Pawn Industry Web Service Template

Table of contents

Preface

Revision History

- 4/24/2009: Initial document
- 5/20/2009: Review/cleanup initial draft
- 5/27/2009: Review cleanup
- 6/9/2009: Added UploadImage and DeleteImage methods
- 7/27/2010: Added PropertyValue support
- 1/17/2013: Updated error codes
- 10/9/2013: Added SetNoTransactionDayForStore method
- 7/22/2014: Fixed VOID Ticket example
- 8/21/2014: Corrected missing error code
- 8/1/2015: Corrected text errors and added property value codes
- 12/17/2015: Error corrections
- 1/18/2016: Corrected ticketDateTime description
- 6/9/2017: Changed URLs to support TLS 1.2
- 11/6/2017: Added optional Pledgor Statement
- 10/31/2024: Added "Pdf' option to ImageType, added "Document" option to ImageCategory, updated WSDL URLs and updated Login Authentication

Intended Audience

This document is written for programmers familiar with Web Services application programming standards such as the Simple Object Access Protocol (SOAP), the Web Services Description Language (WSDL), and XML Schema Definition (XSD) language. It describes a subset of the LeadsOnline Web Service API that is relevant to use by Point-Of-Sale (POS) back office clients.

Notice of Non-Liability

LeadsOnline and the authors assume no liability for errors or omissions, or for damages resulting from the information or use of information contained in this manual.

Pawn Web Service API Reference

Enumerations

Enum TicketType

```
enum TicketType
{
```

```
  Unknown,
  Pawn,
  Buy,
  Trade,
  CheckCashed
}
```

TicketType is used to specify the type of ticket being submitted or updated.

Enum ItemType

```
enum ItemType
{
  Other,
  Jewelry,
  Firearm
}
```

The **ItemType** enumeration is used to specify the type of item associated with a ticket.

Enum ImageType

```
enum ImageType
{
  Jpeg,
  Png,
  Gif,
  Pdf
}
```

The **ImageType** is used to specify the type of image being attached to either a **Customer** or **Item** object.

Enum ImageCategory

```
enum ImageCategory
{
  Customer,
  CustomerID,
  Thumbprint,
  Signature,
```

```
  Document,
  Item
}
```

ImageCategory is used to specify what the captured image contains that is attached to the Customer or Item object.

- **Customer** – specifies a general image for the customer. This is only valid for Ticket.Customer images.
- **CustomerID** – specifies an ID image for the customer. This is only valid for Ticket.Customer images. If multiple ID images are available, multiple CustomerID images may be included.
- **Thumbprint** – specifies a thumbprint image for the customer. This is only valid for Ticket.Customer images.
- **Signature** – specifies a signature image for the customer. This is only valid for Ticket.Customer images.
- **Document** – specifies a scanned document. This is only valid for Ticket.Customer images.
- **Item** – specifies an item image. This is only valid for Ticket.Item images.

The **ImageType** is used to specify the type of image being attached to either a **Customer** or **Item** object.

Structures

Required Fields
Unless explicitly marked, fields are not required programmatically. However, local and state ordinances often require most of these fields to be populated.

Struct LoginInfo

```
struct LoginInfo
{
  int storeId;
  string userName;
  string password;
}
```

The **LoginInfo** structure is used to specify the login information for the POS to connect to the LeadsOnline Web Service. See Section

Login Authentication for more details.

Struct TicketKey

```
struct TicketKey
{
  TicketType ticketType;
  string ticketnumber;
  string ticketDateTime;
}
```

Specify the values that uniquely define a single Ticket in the LeadsOnline system.

- **ticketType** (required) – specify the ticket type.
- **ticketnumber** (required)– specify the ticket number.
- **ticketDateTime** (required) – specify the ticket date and time in local time (current time at the place of business – do not convert to any other time zone). The date value can be specified in one of the following formats, "MM/DD/YYYY", "YYYY-MM-DD" or "MM/DD/YYYY HH:MM".

Note: For a VOID ticket, the **ticketnumber** and **ticketType** values are encouraged but not required.

Struct Ticket

```
struct Ticket
{
  TicketKey key;
  string redeemByDate;
  Customer customer;
  Item[] items;
  bool isVoid;
  PropertyValue[] extraTicket;
}
```

Specify the values for a ticket to be added or updated in the LeadsOnline system. A Ticket is considered a transaction that takes place between the business and the customer for one or more items.

- **key** (required) – specify the TicketKey that uniquely defines the values for this ticket.
- **redeemByDate** – specify the date the items should be redeemed by. The date value can be specified in the following formats, "MM/DD/YYYY" or "YYYY-MM-DD".
- **customer** – specify the customer values.

- **items** (required) – specify an array of Items for this ticket.
- **isVoid** – specify as true to signify this is a VOID ticket.
- **extraTicket** – array of PropertyValue instances to pass additional values for the Ticket. See section Struct PropertyValue for more detail.

Note #1: A Ticket marked as VOID will also result in all **Ticket.items** being marked as VOID. Additionally, a Ticket with all **Ticket.items** marked VOID will result in the Ticket also being marked VOID.

Note #2: A Ticket with no **Ticket.items** must be marked VOID to be successfully submitted. A VOID Ticket does not require items, but is encouraged when available.

Struct Customer

```
struct Customer
{
  string name;
  string fname;
  string lname;
  string address1;
  string address2;
  string city;
  string state;
  string postalCode;
  string phone;
  string idType;
  string idNumber;
  string idType2;
  string idNumber2;
  string dob;
  int weight;
  int height;
  string eyeColor;
  string hairColor;
  string race;
  string sex;
  string remarks;
  Image[] images;
  PropertyValue[] extraCustomer;
}
```

The **Customer** refers to the person that is associated with the ticket and selling the item to the store.

- **name** – specify the name of the customer.

- **fname** – specify the first name of the customer.
- **lname** – specify the last name of the customer.
- **address1**, **address2**, **city**, **state**, **postalCode** – address fields for the customer.
- **phone** – specify the phone number of the customer.
- **idType** – specify the type of ID used by the customer, e.g. "TXDL".
- **idNumber** – specify the ID # used by the customer, e.g. their drivers license number.
- **idType2**, **idNumber2** – values for 2nd ID used by customer.
- **dob** – the Date Of Birth of the customer. The date value can be specified in the following formats, "MM/DD/YYYY" or "YYYY-MM-DD".
- **weight** – specify the weight of the customer in pounds. 0 can be specified if unknown.
- **height** – specify the height of the customer in inches. 0 can be specified if unknown.
- **eyeColor** – specify the eye color of the customer.
- **hairColor** – specify the hair color of the customer.
- **race** – specify the race of the customer.
- **sex** – specify the sex of the customer .
- **remarks** – specify any additional remarks about the customer.
- **images** – specify an array of Image structures to assign images for this Customer. Allowed image category types are Customer, CustomerID, Thumbprint, and Signature.
- **extraCustomer** – array of PropertyValue instances to pass additional values for the Customer. See section Struct PropertyValue for more detail.

Note: If **fname** and **lname** are specified, name may be left blank.

Struct Item

```
struct Item
{
  string make;
  string model;
  string serialNumber;
  string description;
  double amount;
  ItemType itemType;
  string itemStatus;
  Image[] images;
  bool isVoid;
  string employee;
  PropertyValue[] extraItem;
}
```

The Item structure is used to specify an item that is to be associated with a ticket. An Item generally represents a physical item involved in the transaction, such as a DVD player, a piece of jewelry, etc.

- **make** – specify the make of the item.
- **model** – specify the model of the item.
- **serialNumber** – specify the serial number of the item.
- **description** – specify the description of the item.
- **amount** – specify the amount in dollars. A value of 0.0 can be specified to signify that the amount is unknown or does not apply.
- **itemType** (required) – specify the ItemType of the item.
- **itemStatus** – specify the status of the item. Status of an item can be any text, but common values are "Pawn", "Buy", "Trade", "Consigned", "Hold", "Seized", and "Unknown".
- **images** – specify an array of Image structures to assign images to this specific item. The category for the item should be specified as ImageCategory.Item.
- **isVoid** – specify as true to signify that this is a VOID item.
- **employee** – specify the name of the employee handling the transaction.
- **extraItem** – array of PropertyValue instances to pass additional values for the Item. See section Struct PropertyValue for more detail.

Note #1: Though make, model, serialNumber, and description are not specified as being required, for a non-VOID item, it is required that at least one of the values is specified. Otherwise, the Item will be considered to be empty and not submitted with the ticket.

Note #2: For a VOID item, the itemType value is not required.

Struct Image

```
struct Image
{
  ImageCategory imageCategory;
  ImageType imageType;
  byte[] imageData;
}
```

The Image structure is used to upload an image to the web service. Images can be associated with a **Ticket.Customer** and **Ticket.Items**.

- **imageCategory** (required) – specify what the image is of.
- **imageType** (required) – specify what type of image.
- **imageData** (required) – specify the raw data of the image. This must be base64 encoded when sent in the SOAP XML packet. Some SOAP libraries may already take care of this for you and not require that you pre-encode the data bytes (e.g. Microsoft .NET SOAP API).

Struct Response

```
struct Response
{
  int errorCode;
  string errorResponse;
}
```

The **Response** is a structure that is returned by the **SubmitTransaction**, **UpdateTransaction**, and **CheckLogin** Web Service methods. The **Response** is used to specify either success or an error in the method call. If success, the **errorCode** will be 0 and **errorResponse** will be an empty string. If an error, **errorCode** will be non-zero and **errorResponse** will contain a string describing the error. See **Error Codes** for more details on error codes. It is possible that an exception may be thrown instead of a **Response** structure being returned. This can occur when there are communications or other SOAP related errors for improper formatted or specified values.

Struct DoNotBuyInfo

```
struct DoNotBuyInfo
{
  string agency;
  string ID;
  string state;
  string name;
  string dob;
  string contactName;
  string contactNumber;
}
```

The **DoNotBuyInfo** structure is a return value from the **CheckDoNotBuy** Web Service method. This object specifies a person whom has been specified by a law enforcement agency that should be prohibited from selling merchandise.

- **agency** – the agency that specified this person on the "No Buy" list.
- **ID** – the ID number of the person, typically a drivers license number.
- **state** – the 2 letter state abbreviation of the ID.
- **name** – the name of the person.
- **dob** – the date of birth of the person in the format of "MM/DD/YYYY".
- **contactName** – agency contact name .
- **contactNumber** – agency contact number.

Struct PropertyValue

```
struct PropertyValue
{
  string Name;
  string Value;
}
```

The **PropertyValue** structure is used to pass additional name/value pairs for a ticket, customer or item entries. An array of **PropertyValue** instances can be associated with the Ticket, Customer, or Item structures using their **extraTicket**, **extraCustomer**, and **extraItem** values, respectively. The extra values may then be available or displayed in the LeadsOnline web application, or for other custom data processing.

It is required that when the array of values are passed, that the name and value values be valid non-empty string values, and that the name be unique within the array. For example, values such as the following are valid,

"firearm_caliber" => "45"
"firearm_finish" => "chrome"

While, the following would not be valid since the name value is used more than once

"firearm_type" => "45"
"firearm_type" => "chrome"

Typically, the **PropertyValue** instances should only be used when additional values are to be passed that normally cannot be specified in the containing item. For example, do not pass the serial number for an item in the **PropertyValues**, instead using the **Item.serialNumber** field.

- **Name** – name of additional field value.
- **Value** – value of additional field.

A listing of the current PropertyValue names can be found below. If additional names are needed, contact LeadsOnline Support.

extraTicket:

- ORIGINAL_TICKET_NUMBER
- ORIGINAL_ENTER_DATE
- TICKET_LOAN_AMOUNT (The total value of a loan/buy when known.)
- TICKET_RATE_OF_INTEREST
- DEPARTMENT_NUMBER
- PLEDGOR_STATEMENT

extraCustomer:

- CUSTOMER_STORE_NUMBER (Store's unique identifier for the customer)
- CUSTOMER_NAME_MIDDLE
- CUSTOMER_NAME_SUFFIX
- CUSTOMER_ADDR_NUMBER (The house number of address; as "123" in "123 N Main Ave #4")
- CUSTOMER_ADDR_DIRECTION (The street direction; as "N" in "123 N Main Ave #4")
- CUSTOMER_ADDR_STREET (The street name; as "Main" in "123 N Main Ave #4")
- CUSTOMER_ADDR_SUFFIX (The street suffix/type; as "Ave" in "123 N Main Ave #4")
- CUSTOMER_ADDR_APT (The apt number; as "#4" in "123 N Main Ave #4")
- CUSTOMER_ID_EXPIRATION1
- CUSTOMER_ID_EXPIRATION2
- ID1_ISSUE_DT
- ID2_ISSUE_DT
- CUSTOMER_ID_ISSUER1 (Issuer of the ID; Federal, particular state or ?)
- CUSTOMER_ID_ISSUER2 (Issuer of the ID? Federal, particular state or ?)
- CUSTOMER_ID_CATEGORY1 (DL/StateID/SSN/MIL/etc.)
- CUSTOMER_ID_CATEGORY2 (DL/StateID/SSN/MIL/etc.)
- CUSTOMER_OWNER_CONSIGNEE_AGENT
- CUSTOMER_EMPLOYER
- CUSTOMER_EMPLOYER_PHONE (Also known as "work phone")
- CUSTOMER_EMPLOYER_ADDRESS1
- CUSTOMER_EMPLOYER_ADDRESS2
- CUSTOMER_EMPLOYER_CITY
- CUSTOMER_EMPLOYER_STATE
- CUSTOMER_EMPLOYER_ZIP
- CUSTOMER_DELIVERY_METHOD
- CUSTOMER_SIGNED_AGREEMENT
- CUSTOMER_IS_LEGAL_OWNER
- CUSTOMER_IS_LAWFUL_SELLER
- CUSTOMER_EMAIL
- CUSTOMER_IP_ADDRESS

extraItem:

- OAN (Owner-applied number)
- STORE_NCIC_CODE (The NCIC code supplied by the store)
- ITEM_LOAN_AMOUNT (Value of a loan for just a single item when known)
- ITEM_COLOR
- ITEM_CONDITION
- ITEM_QUANTITY
- ITEM_NOTES
- JEWELRY_STYLE
- JEWELRY_METAL

- JEWELRY_KARAT
- JEWELRY_WEIGHT
- JEWELRY_WEIGHT_UNIT ("dwt" or "grams" usually)
- JEWELRY_SIZE_LENGTH
- JEWELRY_SIZE
- JEWELRY_LENGTH
- JEWELRY_GENDER
- JEWELRY_CLASS_NAME (If a class ring, the name of the school)
- JEWELRY_CLASS_YEAR (If a class ring, the year)
- JEWELRY_STONE1_TYPE (Type of stone, such as diamond)
- JEWELRY_STONE1_COLOR
- JEWELRY_STONE1_QUANTITY
- JEWELRY_STONE1_SHAPE (Also known as cut)
- JEWELRY_STONE1_CARAT (Also known as weight)
- JEWELRY_STONE2_TYPE (Type of stone, such as diamond)
- JEWELRY_STONE2_COLOR
- JEWELRY_STONE2_QUANTITY
- JEWELRY_STONE2_SHAPE (Also known as cut)
- JEWELRY_STONE2_CARAT (Also known as weight)
- JEWELRY_STONE1_WT
- JEWELRY_STONE2_WT
- JEWELRY_TYPE
- PHONE_IMEI_NUMBER
- FIREARM_CALIBER
- FIREARM_TYPE
- FIREARM_ACTION
- FIREARM_FINISH
- FIREARM_BARREL
- FIREARM_IMPORTER
- FIREARM_BARREL_LENGTH

Methods

CheckLogin Method

```
Response CheckLogin(LoginInfo login)
```

The **CheckLogin** method can be used to verify the login credentials. No operation will be done on the LeadsOnline system other than the verification. The return **Response** value can be checked for successful authentication. See

Login Authentication for more details.

SubmitTransaction Method

```
Response SubmitTransaction(LoginInfo login, Ticket ticket)
```

The **SubmitTransaction** method is used to add a Ticket in the LeadsOnline system. The **TicketKey** in the Ticket is used to uniquely identify the Ticket to be added. Attempting to enter or update a Ticket with **TicketKey** values that already exists in the LeadsOnline system will normally result in an error.

Note: The return error is suppressed if the Ticket submitted matches the last successful ticket submitted. This allows the client to attempt to resubmit if no **Response** is received, without triggering an error.

UpdateTransaction Method

```
Response UpdateTransaction(LoginInfo login, TicketKey oldTicket, Ticket
ticket)
```

The **UpdateTransaction** method is used to specifically update a Ticket that has already been submitted to the LeadsOnline system. The **TicketKey oldTicket** is used to identify the existing Ticket to be updated. The Ticket after the update will then be associated with the new **Ticket.keys** values.

Note: The **TicketKey** values specified by **oldTicket** must already exist in the LeadsOnline system or an error will be returned.

CheckDoNotBuy Method

```
DoNotBuyInfo[] CheckDoNotBuy (LoginInfo login, string state, string ID)
```

The **CheckDoNotBuy** method can be used to check persons that have been reported by law enforcement agencies as not allowed to sell merchandise. An array of **DoNotBuyInfo** records is returned that match the passed-in state and ID values. Typically, a person's driver's license number and state are passed for a check. For the state value, the state 2-letter abbreviation should be specified.

See the **CheckDoNotBuy2** method; it is now the preferred method of checking for Do Not Buy persons.

Note: Since an array of records is returned, the **Response** value is not used for this method to return a success or error condition. If an error occurs, an Exception will be thrown describing the error code and error **Response**. Generally, only an Exception will be thrown if the state does not match a valid 2-letter abbreviation, or an ID number is not specified.

CheckDoNotBuy2 Method

```
DoNotBuyInfo[] CheckDoNotBuy2(LoginInfo login, string ID, string Name,
string Birthdate)
```

The **CheckDoNotBuy2** method can be used to check persons that have been reported by law enforcement agencies as not allowed to sell merchandise. An array of **DoNotBuyInfo** records is returned that match the passed-values. Either the ID value can be passed only, or any combination of at least 2 of the 3 values.

The **CheckDoNotBuy2** method is now the preferred method of checking for Do Not Buy persons.

Note: Since an array of records is returned, the **Response** value is not used for this method to return a success or error condition. If an error occurs, an Exception will be thrown describing the error code and error **Response**. Generally, only an Exception will be thrown if the required parameters are not specified.

UploadImage Method

```
Response UploadImage (LoginInfo login, TicketKey ticketKey, Image img,
int itemIndex)
```

The **UploadImage** method is used to upload an image for a specific ticket. The image will be attached to the ticket. It is not required that the ticket exist first. For example, the image can be uploaded first and the ticket added later. Any existing images for the ticket are not modified. The **itemIndex** only applies when **the.imageCategory** is set to **ImageCategory.Item**. The **itemIndex** specifies the 0-base index of the item within the ticket that the image is to be associated with.

The **Response.errorCode** is 0 if no error occurred. In this case, the **Response.errorResponse** will contain a unique name for the image created on the LeadsOnline server. The unique name can be used to delete the image if necessary using the **DeleteImage** web service method.

DeleteImage Method

```
Response DeleteImage (LoginInfo login, TicketKey ticketKey,
ImageCategory imageCategory, int itemIndex, string filename)
```

The **DeleteImage** method is used to delete an image that has been previously uploaded to a ticket using **UploadImage**. The **itemIndex** only applies when the **imageCategory** is set to **ImageCategory.Item**. The **itemIndex** specifies the 0-base index of the item within the ticket. The **imageCategory** specifies the **ImageCategory** type of the image to be deleted. The type must match the value that was used when the image was uploaded. The filename is the unique name that was assigned to the image on the LeadsOnline server; it must match the **Response.errorResponse** value that returned when the image was uploaded.

SetNoTransactionDayForStore Method

```
Response SetNoTransactionDayForStore (LoginInfo login, string
TransactionDate)
```

The **SetNoTransactionDayForStore** method is used to set a date as a day the store did not have any transactions. The TransactionDate should be in "MM/DD/YYYY" format.

The **Response.errorCode** is 0 if no error occurred. In this case, the **Response.errorResponse** will be the date that was set.

Other Web Service Related Items

Sandbox

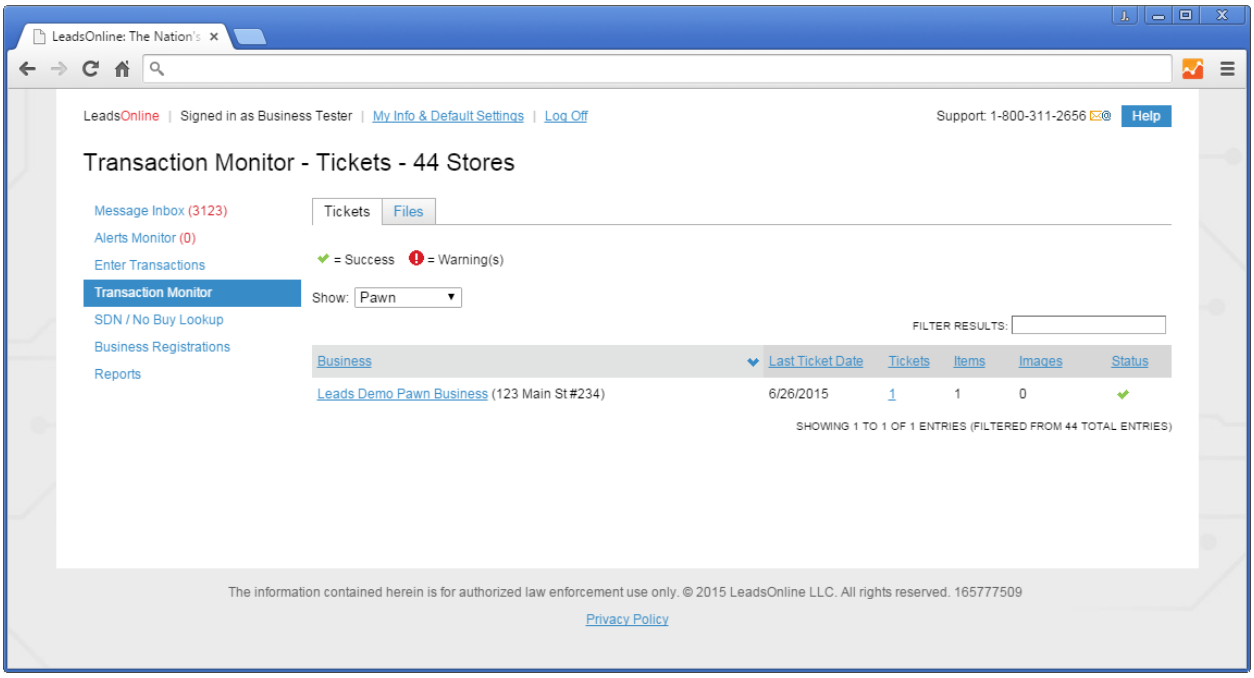
In order to support development, LeadsOnline maintains a sandbox environment which emulates the production environment. Transactions can be submitted to the sandbox and viewed thru a special web browser interface. The Pawn API Web Service URL for the sandbox is located at:

https://w3apisandbox.leadsonline.com/ticketWS.asmx?wsdl

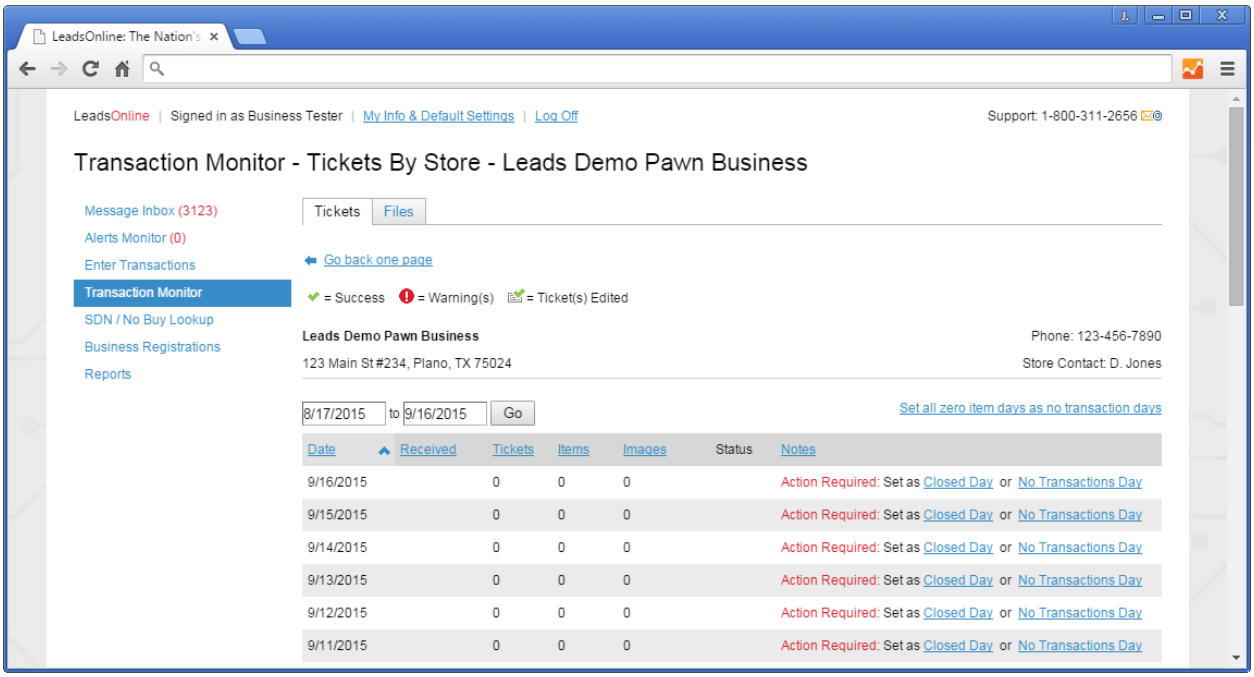
Login credentials for the sandbox are totally isolated from login credentials for the production site. Contact LeadsOnline Business Support for additional information.

Ticket Submit Verify

Tickets submitted using the web service can be verified by inspecting the Transaction Monitor area of the LeadsOnline website. For example, shown below is a sample Transaction Monitor.



Click on the Store Name link to see the tickets submitted.



Click on a ticket to see the full ticket details.

WSDL

The Web Service WSDL can be retrieved from either the sandbox or the production system (once released) at the following URLs:

- https://w3api.leadsonline.com/ticketWS.asmx?wsdl
- https://w3apisandbox.leadsonline.com/ticketWS.asmx?wsdl

Error Codes

The **Response** object is used to return back an error code and error response value. Following are the defined error codes:

- 0 – no error.
- -1 – an internal exception occurred. Please contact LeadsOnline Business Support.
- 1 – a date is specified using an unrecognized format.
- 2 – a ticket number is not specified.
- 3 – the store ID specified is not valid.
- 4 – login information is not correct.
- 5 – the TicketKey specified is not found (occurs in UpdateTransaction).
- 6 – the Ticket cannot be added or modified, it conflicts with an existing Ticket that was created through some other source than the Web Service.
- 7 – the TicketKey.ticketDateTime is out of range. Either the date specifies a future date, or it specifies a past date that is no longer kept in the LeadsOnline system.
- 8 – an invalid state abbreviation for the CheckDoNotBuy call.
- 9 – an invalid ID number for the CheckDoNotBuy call.
- 10 – an error occurred in the UploadImage call.
- 11 – an error occurred in the DeleteImage call.
- 12 – at least 2 of the 3 fields Name, IDNumber, BirthDate were not supplied.
- 13 – ticket cannot be modified if it already exists during a Submit.

Login Authentication

Each call to the Web Service requires authentication. This is done by passing the struct LoginInfo as a parameter. Login authentication is accomplished by passing a username and password in the LoginInfo username and password values. The username and password do not refer to a typical LeadsOnline user, instead there is a separate login name and password for remote access operations such as the Web Service. Contact LeadsOnline support for more details.

A storeId must also be provided. While the login identifies the company, the StoreId identifies the particular store involved. There are two options for StoreId:

> **Option 1** – Use the StoreId assigned by the company. A company can assign any numeric StoreId to a store as long as it is unique within their company.

Option 2 – Use a LeadsOnline assigned StoreId. LeadsOnline assigns a numeric ID to each store that is unique across all companies.

Note: A storeId is always required regardless of which method / option is used.

Image Support

Images can be uploaded for tickets and items using the web service. They can also be uploaded through other external means and attached to the ticket using the unique ticket keys. Contact LeadsOnline Business Support for more details on uploading images independent of this Pawn API Web Service.

Pawn Web Service Code Samples

Overview

This part of the document covers some examples using the Web Service API. The samples are shown using Microsoft .NET C# and the generated proxy classes created from the WSDL. Code using other client libraries and languages will obviously be different.

Submitting a Ticket

This example shows submitting a Ticket:

```
LoginInfo logon = new LoginInfo();
logon.userName = "name";
logon.password = "password";
logon.storeId = 1234;

Ticket t = new Ticket();
t.key = new TicketKey();
t.key.ticketnumber = "12345678";
t.key.ticketDateTime = "03/25/2009";
t.key.ticketType = TicketType.Pawn;
t.customer = new Customer();
t.customer.dob = "1960-05-14";
t.customer.hairColor = "black";
t.customer.height = 58;
t.customer.sex = "male";

Item item = new Item();
item.itemType = ItemType.Other;
item.description = "Glock Pistol";
item.itemStatus = "Pawn";
item.amount = 460.00;
```

```
PropertyValue pv1 = new PropertyValue();
pv1.Name = “caliber”;
pv1.Value = “45”
PropertyValue pv2 = new PropertyValue();
pv2.Name = “finish”;
pv2.Value = “black”
item.extraItem = new PropertyValue[] { pv1, pv2 };

t.items = new Item[] { item };

ticketWS ws = new ticketWS();
Response resp = ws.SubmitTransaction(login, t);
```

Updating a Ticket

This example shows updating a Ticket where the ticketKey values have changed:

```
LoginInfo logon = new LoginInfo();
logon.userName = "name";
logon.password = "password";
logon.storeId = 1234;

Ticket t = new Ticket();

t.key = new TicketKey();
t.key.ticketnumber = "ABCDEFG";
t.key.ticketDateTime = "03/23/2009";
t.key.ticketType = TicketType.Pawn;
t.customer = new Customer();
t.customer.dob = "1960-05-14";
t.customer.hairColor = "black";
t.customer.height = 58;
t.customer.sex = "male";

Item item1 = new Item();
Item1.itemType = ItemType.Other;
Item1.description = "Sony DVD Player";
Item1.itemStatus = "Pawn";
Item1.amount = 60.00;

Item item2 = new Item();
Item2.itemType = ItemType.Other;
Item2.description = "Honda Lawn Mower";
Item2.itemStatus = "Pawn";
```

```
Item2.amount = 40.00;

t.items = new Item[] { item1, item2 };

TicketKey oldKey = new TicketKey();
oldKey.ticketnumber = "12345678";
oldKey.ticketDateTime = "03/25/2009";
oldKey.ticketType = TicketType.Pawn;

ticketWS ws = new ticketWS();
Response resp = ws.UpdateTransaction(login, oldKey, t);
```

VOID Ticket

This example shows voiding a Ticket:

```
LoginInfo logon = new LoginInfo();
logon.userName = "name";
logon.password = "password";
logon.storeId = 1234;

Ticket t = new Ticket();
t.key = new TicketKey();
t.key.ticketnumber = "12345678";
t.key.ticketDateTime = "03/25/2009";
t.key.ticketType = TicketType.pawn;
t.isVoid = true;

ticketWS ws = new ticketWS();
Response resp = ws.UpdateTransaction (login, t);
```

Customer Images

This example shows a code snippet for attaching an Image to a Customer:

```
t.customer = new Customer();

Image img = new Image();
img.imageData = readByteArrayFromFile("customer.jpg");
img.imageType = ImageType.Jpeg;
img.imageCategory = ImageCategory.Customer;
```

```
t.customer.images = new Image[] { img };
```

readByteArrayFromFile() is a method (not shown) that simply reads the raw data bytes from an image file. In this case, it known that Microsoft .NET will automatically base64 encode any byte array values when sending the data.

Checking the Do Not Buy list

This example shows retrieving the "No Buy" list for a given ID#:

```
LoginInfo login = new LoginInfo();
login.userName = "name";
login.password = "password";
login.storeId = 1234;

ticketWS ws = new ticketWS();
DoNotBuyInfo[] dnbs = ws.CheckDoNotBuy(li, "TX", "12345678");
```

Email us at storesupport@leadsonline.com or call (800) 311-2656 for additional assistance.