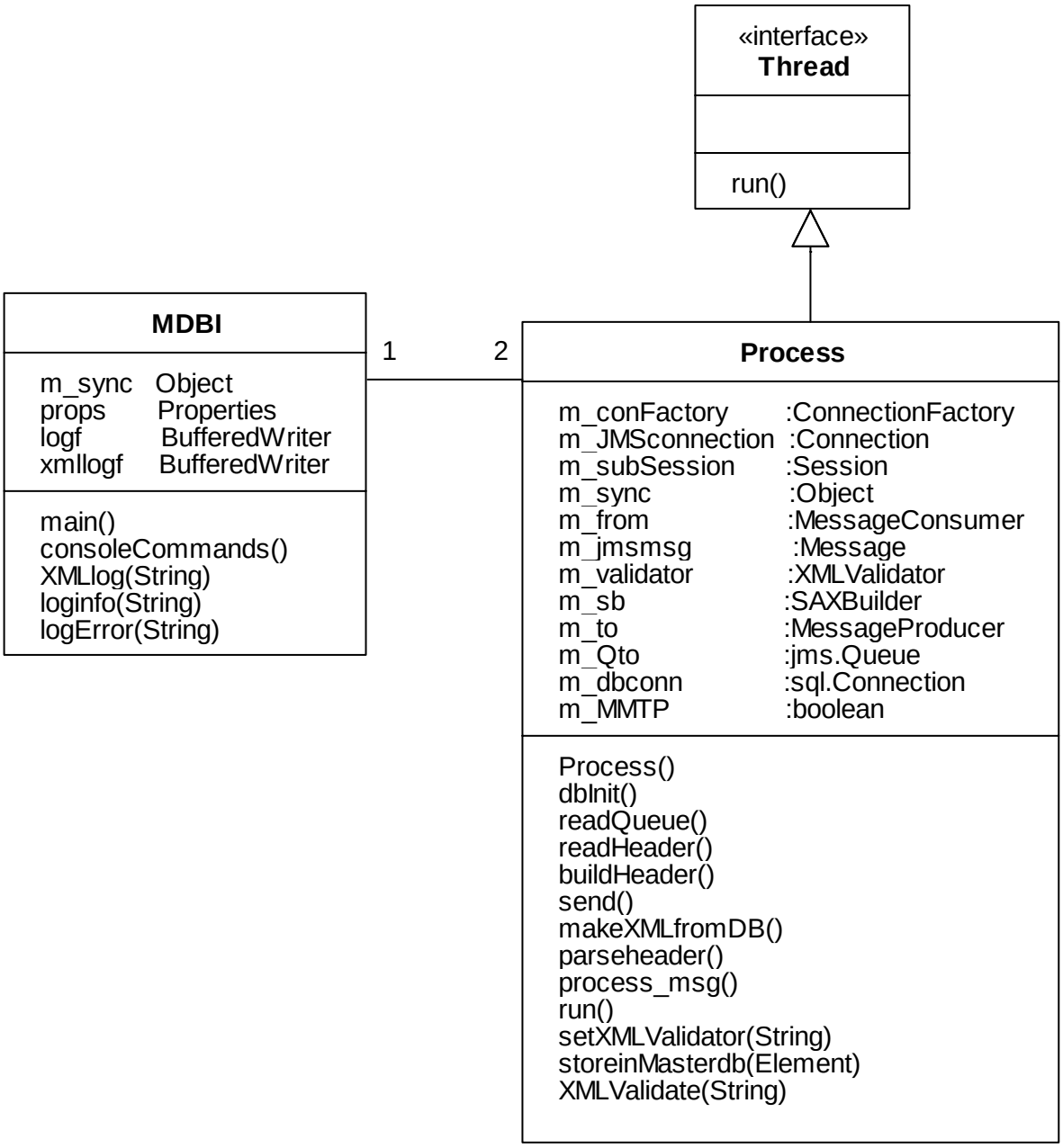


Figure 5. MasterDatabaseInterfaceClasses (Class Diagram)

3.2.2.1 MDBI (Class)

This class contains the main thread of the application. At startup, the logger is initialized and the xsd schema is loaded. The m_PVI and m_MMTP threads are started. These threads are instances of class Process. m_sync - single object used to serialize access to the database between the threads. m_props - property object for the property file logf, xmllogf log file references

Methods:

XMLlog() standard logging proc

Loginfo() log message to file for information messages

Logerror() logging to file for error messages

Createnewlogfiles()

Main() entry point of thie MDBI application.

consoleCommands() -- accepts Quit, Memory as commands.

3.2.2.2 Process (Class)

A single instance of class Process is created for PVI processing. A second instance is created for the MMTP processing. Variables are: m_conFactory, m_dbconn, m_dc, m_to, m_from, m_JMSConnection, m_Qfrom , m_Qto, m_subSession, m_synch.
Methods are:

run() - main loop of the thread and uses JMS MessageConsumer receiveNoWait() to poll for an incoming message from the JMS queue. After validation, it parses the header to find the msg action using process_msg()
Uses JMS MessageProducer.send() to send a message to the JMS queue.

dbInit() - Called at startup to initialize the connection to the parking master database.

readQueue() - read requests from PVI or MMTP JMS queue buildResponse() - use Jdom to create xml response

sendResponse() - send using JMS either to PVI or to the MMTP masterDB() – read or write to master db.

JMSinit() - Called at startup to initialize the JMS makeHeader() - make header for xml response

makeXMLbody() - after a read from the db, the xml msg is created. makeXMLfromDB() parseheader(Element) – Jdom is used to validate and parse the xml.

process_msg(String) - processes incoming message, reads header and either reads from db or writes to db. setXMLValidator(String) - load the schema at startup XMLValidate(String) - validate xml against xsd schema

storeinMasterdb() readfromMaster()

3.2.2.3 Thread (Class)

Java library class.

3.3 Sequence Diagrams

3.3.1 UPPParkingSequence (Sequence Diagram)

The sequence is as follows upon startup:

1. the m_main begins and initializes logging, JMS, and threads.
2. requests are sent to the MDBI to read from the Master DB.
3. vendor input is activated

Following initialization, continual input is received from vendors and responses from the MDBI.

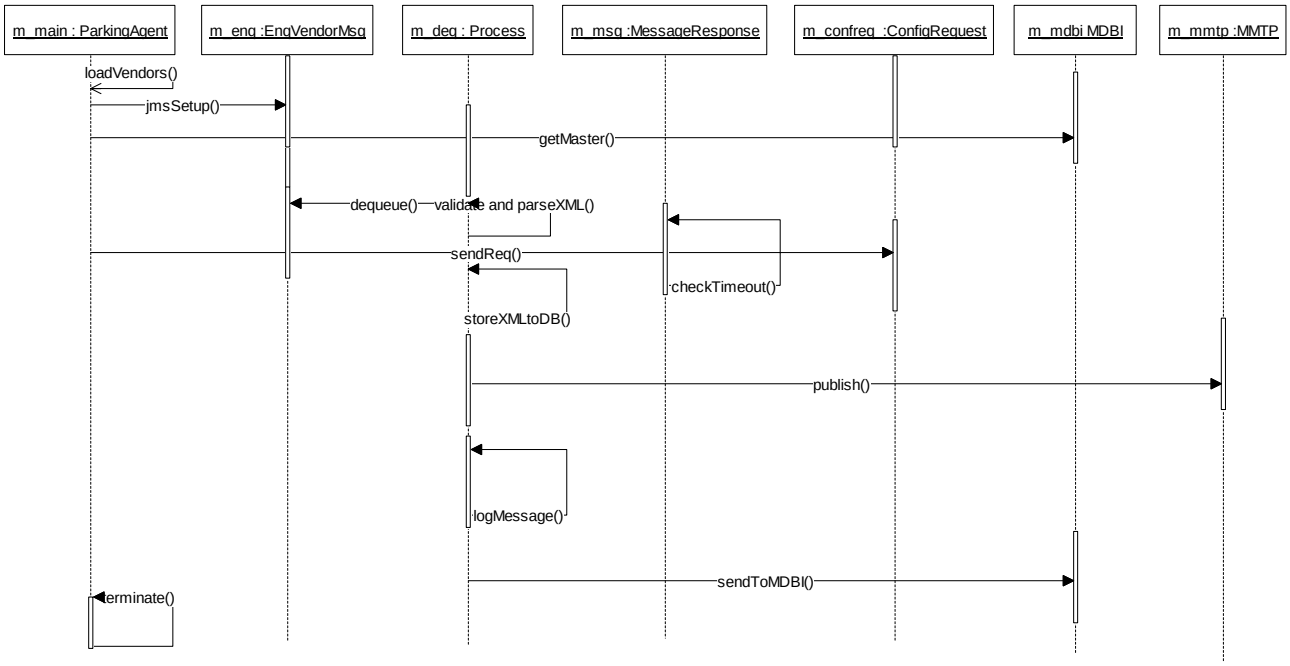


Figure 6. UPPParkingSequence (Sequence Diagram)

3.3.2 MDBIsequence (Sequence Diagram)

The threads for the PVI and MMTP are essentially identical. Both dequeue message requests and generate responses.

After initialization of JMS and the database, the process threads are created for PVI and MMTP. These threads continually run in a loop listening for input, parsing incoming xml messages, reading/writing to the database, building the response message, and sending the response.

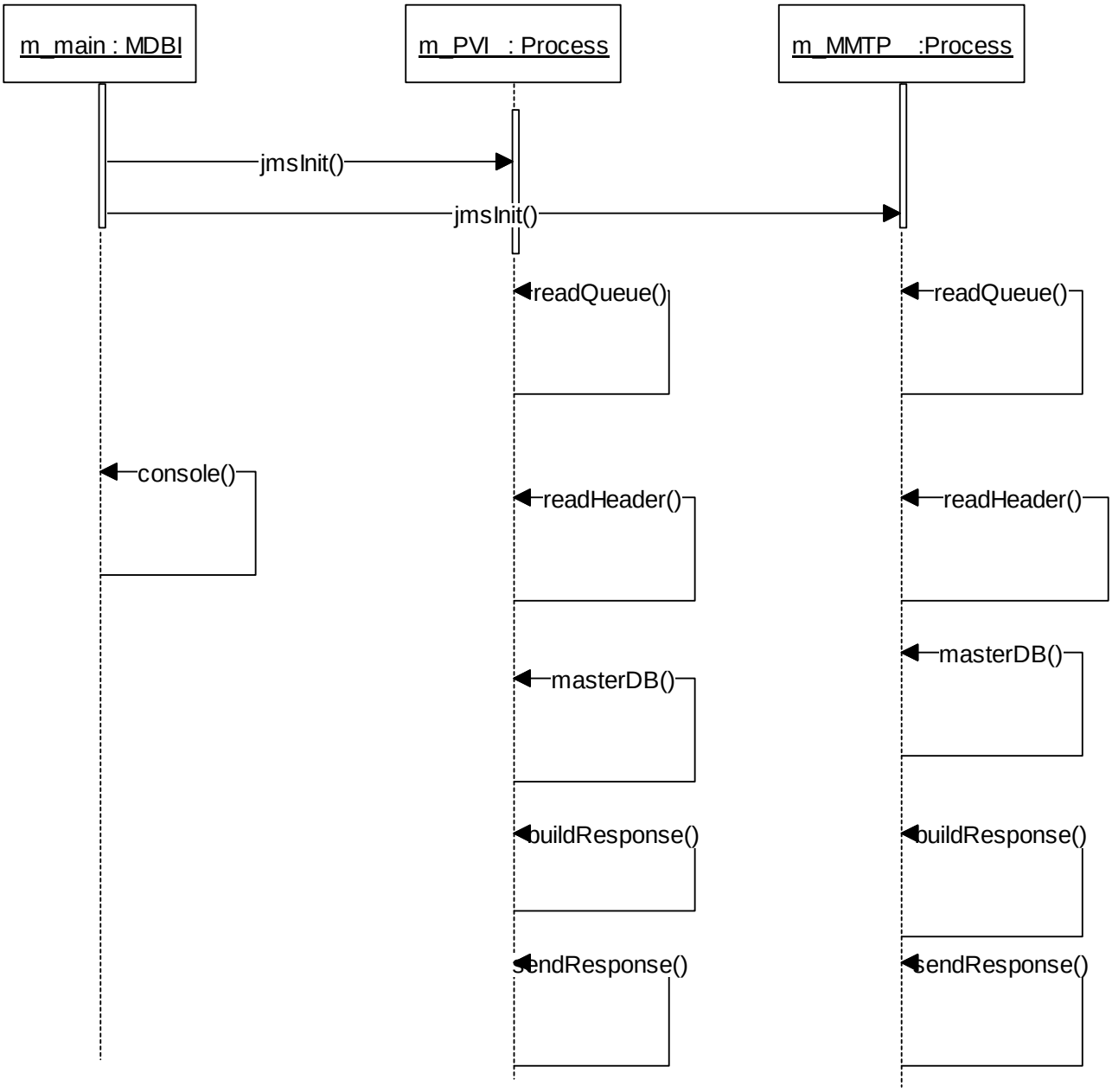


Figure 7. MDBIsequence (Sequence Diagram)

4 Deployment

4.1 Deployment (Deployment Diagram)

The deployment diagram shows where each of the system modules related to the parking service will execute. The parking service typically, but not necessarily, will run on its own server. Upon startup, the system will read parking configuration data from the customer-supplied database server and register for notifications of new messages from the Java Message Service (JMS) that will execute on a server provided by the customer. The system may also periodically read data from parking vendors via a JMS request. Vendors may write parking configuration changes and/or parking facility occupancy changes to the JMS. The system will write changes to the database.

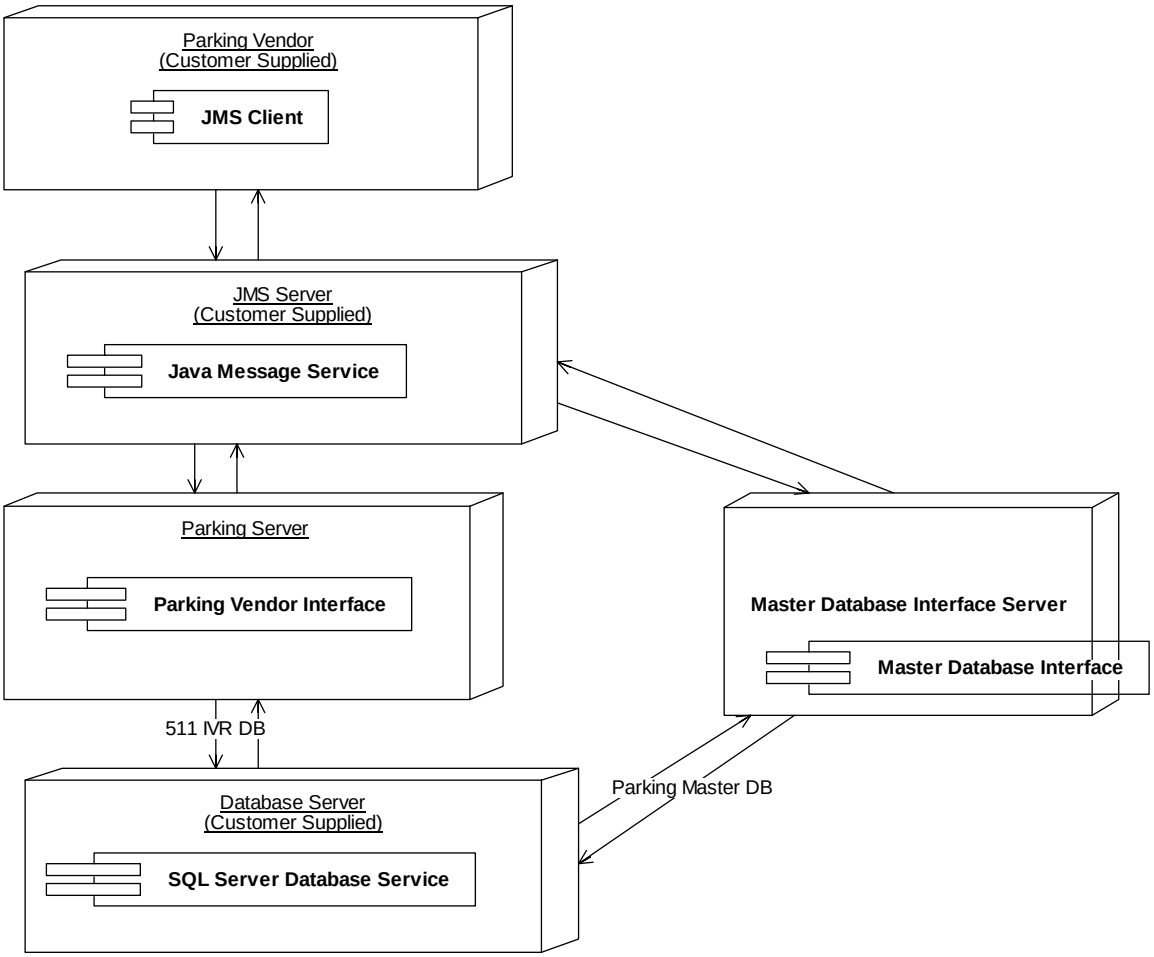


Figure 8. Deployment (Deployment Diagram)

Appendix A. Stored Procedure Interface Definitions

This appendix contains definitions of the interfaces of the stored procedures needed for UPP Parking. The UPP Parking stored procedures that will be used to read/write to the Master database via a JDBC connection are shown below. These procedures can also be used for the IVR db. Stored procedures return an int = 0 for success. Each table will have update, get, and delete procedures. The update procedures should check whether rows exist and if so, perform update operations. If the rows do not exist yet, an insert operation should be performed. The delete procedures should delete all rows of the table.

```
PROCEDURE UPP_UpdateParkingVendor
       @VendorName    nvarchar(32),
       @vendorID      int,
       @jmsQtoVendor  nvarchar(16)

PROCEDURE  UPP _GetParkingVendors

PROCEDURE  UPP_DeleteVendors

PROCEDURE UPP_UpdateParkingFacility
       @parkingFacID      int,
     @lotname           nvarchar(64),
     @vendorID          int,
     @spacesTotal       int,
     @city              nvarchar(64),
     @nhood             nvarchar(64),
     @parkingType       nvarchar(32),
       @paymentOption     nvarchar(32),
       @maxTimeLimit      nvarchar(32),
     @specialservices   nvarchar(128),
     @dttmLastUpdate    datetime,
     @dttmfilltime      datetime



PROCEDURE UPP_getFacilityInfo @parkingFacID

PROCEDURE UPP GetAllFacilityIDs @vendorID
PROCEDURE UPP_DeleteParkingFacility

PROCEDURE  UPP_UpdateBusinessHours
       @parkingFacID      int),
       @daytype           tiny int,
       @opentime          nvarchar(10),
       @closetime         nvarchar(10)

PROCEDURE  UPP_GetBusinessHours @parkingFacID
PROCEDURE  UPP_DeleteBusinessHours


PROCEDURE   UPP_updatePriceSchedule
```

```
                    @ParkingFacID       nvarchar(32),
                    @schedule           int,
                    @scheduleentry      int,
                    @TimeInvl           int,
                    @DayType            nvarchar(16),
                    @SpaceType          nvarchar(16),
                    @rate              money,
                    @seq                int,
                    @startdatetime      nvarchar(32),
                    @startoffset        nchar(5),
                    @enddatetime        nvarchar(32),
                    @endoffset          nchar(5),
                    @pricei             nchar(10),
                    @pricefirstpayment money,
                    @pricefirstpayi     nchar(10),
                    @pricemaximum       money,
                    @pricemaxi          nchar(10),
                    @starttime          nchar(6),
                    @endtime            nchar(6),
                    @priority           int,
                    @earlybird          bit,
                    @rateDescrip        nvarchar(128)

PROCEDURE  UPP_GetPriceSchedule @parkingFacID
PROCEDURE  UPP_DeletePriceSchedule

PROCEDURE  UPP_UpdateMail
                  @mailaddress        nvarchar(32)

PROCEDURE  UPP_GetMail
PROCEDURE  UPP_DeleteMail

PROCEDURE UPP_UpdateFacilityInfo
       @parkingFacID              int,
       @parkingsitename           nvarchar(128),
       @ownershipagencytype       nvarchar(8),
       @structuretype             nvarchar(16),
       @peakusagetime             nvarchar(16),
       @contactphoneOnsite        nvarchar(16),
       @contactphone2             nvarchar(16),
       @contactphoneCustSvc       nvarchar(16),
       @contactFax                nvarchar(16),
       @contactphoneSecurity      nvarchar(16),
       @contactphoneTTY           nvarchar(16),
       @contactEmail              nvarchar(16),
       @contactWebsite1           nvarchar(16),
       @contactWebsite2           nvarchar(16),
       @numberofspaces            nvarchar(4000),
       @transitparkingallowed     nvarchar(4000),
       @realtimeinfoavailable     nvarchar(8),
       @haschargingport           nvarchar(8),
       @intandout                 bit,
       @hasovernightparking       bit,
       @hasATM                    bit,
       @validationsAvailable      bit,
       @validationdescription     nvarchar(128),
       @hasemergencycallbox       nvarchar(8),
       @parkingaccess             nvarchar(4000),
       @paymentdiscountAAA        money,
       @paymentdiscountSenior     money,
```

```
       @paymentdiscountCorporate money,
       @paymentdiscountGroup     money,
       @paymentdiscountHybrid    money,
       @paymentdiscountStudent   money,
       @paymentdiscountTaxExempt bit,
       @paymentdiscountCarpool   money,
       @paycashieratexit         nvarchar(8),
       @kioskpaystations         nvarchar(8),
       @paycashinsertslots       nvarchar(8),
       @epay                     nvarchar(8),
       @payonline                nvarchar(8),
      @servicetypes              nvarchar(4000),
      @reservAvailable           nvarchar(16),
      @reservMethod              nvarchar(16),
      @reservPhone               nvarchar(16),
      @reservEmail               nvarchar(16),
      @reservOnline              nvarchar(16),
      @reservAddInfo             nvarchar(16),
      @Lockersize                float,
      @lockercost                money,
      @lockerquantity            int,
      @lockermaxoccupancy        int,
      @additionalDesc            nvarchar(4000),
      @associatedfiles           nvarchar(4000)

  PROCEDURE UPP_GetFacilityInfo @parkingFacID
PROCEDURE UPP_DeleteFacilityInfo

PROCEDURE UPP_updateCrossStreetProfile
                  @ParkingFacID      int,
                  @pointorlink       nvarchar(8),
                  @onStreet          nvarchar(128),
                  @atStreet          nvarchar(128),
                  @toStreet          nvarchar(128),
                  @fromStreet        nvarchar(128),
                  @pointLatitude     float,
                  @pointLongitude    float,
                  @startLatitude     float,
                  @startLongitude    float,
                  @endLatitude       float,
                  @endLongitude      float

PROCEDURE UPP_GetCrossStreetProfile @ParkingFacID
PROCEDURE UPP_deleteCrossStreetProfile

PROCEDURE UPP_UpdateGridProfile
                  @parkingFacID       int,
                  @sequence           int,
                  @gridID             int,
                  @gridLatitude       int,
                  @gridLongitude      int,
                  @gridZoom           int

PROCEDURE UPP_GetGridProfile @parkingFacID
PROCEDURE UPP_DeleteGridProfile

PROCEDURE UPP_UpdateAssociatedFiles
            @parkingFacID   int,
            @Filename       nvarchar(128),
```

```
            @Description    nvarchar(128),
            @FileDataBase64 varchar(MAX)

 PROCEDURE  UPP_GetAssociatedFiles @parkingFacID
 PROCEDURE UPP_deleteAssociatedFiles


PROCEDURE  UPP_UpdateEntrances
            @parkingFacID      int,
            @EntranceLocation  nvarchar(256),
            @entranceHeight    int

PROCEDURE     UPP_getEntrances  @parkingFacID
PROCEDURE     UPP_deleteEntrances



PROCEDURE  UPP_UpdateParkingAccess
            @parkingFacID      int,
            @Location           nvarchar(256),
            @accesstype        nchar(12),
            @accessfeature     nvarchar(128),
            @additionalinfo    nvarchar(128)

PROCEDURE  UPP_GetParkingAccess  @parkingFacID
PROCEDURE  UPP_DeleteParkingAccess


PROCEDURE  UPP_UpdateSpacesByType
            @parkingFacID       int,
            @SpaceType          nvarchar(16),
            @spaces int

PROCEDURE  UPP_GetSpacesByType @parkingFacID
PROCEDURE    UPP_deleteSpacesByType

PROCEDURE UPP_ClearFacility @parkingFacID
```

Appendix B. ATIS Parking XSD

The following schemas are taken from DRAFT SAE J2354 Message Sets for Advanced Traveler Information System (ATIS) Feb 1, 2005

Data Frame: DF_ParkingLotInformation

Use: Used to describe information about a parking lot.

ASN.1 Representation:

```
ParkingLotInformation ::= SEQUENCE {
      messageHeader    MessageHeader,
      head             Head OPTIONAL,
      location         LRMS.LocationReference,
                       -- location for which this applies
      isForecast       BOOLEAN OPTIONAL,
                       -- forecast or predicted period
      coverageTime     ComplexTime OPTIONAL,
      forecastExpires  DateTimePair OPTIONAL,
      lot-ident        IdentityOrLocation,
      parkingType      TCIP.PI-ParkingType,
      percentFull      Percent OPTIONAL,
      lotDetail        LotInformation,
      prices           PriceSchedule OPTIONAL,
      generalStatus    ITIS.ITIScodesAndText OPTIONAL,
      furtherData      URL-Link OPTIONAL,
      tail             Tail OPTIONAL,
      ...              -- # LOCAL_CONTENT
      }
```

XML Representation:

```
<xs:complexType name="ParkingLotInformation" >
      <xs:sequence>
         <xs:element name="messageHeader" type="MessageHeader" />
         <xs:element name="head" type="Head"  minOccurs="0"/>
         <xs:element name="location" type="lrms:LocationReference" />
         <!-- location for which this applies -->
         <xs:element name="isForecast"  minOccurs="0">
            <xs:simpleType>
               <xs:restriction base="xs:boolean"/>
            </xs:simpleType>
         </xs:element>
         <!-- forecast or predicted period -->
         <xs:element name="coverageTime" type="ComplexTime"  minOccurs="0"/>
         <xs:element name="forecastExpires" type="DateTimePair"  minOccurs="0"/>
         <xs:element name="lot-ident" type="IdentityOrLocation" />
         <xs:element name="parkingType" type="tcip:PI-ParkingType" />
         <xs:element name="percentFull" type="Percent"  minOccurs="0"/>
         <xs:element name="lotDetail" type="LotInformation" />
         <xs:element name="prices" type="PriceSchedule"  minOccurs="0"/>
         <xs:element name="generalStatus" type="itis:ITIScodesAndText"  minOccurs="0"/>
         <xs:element name="furtherData" type="URL-Link"  minOccurs="0"/>
         <xs:element name="tail" type="Tail"  minOccurs="0"/>
         <xs:element       name="localParkingLotInformation"       type="local:ParkingLotInformation"
minOccurs="0"/>
      </xs:sequence>
</xs:complexType>
```

Used by: This item is used by the one other data structures in this standard, a Data Frame called DF_ResponseGroup [ResponseGroup] <ResponseGroup> . In addition, this item may be used by data structures in other ITS standards.

Remarks: Originally from J2354, Section 5.5 Parking Messages.

Data Frame: DF_Lot Information

Use: A detailed summary of the status of the lot including the number of spaces, when they are expected to fill, and the hours of operation for the lot. The spaces element will provide a more detailed summary of the count of each type of parking space.

ASN.1 Representation:

```
LotInformation ::= SEQUENCE {
     spacesTotal          TCIP.PI-ParkingSpacesTotal      OPTIONAL,
     availability         TCIP.PI-ParkingAvailability     OPTIONAL,
     spaces               ParkingSpaceTypes               OPTIONAL,
     lot-Name             Lot-Name                        OPTIONAL,
     hoursofOperation     TimePair                        OPTIONAL,
     rates                TCIP.PI-ParkingRates            OPTIONAL,
     parkingFillTime      DateTimePair                    OPTIONAL,
     ...
     }
```

XML Representation:

```
<xs:complexType name="LotInformation" >
      <xs:sequence>
         <xs:element name="spacesTotal" type="tcip:PI-ParkingSpacesTotal"  minOccurs="0"/>
         <xs:element name="availability" type="tcip:PI-ParkingAvailability"  minOccurs="0"/>
         <xs:element name="spaces" type="ParkingSpaceTypes"  minOccurs="0"/>
         <xs:element name="lot-Name" type="Lot-Name"  minOccurs="0"/>
         <xs:element name="hoursofOperation" type="TimePair"  minOccurs="0"/>
         <xs:element name="rates" type="tcip:PI-ParkingRates"  minOccurs="0"/>
         <xs:element name="parkingFillTime" type="DateTimePair"  minOccurs="0"/>
      </xs:sequence>
</xs:complexType>
```

Used by: This item is used by the one other data structures in this standard, a Data Frame called DF_ParkingLotInformation [ParkingLotInformation] <ParkingLotInformation> . In addition, this item may be used by data structures in other ITS standards.

Remarks: Originally from J2354, Section 5.5 Parking Messages. The PI-ParkingVehicleClass is used only as an overall summary of the type of lot this is.

Data Frame: DF_ParkingSpaceTypes

Use: A set of values defining a type of parking space and how many such spaces exist.

ASN.1 Representation:

```
ParkingSpaceTypes ::= SEQUENCE {
     type         ParkingSpaceKind,
     spaces       TCIP.PI-ParkingSpacesTotal,
     vehicleClass TCIP.PI-ParkingVehicleClass OPTIONAL,
     ...
     }
```

XML Representation:

```
<xs:complexType name="ParkingSpaceTypes" >
      <xs:sequence>
         <xs:element name="type" type="ParkingSpaceKind" />
         <xs:element name="spaces" type="tcip:PI-ParkingSpacesTotal" />
         <xs:element name="vehicleClass" type="tcip:PI-ParkingVehicleClass"  minOccurs="0"/>
      </xs:sequence>
</xs:complexType>
```

Used by: This item is used by the one other data structures in this standard, a Data Frame called DF_LotInformation [LotInformation] <LotInformation> . In addition, this item may be used by data structures in other ITS standards.

Data Element: DE_ParkingSpaceKind

Use: The types of parking spaces. Often used in conjunction with a count of current spaces.

ASN.1 Representation:

```
ParkingSpaceKind ::= ENUMERATED {
   unknown           (0),  -- use for general types when none is known
   auto              (1),  -- regular sized or typical spaces
   compact           (2),
   large             (3),
```

```
   handicapped       (4),  -- and all disabled types
   curbside          (5),
   carpool           (6),
   carpoolHours      (7), -- reserved for carpoolers during certain hours
   midDay            (8), -- reserved for mid-day users (may be certain hours)
   electricVehicles  (9), -- electric vehicles (adjacent to charging station)
   motorcycle        (10),
   bicycle           (11),  -- bicycle spaces/lockers/racks
   loadingOnly       (12),
   reserved          (13),
   private           (14),
   valet             (15),
   shortTermOnly     (16), -- use for 15 min parking, atm parking etc...
   truck             (17),
   carAndTrailer     (18),
   semiTrailer       (19),
   bus               (20),
   outside           (21), -- commonly used for parking where
                           -- oversized vehicles can not proceed
   officialVehicles  (22),
   cityCarShare      (23),
   ...               -- # LOCAL_CONTENT
   }
   -- values to 127 reserved for std use
   -- values 128 to 255 reserved for local use
```

XML Representation:

```
<xs:simpleType name="ParkingSpaceKind" >
      <xs:annotation>
         <xs:appinfo>
            unknown (0) -- use for general types when none is known
            auto (1) -- regular sized or typical spaces
            compact (2)
            large (3)
            handicapped (4) -- and all disabled types
            curbside (5)
            carpool (6)
            carpoolHours (7) -- reserved for carpoolers during certain hours
            midDay (8) -- reserved for mid-day users  (may be certain hours)
            electricVehicles (9) -- electric vehicles  (adjacent to charging station)
            motorcycle (10)
            bicycle (11) -- bicycle spaces/lockers/racks
            loadingOnly (12)
            reserved (13)
            private (14)
            valet (15)
            shortTermOnly (16) -- use for 15 min parking ,
            truck (17)
            carAndTrailer (18)
            semiTrailer (19)
            bus (20)
            outside (21) -- commonly used for parking where
             -- oversized vehicles can not proceed
            officialVehicles (22)
            cityCarShare (23) -- LOCAL_CONTENT
         </xs:appinfo>
         <xs:documentation>
            values to 127 reserved for std use
            values 128 to 255 reserved for local use
         </xs:documentation>
      </xs:annotation>
      <xs:union>
         <xs:simpleType>
            <xs:restriction base="xs:unsignedInt">
               <xs:minInclusive value="0"/>
               <xs:maxInclusive value="23"/>
            </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="unknown"/>
               <xs:enumeration value="auto"/>
               <xs:enumeration value="compact"/>
               <xs:enumeration value="large"/>
               <xs:enumeration value="handicapped"/>
```

```
            <xs:enumeration value="curbside"/>
            <xs:enumeration value="carpool"/>
            <xs:enumeration value="carpoolHours"/>
            <xs:enumeration value="midDay"/>
            <xs:enumeration value="electricVehicles"/>
            <xs:enumeration value="motorcycle"/>
            <xs:enumeration value="bicycle"/>
            <xs:enumeration value="loadingOnly"/>
            <xs:enumeration value="reserved"/>
            <xs:enumeration value="private"/>
            <xs:enumeration value="valet"/>
            <xs:enumeration value="shortTermOnly"/>
            <xs:enumeration value="truck"/>
            <xs:enumeration value="carAndTrailer"/>
            <xs:enumeration value="semiTrailer"/>
            <xs:enumeration value="bus"/>
            <xs:enumeration value="outside"/>
            <xs:enumeration value="officialVehicles"/>
            <xs:enumeration value="cityCarShare"/>
         </xs:restriction>
      </xs:simpleType >
   </xs:union>
</xs:simpleType>
```

Used by: This item is used by the one other data structures in this standard, a Data Frame called DF_ParkingSpaceTypes [ParkingSpaceTypes] <ParkingSpaceTypes> . In addition, this item may be used by data structures in other ITS standards.

Message: MSG_ParkingSpaceReply

Use: A message to contain information about a parking space.

ASN.1 Representation:

```
ParkingSpaceReply ::= SEQUENCE {
     messageHeader     MessageHeader,
     parkingFacID      TCIP.PI-ParkingFacID,  -- the lot in queston
     parkingRates      TCIP.PI-ParkingRates,  -- the rates for this lot
     confirm           BOOLEAN,
     ...
     }
```

XML Representation:

```
<xs:element name="parkingSpaceReply" type="ParkingSpaceReply"/>
<xs:complexType name="ParkingSpaceReply" >
      <xs:sequence>
         <xs:element name="messageHeader" type="MessageHeader" />
         <xs:element name="parkingFacID" type="tcip:PI-ParkingFacID" />
         <!-- the lot in queston -->
         <xs:element name="parkingRates" type="tcip:PI-ParkingRates" />
         <!-- the rates for this lot -->
         <xs:element name="confirm" >
            <xs:simpleType>
               <xs:restriction base="xs:boolean"/>
            </xs:simpleType>
         </xs:element>
      </xs:sequence>
</xs:complexType>
```

Remarks: Originally from J2354, Section 5.5 Parking Messages.

Message: MSG_ParkingSpaceRequest

Use: send to request a parking space from a specific lot.

ASN.1 Representation:

```
ParkingSpaceRequest ::= SEQUENCE {
     messageHeader          MessageHeader,
     traveler-Identity      Traveler-Identity,
     vehicleData            VehicleRestrictions OPTIONAL,
     parkingFacID           TCIP.PI-ParkingFacID,
     duration               TimeInterval OPTIONAL,
     requestDateTime        DateTimePair OPTIONAL,
```

```
    ...
    }
```

XML Representation:

```
<xs:element name="parkingSpaceRequest" type="ParkingSpaceRequest"/>
<xs:complexType name="ParkingSpaceRequest" >
      <xs:sequence>
         <xs:element name="messageHeader" type="MessageHeader" />
         <xs:element name="traveler-Identity" type="Traveler-Identity" />
         <xs:element name="vehicleData" type="VehicleRestrictions"  minOccurs="0"/>
         <xs:element name="parkingFacID" type="tcip:PI-ParkingFacID" />
         <xs:element name="duration" type="TimeInterval"  minOccurs="0"/>
         <xs:element name="requestDateTime" type="DateTimePair"  minOccurs="0"/>
      </xs:sequence>
</xs:complexType>
```

Remarks: Originally from J2354, Section 5.5 Parking Messages.

Data Element: EXT_PI_ParkingAvailability

Use: External data element taken from TCIP. The real-time availability of parking spaces at a particular parking facility.

ASN.1 Representation:

```
PI-ParkingAvailability ::= INTEGER
```

XML Representation:

```
<xs:simpleType name="PI-ParkingAvailability" >
      <xs:restriction base="xs:int"/>
</xs:simpleType>
```

Used by: This item is used by the one other data structures in this standard, a Data Frame called DF_LotInformation [LotInformation] <LotInformation> . In addition, this item may be used by data structures in other ITS standards.

Data Element: EXT_PI_ParkingFacID

Use: External data element taken from TCIP. A unique identification number for a parking facility.

ASN.1 Representation:

```
PI-ParkingFacID ::= INTEGER (0..65535)
```

XML Representation:

```
<xs:simpleType name="PI-ParkingFacID" >
      <xs:restriction base="xs:unsignedShort"/>
</xs:simpleType>
```

Used by: This item is used by the following 3 other data structures in this standard (record type, descriptive name, ASN.1, and XML name of each):

Data Frame DF_Parking-Instructions [ParkingInstructions] <ParkingInstructions>, and

Message MSG_ParkingSpaceReply [ParkingSpaceReply] <ParkingSpaceReply>, and

Message MSG_ParkingSpaceRequest [ParkingSpaceRequest] <ParkingSpaceRequest>.

In addition, this item may be used by data structures in other ITS standards.

Data Element: EXT_PI_ParkingProvided

Use: External data element taken from TCIP. An identifier that tells whether or not parking is available at a transit stop.

ASN.1 Representation:

```
PI-ParkingProvided ::= BOOLEAN
```

XML Representation:

```
<xs:simpleType name="PI-ParkingProvided" >
      <xs:restriction base="xs:boolean"/>
</xs:simpleType>
```

Data Element: EXT_PI_ParkingRates

Use: External data element taken from TCIP. The amount of money required to park at a parking facility. This can be expressed hourly, by the day, etc. This text field allows for a short explanation of the way parking cost is calculated.

ASN.1 Representation:

```
PI-ParkingRates ::= UTF8String (SIZE(0..1000))
```

XML Representation:

```
<xs:simpleType name="PI-ParkingRates" >
      <xs:restriction base="xs:string">
         <xs:minLength value="0"/>
         <xs:maxLength value="1000"/>
      </xs:restriction>
</xs:simpleType>
```

Used by: This item is used by the following 2 other data structures in this standard (record type, descriptive name, ASN.1, and XML name of each):

Data Frame DF_LotInformation [LotInformation] <LotInformation>, and

Message MSG_ParkingSpaceReply [ParkingSpaceReply] <ParkingSpaceReply>.

In addition, this item may be used by data structures in other ITS standards.

Remarks: A size limit was added for ATIS use.

Data Element: EXT_PI_ParkingSpacesTotal

Use: External data element taken from TCIP. The total number of parking spaces in a parking facility.

ASN.1 Representation:

```
PI-ParkingSpacesTotal ::= INTEGER (0..65535)
```

XML Representation:

```
<xs:simpleType name="PI-ParkingSpacesTotal" >
      <xs:restriction base="xs:unsignedShort"/>
</xs:simpleType>
```

Used by: This item is used by the following 2 other data structures in this standard (record type, descriptive name, ASN.1, and XML name of each):

Data Frame DF_LotInformation [LotInformation] <LotInformation>, and

Data Frame DF_ParkingSpaceTypes [ParkingSpaceTypes] <ParkingSpaceTypes>.

In addition, this item may be used by data structures in other ITS standards.

Data Element: EXT_PI_ParkingType

Use: External data element taken from TCIP. The physical characteristics of a parking facility.

ASN.1 Representation:

```
PI-ParkingType ::= INTEGER
```

XML Representation:

```
<xs:simpleType name="PI-ParkingType" >
      <xs:restriction base="xs:int"/>
</xs:simpleType>
```

Used by: This item is used by the following 2 other data structures in this standard (record type, descriptive name, ASN.1, and XML name of each):

Data Frame DF_ParkingLotInformation [ParkingLotInformation] <ParkingLotInformation>, and

Data Frame DF_ParkingRequestDetails [ParkingRequestDetails] <ParkingRequestDetails>.

In addition, this item may be used by data structures in other ITS standards.

Remarks: Consult TCIP for suitable codes.

Data Element: EXT_PI_ParkingVehicleClass

Use: External data element taken from TCIP. The vehicle class applicable to parking facility concerns.

ASN.1 Representation:

```
PI-ParkingVehicleClass ::= INTEGER
```

XML Representation:

```
<xs:simpleType name="PI-ParkingVehicleClass" >
      <xs:restriction base="xs:int"/>
</xs:simpleType>
```

Used by: This item is used by the one other data structures in this standard, a Data Frame called DF_ParkingSpaceTypes [ParkingSpaceTypes] <ParkingSpaceTypes> . In addition, this item may be used by data structures in other ITS standards.

Remarks: Consult TCIP for suitable codes.

Data Frame: DF_DateTimePair

Use: A data and time value. In addition, the timeoffset element can be used to convey the local time zone. This is required when merging data sources over multiple time zones.

ASN.1 Representation:

```
DateTimePair ::= SEQUENCE {
     date    Date,
     time    Time,
     offset  TimeOffset OPTIONAL
     }
```

XML Representation:

```
<xs:complexType name="DateTimePair" >
      <xs:sequence>
         <xs:element name="date" type="Date" />
         <xs:element name="time" type="Time" />
         <xs:element name="offset" type="TimeOffset"  minOccurs="0"/>
      </xs:sequence>
</xs:complexType>
```

Example
"2010-01-10T14:30:42.0000-08:00"

Data Frame: DF_PriceSchedule

Use: A data frame to relate price and cost information.

ASN.1 Representation:

```
PriceSchedule ::= SEQUENCE {
     start           DateTimePair                    OPTIONAL,
     end             DateTimePair                    OPTIONAL,
     entries         SEQUENCE  (SIZE(1..100)) OF PriceScheduleEntry
}
```

XML Representation:

```
<xs:complexType name="PriceSchedule" >
      <xs:sequence>
         <xs:element name="start" type="DateTimePair"  minOccurs="0"/>
         <xs:element name="end" type="DateTimePair"  minOccurs="0"/>
         <xs:element name="entries" >
            <xs:complexType>
               <xs:sequence minOccurs="1" maxOccurs="100">
                  <xs:element name="entry" type="PriceScheduleEntry" />
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
</xs:complexType>
```

Used by: This item is used by the following 2 other data structures in this standard (record type, descriptive name, ASN.1, and XML name of each):

Data Frame DF_Parking-Instructions [ParkingInstructions] <ParkingInstructions>, and

Data Frame DF_ParkingLotInformation [ParkingLotInformation] <ParkingLotInformation>.

In addition, this item may be used by data structures in other ITS standards.

Remarks: Originally from J2354, Section 5.1 Global Messages.

Data Frame: DF_PriceScheduleEntry

Use: A summary of the prices for a lot.

ASN.1 Representation:

```
PriceScheduleEntry ::= SEQUENCE {
     price-DayType            Price-DayType,
     time                     TimePair OPTIONAL,
                              -- a range of times over which the next
                              -- two items are vaild
     priceFirstPayment        Cost OPTIONAL,
                              -- the cost of entry the first time, or min cost
     priceMaximum             Cost OPTIONAL,
                              -- typically the daily max price
     times                    SEQUENCE  (SIZE(1..100)) OF TimePriceInterval OPTIONAL,
                              -- used to convey prices vs time data such as
                              -- discounted parking if entry before 7 AM
     ...
     }
```

XML Representation:

```
<xs:complexType name="PriceScheduleEntry" >
      <xs:sequence>
         <xs:element name="price-DayType" type="Price-DayType" />
         <xs:element name="time" type="TimePair"  minOccurs="0"/>
         <!-- a range of times over which the next
         two items are vaild -->
         <xs:element name="priceFirstPayment" type="Cost"  minOccurs="0"/>
         <!-- the cost of entry the first time,  or min cost -->
         <xs:element name="priceMaximum" type="Cost"  minOccurs="0"/>
         <!-- typically the daily max price -->
         <xs:element name="times"  minOccurs="0">
            <xs:complexType>
               <xs:sequence minOccurs="1" maxOccurs="100">
                  <xs:element name="time" type="TimePriceInterval" />
                  <!-- used to convey prices vs time data such as discounted parking if entry before
7 AM -->
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
</xs:complexType>
```

Used by: This item is used by the one other data structures in this standard, a Data Frame called DF_PriceSchedule [PriceSchedule] <PriceSchedule> . In addition, this item may be used by data structures in other ITS standards.

Remarks: Originally from J2354, Section 5.5 Parking Messages. The priceFirstPayment is the first payment where it differs from succeeding payments or the fixed price of a stay.

Data Element: DE_PRICE_DayType_code

Use: Used as part of Sending a Price Schedule - The Days of Week to which the following information pertains - also includes designator "Hol" for Holiday.

ASN.1 Representation:

```
Price-DayType ::= ENUMERATED {
     holiday                          (0),
     sunday                           (1),
     monday                           (2),
     tuesday                          (3),
     wednesday                        (4),
     thursday                         (5),
     friday                           (6),
     saturday                         (7),
     all                              (8),
     mondayToFriday                   (9),
     mondayToFridayExceptHolidays     (10),
     weekend                          (11),
     mondayToThursday                 (12),
     unknown                         (255),
     ... -- # LOCAL_CONTENT
     }
     -- values to 127 reserved for std use
     -- values 128 to 254 reserved for local use
```

XML Representation:

```
<xs:simpleType name="Price-DayType" >
      <xs:annotation>
         <xs:appinfo>
            holiday (0)
            sunday (1)
            monday (2)
            tuesday (3)
            wednesday (4)
            thursday (5)
            friday (6)
            saturday (7)
            all (8)
            mondayToFriday (9)
            mondayToFridayExceptHolidays (10)
            weekend (11)
            mondayToThursday (12)
            unknown (255) -- LOCAL_CONTENT
         </xs:appinfo>
         <xs:documentation>
            values to 127 reserved for std use
            values 128 to 254 reserved for local use
         </xs:documentation>
      </xs:annotation>
      <xs:union>
         <xs:simpleType>
            <xs:restriction base="xs:unsignedInt">
               <xs:minInclusive value="0"/>
               <xs:maxInclusive value="255"/>
            </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="holiday"/>
               <xs:enumeration value="sunday"/>
               <xs:enumeration value="monday"/>
               <xs:enumeration value="tuesday"/>
               <xs:enumeration value="wednesday"/>
```

```
            <xs:enumeration value="thursday"/>
            <xs:enumeration value="friday"/>
            <xs:enumeration value="saturday"/>
            <xs:enumeration value="all"/>
            <xs:enumeration value="mondayToFriday"/>
            <xs:enumeration value="mondayToFridayExceptHolidays"/>
            <xs:enumeration value="weekend"/>
            <xs:enumeration value="mondayToThursday"/>
            <xs:enumeration value="unknown"/>
         </xs:restriction>
      </xs:simpleType >
   </xs:union>
</xs:simpleType>
```

Used by: This item is used by the one other data structures in this standard, a Data Frame called DF_PriceScheduleEntry [PriceScheduleEntry] <PriceScheduleEntry> . In addition, this item may be used by data structures in other ITS standards.

XML Representation:

```
<xs:complexType name="IdentityOrLocation" >
   <xs:choice>
      <xs:element name="location" type="lrms:LocationReference" />
      <xs:element name="dbIdentity" type="DatabaseIdentity" />
   </xs:choice>
```

</xs:complexType

XML Representation:

```
<xs:complexType name="LocationReference" >
   <xs:annotation>
      <xs:documentation>
         the ModelProfile has been removed
      </xs:documentation>
   </xs:annotation>
   <xs:choice>
      <xs:element name="geometry" type="GeometryProfile" />
      <xs:element name="geocoordinate" type="GeographicCoordinate" />
      <xs:element name="grid" type="GridProfile" />
      <xs:element name="linearReferencing" type="LinearReferenceProfile" />
      <xs:element name="crossStreets" type="ISP-Vehicle" />
      <!-- was: CrossStreetsProfile -->
      <xs:element name="address" type="AddressProfile" />
      <xs:element name="pubgrid" type="PublicGridProfile" />
   </xs:choice>
</xs:complexType>
```

XML Representation:

```
<xs:complexType name="DatabaseIdentity" >
   <xs:sequence>
      <xs:element name="identity"  minOccurs="0">
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:minLength value="1"/>
               <xs:maxLength value="10"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:element>
      <xs:element name="number" type="DatabaseNumber"  minOccurs="0"/>
      <xs:element name="name" type="Database"  minOccurs="0"/>
      <xs:element name="version" type="DatabaseVersion"  minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

Appendix C. XML Formats For UPP Parking

The schema defined for this project should be used by the following entities:

1. Parking Vendors
2. The Parking Vendor Interface (java application)
3. The Master Database Interface (java application)
4. The Multi Modal Trip Planner.

This schema contains a header that has source, destination, request/response, and parking vendor inventory information (see UPPParking.xsd).

From Parking Vendors to PVI

There are two formats:

1. From Vendor sending parking facility configuration static format.
2. From Vendor sending parking facility dynamic. This data is sent by the vendor to update the availability and pricing.

Below is an example of the static format (not all optional fields are shown):


```
<  <?xml version="1.0" encoding="UTF-8" ?>
- <n:park xmlns:n="http://www.dummy-UPPParking-address">
  <timestamp>2010-11-30T23:59:59</timestamp>
- <header>
  <type>response</type>
  <source>VENDOR</source>
  <dest>PVI</dest>
  <action />
  </header>
  <success>true</success>
  <responsetype>vendorinventorystatic</responsetype>
- <vendorinventorystatic>
  <vendorid>02030</vendorid>
  <vendorname>sf park</vendorname>
- <facilityinfo>
  <parkingFacID>48284</parkingFacID>
- <location>
- <grid>
- <gridArray>
- <gridArray-item>
  <id>253</id>
  <latitude>3930</latitude>
  <longitude>-7695</longitude>
  <zoom>2</zoom>
  </gridArray-item>
- <gridArray-item>
```

<id>247</id>
<latitude>3930</latitude>
<longitude>-7695</longitude>
<zoom>2</zoom>
</gridArray-item>
- <gridArray-item>
<id>246</id>
<latitude>3930</latitude>
<longitude>-7695</longitude>
<zoom>2</zoom>
</gridArray-item>
- <gridArray-item>
<id>255</id>
<latitude>3935</latitude>
<longitude>-7691</longitude>
<zoom>2</zoom>
</gridArray-item>
- <gridArray-item>
<id>254</id>
<latitude>3830</latitude>
<longitude>-7691</longitude>
<zoom>2</zoom>
</gridArray-item>
</gridArray>
</grid>
- <crossStreet>
- <crossStreetsPoint>
- <onStreetInfo>
<name>main street</name>
</onStreetInfo>
- <atStreetInfo>
<name>maple st</name>
</atStreetInfo>
- <geoLocation>
<latitude>3804</latitude>
<longitude>-7702</longitude>
</geoLocation>
</crossStreetsPoint>
</crossStreet>
</location>
<parkingType>2</parkingType>
- <lotDetail>
<city>San Francisco</city>
<neighborhood>The Haight</neighborhood>
<spacesTotal>106</spacesTotal>
<lot-Name>very best parking inc</lot-Name>
- <hoursofOperation>
- <weekly>
- <weekly-item>
<day>monday</day>
- <times>
<start>093000</start>
<end>163001</end>

```
  </times>
  </weekly-item>
- <weekly-item>
  <day>tuesday</day>
- <times>
  <start>103000</start>
  <end>173030</end>
  </times>
  </weekly-item>
  </weekly>
  </hoursofOperation>
- <parkingFillTime>
  <date>20100105</date>
  <time>120305</time>
  </parkingFillTime>
- <entrance>
  <entranceHeight>220</entranceHeight>
- <entranceLocation>
  <name>122 West 57th St</name>
  </entranceLocation>
  </entrance>
- <entrance>
  <entranceHeight>243</entranceHeight>
- <entranceLocation>
  <name>122 East 57th St</name>
  </entranceLocation>
  </entrance>
  <paymentOptions>personal check</paymentOptions>
  <maxTimeLimit>600</maxTimeLimit>
  </lotDetail>
- <moreDetail>
  <ownershipAgencyType>public</ownershipAgencyType>
  <structureType>garage</structureType>
  <peakUsageTimes>afternoon peak</peakUsageTimes>
  <onsiteSecurityInfo>yes</onsiteSecurityInfo>
  <contactphone2>202 555 7184</contactphone2>
- <numberofspaces>
  <structureType>electric car spaces</structureType>
  <numspaces>233</numspaces>
  </numberofspaces>
- <numberofspaces>
  <structureType>motorcycle spaces</structureType>
  <numspaces>3</numspaces>
  </numberofspaces>
- <numberofspaces>
  <structureType>standard spaces</structureType>
  <numspaces>203</numspaces>
  </numberofspaces>
- <transitagency>
  <agency>transit agency 1</agency>
  <agency>transit agency two</agency>
  <agency>transit agency three</agency>
  </transitagency>
```

<realtimeinfoavailable>no</realtimeinfoavailable>
<haschargingport>no</haschargingport>
- <parkingaccess>
<name>main stairwell</name>
<accesstype>entrance</accesstype>
<accessfeature>stair</accessfeature>
<addlInfo>additional info is here</addlInfo>
</parkingaccess>
- <parkingaccess>
<name>main stairwell</name>
<accesstype>exit</accesstype>
<accessfeature>stair</accessfeature>
</parkingaccess>
- <paymentdiscountAAA>
<usDollar>3.98</usDollar>
</paymentdiscountAAA>
<paycashieratexit>yes</paycashieratexit>
<servicetypes>valet</servicetypes>
<servicetypes>security</servicetypes>
<reservMethod>at desk in lobby</reservMethod>
<lockersize>large</lockersize>
- <lockercost>
<usDollar>3.25</usDollar>
</lockercost>
</moreDetail>
- <prices>
- <start>
<date>20100429</date>
<time>093000</time>
</start>
- <end>
<date>20100430</date>
<time>173001</time>
</end>
- <entries>
- <entry>
<price-DayType>monday</price-DayType>
- <spacetype>
<type>compact</type>
</spacetype>
- <time>
<start>083000</start>
<end>221000</end>
</time>
- <priceFirstPayment>
<usDollar>25.59</usDollar>
</priceFirstPayment>
- <priceMaximum>
<usDollar>35.99</usDollar>
</priceMaximum>
</entry>
- <entry>
<price-DayType>holiday</price-DayType>

- <spacetype>
<type>compact</type>
</spacetype>
- <times>
- <time>
<timeInterval>25</timeInterval>
- <cost>
<usDollar>27.00</usDollar>
</cost>
</time>
</times>
- <times>
- <time>
<timeInterval>25</timeInterval>
- <cost>
<usDollar>3.50</usDollar>
</cost>
</time>
</times>
- <times>
- <time>
<timeInterval>251</timeInterval>
- <cost>
<usDollar>3.51</usDollar>
</cost>
</time>
</times>
</entry>
</entries>
</prices>
</facilityinfo>
- <facilityinfo>
<parkingFacID>48386</parkingFacID>
- <location>
- <grid>
- <gridArray>
- <gridArray-item>
<id>253</id>
<latitude>3930</latitude>
<longitude>-7695</longitude>
<zoom>2</zoom>
</gridArray-item>
- <gridArray-item>
<id>247</id>
<latitude>3930</latitude>
<longitude>-7695</longitude>
<zoom>2</zoom>
</gridArray-item>
- <gridArray-item>
<id>246</id>
<latitude>3930</latitude>
<longitude>-7695</longitude>
<zoom>2</zoom>

</gridArray-item>
- <gridArray-item>
<id>255</id>
<latitude>3935</latitude>
<longitude>-7691</longitude>
<zoom>2</zoom>
</gridArray-item>
- <gridArray-item>
<id>254</id>
<latitude>3830</latitude>
<longitude>-7691</longitude>
<zoom>2</zoom>
</gridArray-item>
</gridArray>
</grid>
- <crossStreet>
- <crossStreetsLink>
- <onStreetInfo>
<name>main street</name>
</onStreetInfo>
- <fromStreetInfo>
<name>eggmont st</name>
</fromStreetInfo>
- <toStreetInfo>
<name>crownshield way</name>
</toStreetInfo>
- <startGeoLocation>
<latitude>3804</latitude>
<longitude>-7702</longitude>
</startGeoLocation>
- <endGeoLocation>
<latitude>3845</latitude>
<longitude>-7704</longitude>
</endGeoLocation>
</crossStreetsLink>
</crossStreet>
</location>
<parkingType>2</parkingType>
- <lotDetail>
<city>nice city name</city>
<spacesTotal>105</spacesTotal>
- <hoursofOperation>
- <weekly>
- <weekly-item>
<day>monday</day>
- <times>
<start>093000</start>
<end>163000</end>
</times>
</weekly-item>
- <weekly-item>
<day>tuesday</day>
- <times>

<start>103000</start>
<end>173030</end>
</times>
</weekly-item>
</weekly>
</hoursofOperation>
- <entrance>
<entranceHeight>220</entranceHeight>
- <entranceLocation>
<name>122 West 57th St</name>
</entranceLocation>
</entrance>
- <entrance>
<entranceHeight>243</entranceHeight>
- <entranceLocation>
<name>122 East 57th St</name>
</entranceLocation>
</entrance>
<paymentOptions>personal check</paymentOptions>
<maxTimeLimit>600</maxTimeLimit>
</lotDetail>
- <moreDetail>
<ownershipAgencyType>public</ownershipAgencyType>
<structureType>garage</structureType>
- <numberofspaces>
<structureType>electric car spaces</structureType>
<numspaces>13</numspaces>
</numberofspaces>
- <numberofspaces>
<structureType>motorcycle spaces</structureType>
<numspaces>3</numspaces>
</numberofspaces>
<haschargingport>yes</haschargingport>
<servicetypes>valet</servicetypes>
<servicetypes>security</servicetypes>
- <associatedfiles>
<filename>file3.dat</filename>
<base64data />
</associatedfiles>
- <associatedfiles>
<filename>fpfile.dat</filename>
<base64data />
</associatedfiles>
</moreDetail>
- <prices>
<priority>1</priority>
- <start>
<date>20100425</date>
<time>113000</time>
</start>
- <end>
<date>20100425</date>
<time>143000</time>

```
  </end>
- <entries>
- <entry>
  <price-DayType>monday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>30</timeInterval>
- <cost>
  <usDollar>4.50</usDollar>
  </cost>
  </time>
  <rateDescription>$4.50 per half hour</rateDescription>
  </times>
- <times>
- <time>
  <timeInterval>60</timeInterval>
- <cost>
  <usDollar>8.50</usDollar>
  </cost>
  </time>
  <rateDescription>$8.50 per hour</rateDescription>
  </times>
- <times>
- <time>
  <timeInterval>120</timeInterval>
- <cost>
  <usDollar>16.00</usDollar>
  </cost>
  </time>
  <rateDescription>$16.00 for two hours</rateDescription>
  </times>
  </entry>
- <entry>
  <price-DayType>holiday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>26</timeInterval>
- <cost>
  <usDollar>427.00</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>45</timeInterval>
- <cost>
  <usDollar>4.50</usDollar>
```

```
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>251</timeInterval>
- <cost>
  <usDollar>4.51</usDollar>
  </cost>
  </time>
  </times>
  </entry>
  </entries>
  </prices>
- <prices>
  <priority>2</priority>
- <entries>
- <entry>
  <price-DayType>monday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>26</timeInterval>
- <cost>
  <usDollar>45.59</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>42</timeInterval>
- <cost>
  <usDollar>4.20</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>134</timeInterval>
- <cost>
  <usDollar>4.51</usDollar>
  </cost>
  </time>
  </times>
  </entry>
- <entry>
  <price-DayType>holiday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
```

- <time>
<timeInterval>26</timeInterval>
- <cost>
<usDollar>427.00</usDollar>
</cost>
</time>
</times>
- <times>
- <time>
<timeInterval>45</timeInterval>
- <cost>
<usDollar>4.50</usDollar>
</cost>
</time>
</times>
- <times>
- <time>
<timeInterval>251</timeInterval>
- <cost>
<usDollar>4.51</usDollar>
</cost>
</time>
</times>
</entry>
</entries>
</prices>
- <prices>
- <entries>
- <entry>
<price-DayType>monday</price-DayType>
- <spacetype>
<type>compact</type>
</spacetype>
- <times>
- <time>
<timeInterval>26</timeInterval>
- <cost>
<usDollar>45.59</usDollar>
</cost>
</time>
</times>
- <times>
- <time>
<timeInterval>42</timeInterval>
- <cost>
<usDollar>4.20</usDollar>
</cost>
</time>
</times>
- <times>
- <time>
<timeInterval>134</timeInterval>
- <cost>

```
  <usDollar>4.51</usDollar>
  </cost>
  </time>
  </times>
  </entry>
- <entry>
  <price-DayType>holiday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>26</timeInterval>
- <cost>
  <usDollar>427.00</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>45</timeInterval>
- <cost>
  <usDollar>4.50</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>251</timeInterval>
- <cost>
  <usDollar>4.51</usDollar>
  </cost>
  </time>
  </times>
  </entry>
  </entries>
  </prices>
  </facilityinfo>
  </vendorinventorystatic>
  </n:park>
```


For LocationReference, only the GridProfile and CrossStreetsProfile will be supported.

The GridProfile offers a GridArray which allows up to ten points. Each point consists of a latitude, longitude, ID, and zoom parameter.

Below is an example of the dynamic format:


```
   <?xml version="1.0" encoding="UTF-8" ?>
- <n:park xmlns:n="http://www.dummy-UPPParking-address">
  <timestamp>2010-11-30T23:59:59</timestamp>
- <header>
  <type />
```

```
  <source>VENDOR</source>
  <dest>PVI</dest>
  <action />
  </header>
- <vendorinventorydynamic>
  <vendorid>02030</vendorid>
  <vendorname>bestparking inc</vendorname>
- <facilityinfo>
  <parkingFacID>2489084</parkingFacID>
  <percentFull>35</percentFull>
  <spacesTotal>105</spacesTotal>
  <availability>24</availability>
  <occupied>81</occupied>
- <prices>
  <priority>1</priority>
- <start>
  <date>20100425</date>
  <time>113000</time>
  </start>
- <end>
  <date>20100425</date>
  <time>143000</time>
  </end>
- <entries>
- <entry>
  <price-DayType>monday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>30</timeInterval>
- <cost>
  <usDollar>4.50</usDollar>
  </cost>
  </time>
  <rateDescription>$4.50 per half hour</rateDescription>
  </times>
- <times>
- <time>
  <timeInterval>60</timeInterval>
- <cost>
  <usDollar>8.50</usDollar>
  </cost>
  </time>
  <rateDescription>$8.50 per hour</rateDescription>
  </times>
- <times>
- <time>
  <timeInterval>120</timeInterval>
- <cost>
  <usDollar>16.00</usDollar>
  </cost>
  </time>
  <rateDescription>$16.00 for two hours</rateDescription>
  </times>
  </entry>
- <entry>
  <price-DayType>holiday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>26</timeInterval>
- <cost>
  <usDollar>427.00</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>45</timeInterval>
- <cost>
  <usDollar>4.50</usDollar>
```

```
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>251</timeInterval>
- <cost>
  <usDollar>4.51</usDollar>
  </cost>
  </time>
  </times>
  </entry>
  </entries>
  </prices>
- <prices>
  <priority>2</priority>
- <entries>
- <entry>
  <price-DayType>monday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>26</timeInterval>
- <cost>
  <usDollar>45.59</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>42</timeInterval>
- <cost>
  <usDollar>4.20</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>134</timeInterval>
- <cost>
  <usDollar>4.51</usDollar>
  </cost>
  </time>
  </times>
  </entry>
- <entry>
  <price-DayType>holiday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>26</timeInterval>
- <cost>
  <usDollar>427.00</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>45</timeInterval>
- <cost>
  <usDollar>4.50</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>251</timeInterval>
- <cost>
  <usDollar>4.51</usDollar>
  </cost>
  </time>
```

```
  </times>
  </entry>
  </entries>
  </prices>
- <prices>
- <entries>
- <entry>
  <price-DayType>monday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>26</timeInterval>
- <cost>
  <usDollar>45.59</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>42</timeInterval>
- <cost>
  <usDollar>4.20</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>134</timeInterval>
- <cost>
  <usDollar>4.51</usDollar>
  </cost>
  </time>
  </times>
  </entry>
- <entry>
  <price-DayType>holiday</price-DayType>
- <spacetype>
  <type>compact</type>
  </spacetype>
- <times>
- <time>
  <timeInterval>26</timeInterval>
- <cost>
  <usDollar>427.00</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>45</timeInterval>
- <cost>
  <usDollar>4.50</usDollar>
  </cost>
  </time>
  </times>
- <times>
- <time>
  <timeInterval>251</timeInterval>
- <cost>
  <usDollar>4.51</usDollar>
  </cost>
  </time>
  </times>
  </entry>
  </entries>
- <discount>
  <discountpct>20.0</discountpct>
  <reason>senior</reason>
  </discount>
  </prices>
  </facilityinfo>
  </vendorinventorydynamic>
  </n:park>
```

```
  >
>
  <
```

Requests and Responses

The PVI will issue requests either to parking vendors or to the MDBI. Neither the parking vendors nor the MDBI can issue requests. They only respond to existing requests. The MMTP does not issue requests to the PVI.

All requests and responses have a header containing the following information:

Source of request or response	examples are PVI, Vendor, MDBI, MMTP
Destination of request or response	examples are PVI, Vendor, MDBI, MMTP
Success or failure indicator	true or false
Request	Vendor list Vendor inventory static
Request type	request, response
Action	read or write (write is a store)

If message is not a request or response the Request type should be blank.
The main body of the message appears immediately after the header. For requests it is the required parameters. For responses, it is the response body.

A request for vendors sent to the MDBI returns a list of vendors. A request for Vendor inventory given vendor id returns the inventory of all facilities for that vendor.

The Time To Live should not be infinite. A string property called “SenderIP” should contain the IP address of the sender as a string (e.g. 125.30.35.111).

Below is an example of a response returning all the vendors:


```
<?xml version="1.0" encoding="UTF-8" ?>
- <n:park xmlns:n="http://www.dummy-UPPParking-address">
- <header>
  <type>response</type>
  <source>MDBI</source>
  <dest>PVI</dest>
  <action></action>
  </header>
  <success>true</success>
  <responsetype>vendors</responsetype>
- <vendor>
  <vid>83483</vid>
  <vname>sf park</vname>
  <JMSQPVI2VENDOR>jmsqname</JMSQPVI2VENDOR>
  </vendor>
- <vendor>
  <vid>v834038</vid>
  <vname>another vendor</vname>
  <JMSQPVI2VENDOR>jmsqname2</JMSQPVI2VENDOR>
</vendor>
</n:park>
```

JMS Messages between Data Interface and Parking Vendors

Vendors must receive messages from the PVI. Below is an example of a request sent by the PVI to the Vendors. **vendorinventorystatic** is the only type of request the PVI sends to the parking vendors.

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <n:park xmlns:n="http://www.dummy-UPPParking-address">
- <header>
  <type>request</type>
  <source>PVI</source>
  <dest>VENDOR</dest>
  <action />
  </header>
  <request>vendorinventorystatic</request>
  </n:park>
```

Receipt of such a message should cause the vendors to reply with an XML vendorinventorystatic. Vendorinventorystatic contains all vendor facilities in the static format message as shown in this document. This should be published through the pFromVendor2PVI topic. A parking zone is considered a type of facility. The JMSCorrelationID should be returned in the response. The vendors should set the JMS Time to Live to 24 hours for static data.

Dynamic format messages should also use the pFromVendor2PVI topic. These messages are not requested but are sent when vendors report changes in pricing or availability. For Dynamic messages the Time to Live should be 5 minutes. A string property called "SenderIP" should contain the IP address of the sender as a string (e.g. 125.30.35.111). An optional statusBlock may be included in the response to indicate an error. This has an Error-NotificationLevel that can be either "information", "warning", or "error"

Differences from the ATIS Standard

ATIS Items Not Supported

Type ParkingLotInformation
Head , MessageHeader, generalStatus, furtherData
Head contains fields that are not applicable to parking such as Pedigree, Language, Charset, Table, Confidence and Urgency.

databaseIdentity is not applicable to the parking application.

The MessageHeader does have some similarities to the header mentioned earlier in this document but uses TMDD Message Number. GeneralStatus is used for referencing ITIS code. furtherData is for an optional URL link.

The following are additional types and fields not in the standard that were added for UPP Parking. These are candidates for inclusion in the ATIS standard.

Type ParkingLotDynamic is a new type as a subset of parking data. This is shorter than ParkingLotInformation which is used for static data.

Type Vendors contains vendor name, id, and the jms queue name toward the vendor.

Type ParkingVendorInventory contains all the parking facilities for a vendor.

Type PaymentOptions and PaymentMethods were added for UPP Parking.

Type AdditionalInfo was added to support the MMTP.
Adds the following elements:
ownershipAgencyType, structureType, peakUsageTimes,
contactphoneOnsite, constactphone2, contactphoneCustSvc, contactFax,
contaactphoneSecurity, contactphoneTTY, contactEmail,
contactWebsite1, contactWebsite2,
numberofspaces (vacancy by structure type),
hastransitagency, realtimeinfoavailable, haschargingport, hasemergencycallbox,
paymentdiscountAAA, paymentdiscountSenior, paymentdiscountCorporate,
paymentdiscountGroup, paymentDiscountHybrid,
paycashieratexit, kioskpaystations, paycahinsertslots,
epay, payonline, servicetypes,
reservAvailable, reservMethod, reservPhone, reservOnline, reservAddInfo,
lockersize, lockercost, lockerqty, maxoccupancy,additionaldesc, associatedfiles

Pricing for special events is handled as a PriceScheduleEntry.

The header that is used differs from the header in the ATIS standard. The header defined here has request type, action type, source, and destination.

Appendix D. Sample PVI Properties File

The parking vendor interface uses configurable parameters from a properties file. The parameters are defined in the table below.

PVI Parameters

Parameter Name	Description	Format
pJMSbroker	URL of JMS server.	Alpha-numeric
pJMSusername	User name to login to the JMS server (if necessary).	Alpha-numeric
pJMSpassword	Password for user to login to the JMS server (if necessary).	Alpha-numeric
pDB511Login	User name to be used to connect to the database.	Alpha-numeric
pDB511Password	Password to be used to connect to the database.	Alpha-numeric
pDB511Server	URL of the database server	Alpha-numeric
pMaximum	Max number of messages allowed on queue.	Integer > 0
pNoMessageTimeout	Maximum time without receiving message from a vendor before logging an error.	Integer > 0
pFROMVENDOR2PVI	Name of the JMS queue to receive parking information sent by parking vendors	Alpha-numeric
pJMSFromPVI2MMTP	Name of JMS topic to publish parking information to the MMTP	Alpha-numeric
pJMSFROMPVI2MDBI	Name of the JMS queue to send to the MDBI	Alpha-numeric
pJMSFROMMDBI2PVI	Name of the JMS queue PVI receives messages from the MDBI	Alpha-numeric
pReportEmailAddress	Address of where to send reports	Alpha-numeric
pConfigRequestTime	time of day to automatically send conf request to all vendors	Alpha-numeric (format HH:MM AM/PM)
pDynDataLogging	The allows writing all dynamic xml received to be stored in the database tblJMSMessageLog	Alpha-numeric

```
!Sample props file for UPPParkingVendorInterface

pDSNdbParking          = DSN_UPPParking
pJMSBROKER_URL         = localhost:2506
pJMSJNDI               = jndi
pFROMVENDOR2PVI         = FROMVENDOR2PVI
pJMSFromPVI2MMTP           = FromPVI2MMTP
pJMSFROMPVI2MDBI          = FROMPVI2MDBI
pJMSFROMMDBI2PVI          = FROMMDBI2PVI
pDB511URL               = SQL2005\PC-830257
pDB511USER             = operator
pDB511PWD              = dbpwd
pJMSUSER               = jmsusr
pJMSPWD                = jmsusrjmsuser
pJMSTTL                = 3600000
pAppPathRoot           = c:\
```

```
pDailyConfigRequest = 1
pConfigRequestTime  = 10:30
pReportEmailAddress = rpt@upp.com
pMaxMsgQueueSize    = 99999
pNoMsgTimeout       = 100
pLoggerOn           = 1
pLoggerConsole      = 1
pLoggerFile         = 1
pLoggerDebug        = 1
pLogfileName        = ParkingAgent.log
pDynDataLogging     = 1
pArchiveFoldername  = d:\uppparking\archive
pArchiveDate        = 2010-Dec-01
pLogKeep            = 120
```

Appendix E. Sample Log File

ParkingXML.log
This file logs all XML messages received from JMS. The prefix is the Agent system time. Below is a sample fragment of a parking xml log:

```
2010-02-03 09:31:22-0500, 172.18.209.105, <?xml version="1.0" encoding="UTF-8"?><n:root
xmlns:n="http://www.dummy-UPPParking-address">
<dynamicfmt>
    <parkingFacID>L2489084</parkingFacID>
    <vendorID>V347908348a</vendorID>
    <percentFull>35</percentFull>
    <spacesTotal>105</spacesTotal>
    <prices>
```

Parking.log
This file shows three types of messages:

Info:
Error:
Debug This can be disabled using the LoggerDebug property in the props file.

Below is a sample fragment of the PVI log:

```
INFO  2010-02-04 16:00:12-0500 loading parking xsd...
INFO  2010-02-04 16:04:15-0500 Queue received: 0  Invalidxml: 0
INFO  2010-02-04 16:04:15-0500 Config req sent to all vendors.
INFO  2010-02-04 16:04:15-0500 JMS and Deq thread have started.
ERROR 2010-02-04 16:04:15-0500 xml string is not valid because
ERROR 2010-02-04 16:04:15-0500 cvc-elt.1: Cannot find the declaration of element
'facilityconfiguration'.
ERROR 2010-02-04 16:04:15-0500 xml failed validation
ERROR 2010-02-04 16:04:15-0500 <?xml version="1.0" encoding="UTF-8"?>
```