

United States Department of Agriculture (USDA)  
Agricultural Marketing Service (AMS)  
Livestock Mandatory Price Reporting System (LMPRS)



**Livestock Mandatory Reporting (LMR)  
Web Service  
Client User Guide, v1.0**

U.S. Department of Agriculture  
Agricultural Marketing Service  
1400 Independence Avenue SW  
Washington DC 20250

**April 24, 2012**

## Table of Contents

<b>1</b>	<b>Overview .....</b>	<b>4</b>
1.1	Purpose of LMR Web Service .....	4
1.2	Document Audience .....	4
1.3	Definitions .....	4
<b>2</b>	<b>Design .....</b>	<b>5</b>
2.1	Goals.....	5
2.2	Implementation .....	5
<b>3</b>	<b>Usage .....</b>	<b>5</b>
3.1	Support and Announcement Notifications .....	5
3.2	Considerations for the Creation of Web Service Clients .....	6
3.3	Considerations for the Runtime Execution of Web Service Clients .....	6
3.4	The Anatomy of the REST URL for Obtaining Commodity Data.....	6
3.5	Path.....	7
3.6	Report Listings.....	8
3.6.1	Cattle .....	9
3.6.2	Hogs .....	10
3.6.3	Sheep .....	11
3.6.4	Beef .....	12
3.6.5	Lamb.....	12
3.7	Query String.....	12
<b>4</b>	<b>Further Support Information .....</b>	<b>13</b>
4.1	Example Query Strings.....	13
4.1.1	Retrieving a Report for a Specific Date .....	13
4.1.2	Retrieving all Reports after a Specific Date .....	14
4.1.3	Retrieving all Reports between two Dates .....	14
4.2	XML Format Overview .....	14
4.3	Sample Output.....	14
4.3.1	No results case.....	14
4.3.2	Simple results case .....	15
4.3.3	Complex report case .....	15
4.4	Error Handling.....	16
4.5	Consuming XML Data.....	16
4.5.1	XSD Schema .....	16
4.5.2	Example of XML being consumed by Microsoft Excel 2010 .....	17
4.6	Sample Code for Querying the Service (Java).....	21
4.7	Report Holidays .....	22

## Table of Exhibits

Exhibit 1	Example of Retrieved XML Data.....	7
Exhibit 2	Retrieving the Query Format for a Report.....	8
Exhibit 3	Error and Their Causes.....	16
Exhibit 4	Common LMR XML Elements.....	17
Exhibit 5	Saving and Opening the XML from MS Excel.....	18
Exhibit 6	MS Excel Open XML dialogue.....	18
Exhibit 7	MS Excel schema creation prompt.....	18
Exhibit 8	Example XML table in MS Excel showing sortable and filterable columns.....	19
Exhibit 9	MS Excel Save As dialogue with CSV selected.....	19
Exhibit 10	Sample CSV output in Notepad.....	20
Exhibit 11	Sample HTML output in Chrome.....	20

# 1 Overview

## 1.1 Purpose of LMR Web Service

The LMR Web Service for USDA AMS consists of a web interface for finding reports. In addition to this web interface, many users of the LMR Web Service may find it useful to have an interface that is more structured for automated consumption by external systems.

Web Services and XML are a broadly accepted solution to providing this kind of bridge between the producer of data services and automated consumers. XML offers advantages in that it is much easier to program against predictably demarcated output than it is to scrape user interfaces that are frequently subject to changes. User interfaces can often result in structural changes in a way that is not recognizable to automated systems or backwards compatible.

## 1.2 Document Audience

This document is technical in nature. This document was written to assist technical support staff in configuring this web service to pull data into their own environment or network. This document contains highly technical information and is not intended for non technical audience(s).

## 1.3 Definitions

Abbreviation	Definition
AMS	Agricultural Marketing Service
AJAX	Asynchronous JavaScript and XML
CSV	Comma-Separated Values
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LGMN	Livestock & Grain Market News
LMR	Livestock Reporting System
LMPRS	Livestock Mandatory Price Reporting System
MN	Market News
REST	Representational State Transfer
URL	Uniform Resource Locator
USDA	United States Department of Agriculture
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition

## 2 Design

### 2.1 Goals

The design of the LMR Web Service has been driven towards meeting these goals:

- Simple
- Lightweight
- Flexible
- Intuitive
- Extendable
- Integration available
- Minimal development effort required
- Structurally predictable
- Throughput predictable

### 2.2 Implementation

The LMR Web Service uses a REST interface with JSON embedded URLs and XML formatted data as the output in a comprehensive format. The URL allows filtering to the desired data with key fields such as the report date.

REST is technology that has recently gained traction as a web service implementation. REST has several advantages over traditional web services which do not standardize operations and may end up being too proprietary. REST is also well-suited to locating data through resource-intensive systems such as data repositories like LMR Web Service with maximum scalability and extensibility. This allows implementing the goals above over the WSDL interface. This means updates to the web service will not break existing clients and clients require less maintenance in general. REST requires the use of a URL to locate Resources using basic operations over HTTP using the existing infrastructure of the protocol. This also allows for a variety of interfaces as well that are more compatible with Web 2.0 technologies including AJAX, JSON and others.

## 3 Usage

### 3.1 Support and Announcement Notifications

Client users of the web service can receive announcements and change notifications by registering an email address online at the Market News Livestock and Grain portal here: <http://www.marketnews.usda.gov/portal/lg/lmprswsreg>. A link to subscribe to these LMR Web Service notifications will also be available in the “Tools” section of the Market News Livestock and Grain portal on the left hand navigation near the bottom.

You can email support at: [WASH.LGMN@AMS.USDA.GOV](mailto:WASH.LGMN@AMS.USDA.GOV) if you are unable to view the link. Questions about the web service or requests for the latest copy of this guide may also be sent to this same address.

### 3.2 Considerations for the Creation of Web Service Clients

The web service client can be implemented in a variety of ways. The simplest way would be a client that can create and encode HTTP URLs, format JSON from objects to string data types and parse the resulting XML, but because HTTP and XML are very text compatible, any text rendering program with sufficient logic for the formatting of these kinds of objects would be suitable; however, libraries that support these technologies will drastically reduce the implementation and improve reliability. However, the demands of other consuming systems and platforms may place other limitations.

The web service is designed to be flexible enough that these libraries are not absolutely required, but the use of languages that are rich with these libraries, or REST, in particular, are the easiest to support.

### 3.3 Considerations for the Runtime Execution of Web Service Clients

It is highly recommended that the date filters on web service requests be specific enough to retrieve a reasonable volume of data at any given time so as to allow for concurrent use or repeat use should a connection failure occur. The recommended date range is one week's worth of data at a time and no more than a month. It is also recommended that data be retrieved after 9 p.m. EST and before 8 a.m. EST during off peak business hours.

The web service uses times in the Central time zone. Standard and daylight time rules apply.

### 3.4 The Anatomy of the REST URL for Obtaining Commodity Data

The following is an example URL for the LMR web service. In the following paragraphs, we will break this down further to get an understanding of each part of the URL:

[http://mpr.datamart.ams.usda.gov/ws/report/v1/cattle/LM\\_CT100?filter={\"filters\":\[{\"fieldName\":\"Report%20date\",\"operatorType\":\"EQUAL\",\"values\":\[\"12/06/2010\"\]}\]}](http://mpr.datamart.ams.usda.gov/ws/report/v1/cattle/LM_CT100?filter={\)

The URL above is broken into 3 main parts.

1. Livestock Mandatory Price Reporting, Data Mart URL - This will always be the same.
2. Report Web Service Path – This is the path to the service which will provide data for any individual report. This contains the version of the web service as well as the commodity and report identifier.
3. Report Filter Parameters – These are used to choose the date or date range for the reports that are being requested. The filter follows a JSON format.

In this example, we are looking for the version 1 web service and getting a cattle report with the given a SLUG number of LM\_CT104. We are also specifying that we want the report from 12/06/2010. This request will return the XML for this one report.

The content of the report XML that is returned will vary from report to report and will follow the syntax used in LMR Data Mart's current XML structure. This XML output provided is structured in the current report/sub-report structure, which provides the user an easily usable hierarchical structure for each report.

The following sections will break down the URL in further detail.

### 3.5 Path

This first section of the report URL that can be altered is:

**/ws/report/v1/cattle/LM\_CT100**

The path example contains 5 parts:

1. Application: ws – web service
2. Module: can be “report” or “config”
  - report – gives the actual xml data and requires a query string.

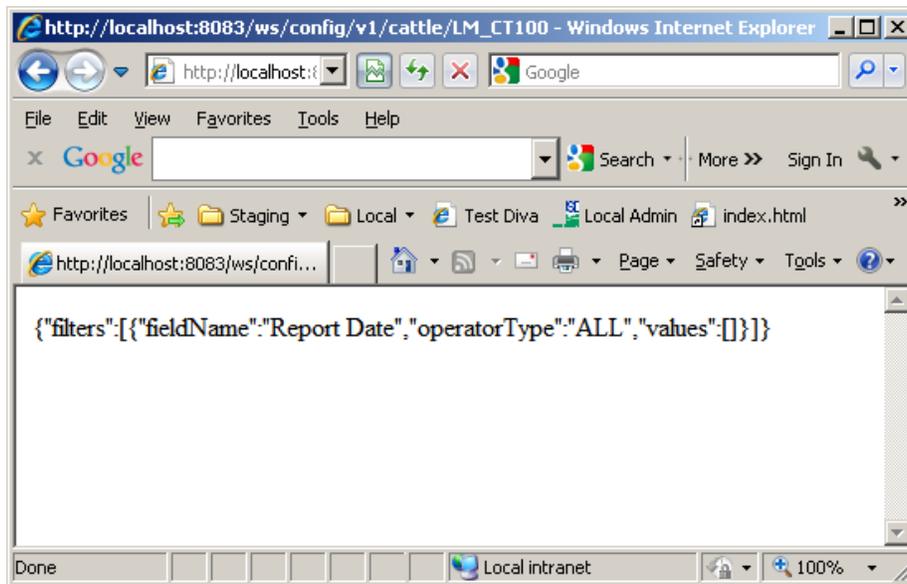
#### Exhibit 1 Example of Retrieved XML Data

```
<?xml version="1.0" encoding="UTF-8" ?>
- <results exportTime="2012-01-03 17:26:25 EST">
- <report label="5 Area Daily Weighted Average Direct Slaughter Cattle - Negotiated" slug="LM_CT100">
- <record report_date="12/06/2010" previous_day_head_count="2,317" narrative="null">
- <report label="Detail">
- <record class_description="Steer" selling_basis_description="Dressed" grade_description="Over 80% Choice" head_count="null" weight_range_low="null" weight_range_high="null" weight_range_avg="null" price_range_low="null" price_range_high="null" weighted_avg_price="null" />
- <record class_description="Steer" selling_basis_description="Dressed" grade_description="65 - 80% Choice" head_count="50" weight_range_low="942" weight_range_high="942" weight_range_avg="942" price_range_low="165.00" price_range_high="165.00" weighted_avg_price="165.00" />
- <record class_description="Steer" selling_basis_description="Dressed" grade_description="35 - 65% Choice" head_count="505" weight_range_low="909" weight_range_high="950" weight_range_avg="942" price_range_low="165.00" price_range_high="165.00" weighted_avg_price="165.00" />
- <record class_description="Steer" selling_basis_description="Dressed" grade_description="0 - 35% Choice" head_count="null" weight_range_low="null" weight_range_high="null" weight_range_avg="null" price_range_low="null" price_range_high="null" weighted_avg_price="null" />
- <record class_description="Steer" selling_basis_description="Dressed" grade_description="Total all grades" head_count="555" weight_range_low="909" weight_range_high="950" weight_range_avg="942" price_range_low="165.00" price_range_high="165.00" weighted_avg_price="165.00" />
- <record class_description="Steer" selling_basis_description="Live" grade_description="Over 80% Choice" head_count="136" weight_range_low="1,500" weight_range_high="1,500" weight_range_avg="1,500" price_range_low="102.00" price_range_high="102.00" weighted_avg_price="102.00" />
- <record class_description="Live" selling_basis_description="Live" grade_description="65 - 80% Choice" head_count="null" weight_range_low="null" weight_range_high="null" weight_range_avg="null" price_range_low="null" price_range_high="null" weighted_avg_price="null" />
- <record class_description="Steer" selling_basis_description="Live" grade_description="35 - 65% Choice" head_count="66" weight_range_low="1,225" weight_range_high="1,225" weight_range_avg="1,225" price_range_low="102.00" price_range_high="102.00" weighted_avg_price="102.00" />
- <record class_description="Steer" selling_basis_description="Live" grade_description="0 - 35% Choice" head_count="null" weight_range_low="null" weight_range_high="null" weight_range_avg="null" price_range_low="null" price_range_high="null" weighted_avg_price="null" />
- <record class_description="Steer" selling_basis_description="Live" grade_description="Total all grades" head_count="202" weight_range_low="1,225" weight_range_high="1,500" weight_range_avg="1,410" price_range_low="102.00" price_range_high="102.00" weighted_avg_price="102.00" />
- <record class_description="Heifer" selling_basis_description="Dressed" grade_description="Over 80% Choice" head_count="null" weight_range_low="null" weight_range_high="null" weight_range_avg="null" price_range_low="null" price_range_high="null" weighted_avg_price="null" />
- <record class_description="Heifer" selling_basis_description="Dressed" grade_description="65 - 80% Choice" head_count="116" weight_range_low="850" weight_range_high="850" weight_range_avg="850" price_range_low="165.00" price_range_high="165.00" weighted_avg_price="165.00" />
- <record class_description="Heifer" selling_basis_description="Dressed" grade_description="35 - 65% Choice" head_count="null" weight_range_low="null" weight_range_high="null" weight_range_avg="null" price_range_low="null" price_range_high="null" weighted_avg_price="null" />
- <record class_description="Heifer" selling_basis_description="Dressed" grade_description="0 - 35% Choice" head_count="null" weight_range_low="null" weight_range_high="null" weight_range_avg="null" price_range_low="null" price_range_high="null" weighted_avg_price="null" />
```

- config – provides the syntax used to select reports by more specific attributes like reporting date. More detail on the output of this module is to be explained in the Query String section that follows. Config itself does not require any query string,

but shows a sample JSON template that can be used to simplify formatting requests with the details below that can be plugged into the JSON format.

Exhibit 2 Retrieving the Query Format for a Report



**Note: “ALL” is a placeholder and should not be used in the client. This should be replaced with the actual operations discussed below i.e.: “EQUAL”, “GREATER”, “BETWEEN”.**

3. Version: in this case, “v1” for version 1. In the future, AMS may publish updated versions for each report and a new URL will be provided (v2, v3, etc.) AMS will notify users of the updated service versions via announcements on the AMS market news portal and via the Livestock and Grain Market News Portal email distribution list.
4. Species: in this case, “cattle”. The species available in the web service are the same as those available in the LMR Data Mart. These are:
  - Cattle
  - Hogs
  - Sheep
  - Beef
  - Lamb
5. Slug Number: in this case, “LM\_CT100”. The slug numbers available in the web service are the same as those available in LMR Data Mart. These are listed by species and frequency, below:

### 3.6 Report Listings

Below are the report listings grouped by species and then by report name which shows the slug number and the report title, along with the time period of the data contained in each report

relative to the report date. The description of the report period always refers to the previous business days and not calendar days. The slug numbers used are case sensitive and should remain in uppercase. The time periods referred to below are in Central Time.

### 3.6.1 Cattle

- **Daily Cattle**
  - (LM\_CT100) 5 Area Daily Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT103) Daily Prior Day Direct Cow and Bull Report - Negotiated (12:00 AM to 11:59 PM previous day)
  - (LM\_CT106) National Daily Direct Slaughter Cattle - Committed and Delivered Cattle - Summary (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT109) National Daily Direct Slaughter Cattle - Formulated Purchases - Summary (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT115) National Daily Direct Slaughter Cattle - Negotiated Purchases - Summary (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT118) TX/OK/NM Daily Direct Slaughter Cattle - Negotiated Purchases - Summary (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT121) Kansas Daily Direct Slaughter Cattle - Negotiated Purchases - Summary (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT124) Nebraska Daily Direct Slaughter Cattle - Negotiated Purchases - Summary (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT125) Texas/Oklahoma Daily Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT126) Kansas Daily Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT127) Nebraska Daily Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT128) Colorado Daily Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT129) Iowa/Minnesota Daily Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT134) Colorado Daily Direct Slaughter Cattle - Negotiated Purchases - Summary (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT137) Iowa/Minnesota Daily Direct Slaughter Cattle - Negotiated Purchases - Summary (9:30 AM previous day to 9:30 AM current day)
  - (LM\_CT138) Direct Slaughter Cow and Bull Report - Plant Delivered Bids (9:30 AM previous day to 9:30 AM current day)
- **Weekly Cattle**
  - (LM\_CT139) TX-OK-NM Weekly Direct Slaughter Cattle - Formula, Grid and Contract Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
  - (LM\_CT140) Kansas Weekly Directly Slaughter Cattle - Formula, Grid and Contract Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
  - (LM\_CT141) Nebraska Weekly Directly Slaughter Cattle - Formula, Grid and Contract Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
  - (LM\_CT142) National Weekly Direct Slaughter Cattle - Committed and Delivered Cattle (9:30 AM previous Monday to 9:30 AM current Monday)
  - (LM\_CT150) 5 Area Weekly Weighted Average Direct Slaughter Cattle (9:30 AM previous Monday to 9:30 AM Current Monday)
  - (LM\_CT151) National Weekly Direct Slaughter Cattle - Formulated and Forward Contract - Domestic (9:30 AM previous Monday to 9:30 AM current Monday)
  - (LM\_CT152) National Weekly Direct Slaughter Cattle - Formulated and Forward Contract - Import (9:30 AM previous Monday to 9:30 AM current Monday)

- (LM\_CT153) National Weekly Direct Slaughter Cattle - Packer Owned Cattle (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT154) National Weekly Direct Slaughter Cattle - Negotiated Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT155) National Weekly Direct Slaughter Cattle - Premiums and Discounts (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT156) Texas-Oklahoma Weekly Direct Slaughter Cattle - Negotiated Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT157) Kansas Weekly Direct Slaughter Cattle - Negotiated Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT158) Nebraska Weekly Direct Slaughter Cattle - Negotiated Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT159) National Weekly Direct Slaughter Cattle - Formula Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT160) Texas-Oklahoma Weekly Direct Slaughter Cattle - Formula Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT161) Kansas Weekly Direct Slaughter Cattle - Formula Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT162) Nebraska Weekly Direct Slaughter Cattle - Formula Purchases (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT163) TX/OK/NM Weekly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT164) Kansas Weekly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT165) Nebraska Weekly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT166) Colorado Weekly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT167) IA/MN Weekly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT168) National Weekly Direct Cow and Bull Report - Negotiated Price (9:30 AM previous Monday to 9:30 AM current Monday)
- (LM\_CT169) 5-Area Weekly Slaughter Cattle - Premiums and Discounts (9:30 AM previous Monday to 9:30 AM current Monday)
- **Monthly Cattle**
  - (LM\_CT180) 5-Area Monthly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM first day of previous month to 9:30 AM the first day of current month)
  - (LM\_CT181) TX/OK/NM Monthly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM first day of previous month to 9:30 AM the first day of current month)
  - (LM\_CT182) Kansas Monthly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM first day of previous month to 9:30 AM the first day of current month)
  - (LM\_CT183) Nebraska Monthly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM first day of previous month to 9:30 AM the first day of current month)
  - (LM\_CT184) Colorado Monthly Weighted Average Direct Slaughter Cattle - Negotiated (9:30 AM first day of previous month to 9:30 AM the first day of current month)
  - (LM\_CT185) IA/MN Monthly Weighted Average Direct Slaughter Cattle – Negotiated(9:30 AM first day of previous month to 9:30 AM the first day of current month)

### 3.6.2 Hogs

- **Daily Swine**
  - (LM\_HG200) National Daily Direct Hog Prior Day - Purchased Swine (Includes all purchases from the prior day-midnight to midnight.)

- (LM\_HG201) National Daily Direct Hog Prior Day - Slaughtered Swine (Includes all data for any swine slaughtered from the prior day-midnight to midnight.)
- (LM\_HG202) National Daily Direct Hogs - Morning (Includes all purchases from midnight to 9:30 AM.)
- (LM\_HG203) National Daily Direct Hogs - Afternoon (Includes all purchases from midnight to 1:30 PM.)
- (LM\_HG204) Iowa/Minnesota Daily Direct Prior Day Hog - Purchased Swine (Includes all purchases from the prior day-midnight to midnight.)
- (LM\_HG205) Iowa/Minnesota Daily Direct Hogs - Morning (Includes all purchases from midnight to 9:30 AM in Iowa and Minnesota.)
- (LM\_HG206) Iowa/Minnesota Daily Direct Hogs - Afternoon (Includes all purchases from midnight to 1:30 PM in Iowa and Minnesota.)
- (LM\_HG207) Eastern Cornbelt Daily Direct Prior Day Hog - Purchased Swine (Includes all purchases from the prior day-midnight to midnight.)
- (LM\_HG208) Western Cornbelt Daily Direct Prior Day Hog - Purchased Swine (Includes all purchases from the prior day-midnight to midnight.)
- (LM\_HG209) Eastern Cornbelt Daily Direct Hogs - Morning (Includes all purchases from midnight to 9:30 AM in Eastern Region.)
- (LM\_HG210) Eastern Cornbelt Daily Direct Hogs - Afternoon (Includes all purchases from midnight to 1:30 PM in Eastern Region.)
- (LM\_HG211) Western Cornbelt Daily Direct Hogs - Morning (Includes all purchases from midnight to 9:30 AM in Western Region.)
- (LM\_HG212) Western Cornbelt Daily Direct Hogs - Afternoon (Includes all purchases from midnight to 1:30 PM in Western Region.)
- (LM\_HG213) National Daily Base Lean Hog Carcass - Slaughter Cost (Includes data from slaughtered swine from the prior day-midnight to midnight.)
- (LM\_HG215) National Daily Net Price Distribution (Includes data from slaughtered swine from the prior day-midnight to midnight.)
- (LM\_HG230) National Daily Direct Prior Day Sow and Boar - Purchased Swine (Includes all purchases from the prior day-midnight to midnight.)
- (LM\_HG231) Iowa/Minnesota Daily Direct Prior Day Sow and Boar - Purchased Swine (Includes all purchases from the prior day-midnight to midnight Iowa Minnesota.)
- (LM\_HG232) Eastern Cornbelt Daily Direct Prior Day Sow and Boar - Purchased Swine (Includes all purchases from the prior day-midnight to midnight Eastern Region.)
- (LM\_HG233) Western Cornbelt Daily Direct Prior Day Sow and Boar - Purchased Swine (Includes all purchases from the prior day-midnight to midnight Western Region.)
- **Weekly Swine**
  - (LM\_HG214) National Weekly Direct Prior Day Hog - Purchased Swine (Includes swine purchase and slaughter information during the previous week-Monday to Friday, report released on Monday.)
  - (LM\_HG250) National Weekly Direct Swine Non-Carcass Merit Premium (Includes premiums paid for swine during the previous week-Monday to Friday, report released on Monday.)

### 3.6.3 Sheep

- **Daily Lamb**
  - (LM\_LM301) National Daily Lamb - Committed and Delivered Lambs (Data discontinued Feb 8, 2006)
  - (LM\_LM302) National Daily Lamb - Negotiated and Formulated Purchases (1:30 PM previous day to 1:30 PM current day)
  - (LM\_LM303) Western U.S. Daily Lamb - Committed and Delivered (Data discontinued Feb 8, 2006)
  - (LM\_LM304) Western U.S. Daily Lamb - Negotiated and Formula Base Price (not reported due to confidentiality since Nov 23, 2010)
- **Weekly Lamb**
  - (LM\_LM351) National Weekly Lamb (Monday to Friday previous week)
  - (LM\_LM355) Western U.S. Weekly Lamb (not reported due to confidentiality since Nov 23, 2010)

### 3.6.4 Beef

- **Daily Boxed Beef**
  - (LM\_XB401) National Daily Boneless Cow Beef & Beef Trimmings - Negotiated Sales - Afternoon (From 1:30 PM previous day to 1:30 PM current day)
  - (LM\_XB403) National Daily Boxed Beef Cutout & Boxed Beef Cuts - Negotiated Sales - Afternoon (From 1:30 PM previous day to 1:30 PM current day)
  - (LM\_XB405) National Daily Cutter Cow Cutout and Boxed Cow Beef Cuts - Negotiated – Afternoon(From 1:30 PM 7 days earlier to 1:30 PM current day)
- **Weekly Boxed Beef**
  - (LM\_XB450) National Weekly Boneless Cow Beef & Beef Trimmings - Formulated Sales (From 1:30 PM Previous Friday to 1:30 PM current Friday)
  - (LM\_XB452) National Weekly Boxed Beef Cuts - Branded Product (From 1:30 PM two Friday's ago to 1:30 PM previous Friday)
  - (LM\_XB454) National Weekly Boxed Beef Cuts - Formulated Sales (From 1:30 PM two Friday's ago to 1:30 PM previous Friday))
  - (LM\_XB455) National Weekly Boxed Beef Cuts - Forward Negotiated Sales (From 1:30 PM two Friday's ago to 1:30 PM previous Friday)
  - (LM\_XB456) National Weekly Boxed Beef Cuts - Prime Product (From 1:30 PM two Friday's ago to 1:30 PM previous Friday)
  - (LM\_XB459) National Weekly Boxed Beef Cutout & Boxed Beef Cuts - Negotiated Sales (From 1:30 PM Previous Friday to 1:30 PM current Friday)
  - (LM\_XB460) National Weekly Boneless Cow Beef & Beef Trimmings - Negotiated Sales (From 1:30 PM Previous Friday to 1:30 PM current Friday)
  - (LM\_XB461) Final National Weekly Cutter Cow Cutout and Boxed Cow Beef Cuts - Negotiated (From 1:30 PM Previous Friday to 1:30 PM current Friday)
  - (LM\_XB462) National Weekly Boxed Beef Cuts - Ungraded Product(From 1:30 PM two Friday's ago to 1:30 PM previous Friday)

### 3.6.5 Lamb

- **Daily Lamb**
  - (LM\_XL500A) National 5-Day Rolling Average Boxed Lamb Cuts - Negotiated Sales (From 2:00 PM 7 days earlier to 2:00 PM current day)
  - (LM\_XL501) National Daily Lamb Carcass Report (From 2:30 PM previous day to 2:30 PM current day) except Friday where the bottom half of the report is (From 2:30 PM Previous Friday to 2:30 PM current Friday day)
- **Weekly Lamb**
  - (LM\_XL552) National Weekly Boxed Lamb Cuts - Imported Product (From 12:01 AM C Sunday to 11:59 PM Saturday of Previous week)
  - (LM\_XL555) National Weekly Comprehensive Lamb Carcass Report (From 2:30 PM Previous Friday to 2:30 PM current Friday day)

## 3.7 Query String

The second part of the URL is the query string portion which follows:

```
filter={"filters":[{"fieldName":"Report Date","operatorType":"EQUAL","values":["12/06/2010"]}]}
```

This query string begins with the “?” character and is followed by name value pairs like so:  
“?name1=value1&name2=value2&name3=value3.....”

Here, we have only one parameter name “filter=\*” where \* is a JSON formatted name value pair. JSON stands for JavaScript Object Notation and is a standard format for performing data exchange. In this case, we are using it to provide filters for the reports that will be retrieved from the LMR Web Service. For more information on JSON, please click [here](#).

Each JSON set starts and ends with curly brackets { }. Within the curly brackets are a number of “name”: “value” pairs that represent the parameters for the filter. The values can either be simple values, a list of values within square brackets [ ], or another JSON value held within curly brackets { }.

The JSON for this filter is outlined below:

1. filters - the value in square brackets is another JSON set.
  - a. fieldName – the name of the field to use in the filter. In this case, it’s “Report Date”. The %20 may show up as URL encoded whitespace for the space in Report Date and should not be a cause for alarm.
  - b. operatorType – we usually want to limit this to “EQUAL” so we don’t have excessive volume. There is support for the operators: GREATER, and BETWEEN. All of them take one operand except BETWEEN which has two operands with this syntax alteration:

```
{"filters":[{"fieldName":"Report Date","operatorType":"BETWEEN","values":["12/06/2010","12/12/2010"]}]}
```

- c. values – Here is what we want the “Report Date” field equal to. In this case, “12/06/2010”. This is in square brackets as more than one value could be listed, as is the case of when the operatorType is “BETWEEN”.

Currently, the LMR web service will support “Report Date” as the only parameter. Support for more advanced searches may be provided at a future date.

## 4 Further Support Information

### 4.1 Example Query Strings

#### 4.1.1 Retrieving a Report for a Specific Date

In this example, we obtain all the LM\_CT100 – 5 Area Daily Weighted Average Direct Slaughter Cattle - Negotiated files on 12/06/2010:

```
http://mpr.datamart.ams.usda.gov/ws/report/v1/cattle/LM_CT100?filter={"filters":[{"fieldName":"Report Date","operatorType":"EQUAL","values":["12/06/2010"]}]}
```

## 4.1.2 Retrieving all Reports after a Specific Date

In this example, we obtain all the LM\_CT106 files after, but not on 05/20/2010:

```
http://mpr.datamart.ams.usda.gov/ws/report/v1/cattle/LM_CT106?filter={"filters": [{"fieldName": "Report Date", "operatorType": "GREATER", "values": ["05/20/2010"]}]}
```

## 4.1.3 Retrieving all Reports between two Dates

In this example, we obtain all the LM\_CT115 - National Daily Direct Slaughter Cattle - Negotiated Purchases - Summary files between 12/06/2010 and 12/13/2010 inclusively:

```
http://mpr.datamart.ams.usda.gov/ws/report/v1/cattle/LM_CT115?filter={"filters": [{"fieldName": "Report Date", "operatorType": "BETWEEN", "values": ["12/06/2010", "12/13/2010"]}]}
```

## 4.2 XML Format Overview

The XML follows a fairly simple pattern. Here we'll go through the elements you will frequently encounter and show a simplified sample. All XML contains elements in this order of nesting:

- Results – a set of results. The results will usually provide the exportTime.
- Report – a report counts as a result. The Report element will usually have a label and/or SLUG attribute.
- Record – A record is an element with attribute name and attribute value pairs. These are like table rows and the attribute names map to columns. These attributes are meant to be intuitive enough for human reading and will remain consistent within a report. A record element may contain another report inside of it as a sub-report.

## 4.3 Sample Output

Here are some samples of output varying in complexity:

### 4.3.1 No results case

```
<results exportTime="2020-02-06 16:03:07 CST">  
  <report label="National Daily XX - Purchased XX" slug="ZX_YY100"/>  
</results>
```

### 4.3.2 Simple results case

```
<results exportTime="2012-01-17 13:29:19">
  <report label="National Daily XX - Purchased XX" slug="ZX_YY100">
    <record report_date="01/09/2012" reported_for_date="01/06/2012">
      <report label="Volume By Purchase ">
        <record type="Negotiated" volume="9,999" />
        <record type="Packages" volume="11,111" />
      </report>
    </record>
  </report>
</results>
```

### 4.3.3 Complex report case

```
<results exportTime="2012-02-06 16:18:30 CST">
  <report label="National Daily Direct Hog Prior Day - Slaughtered Swine" slug="LM_HG201">
    <record report_date="12/20/2010" for_date_begin="12/17/2010" narrative="null">
      <report label="Summary">
        <record barrows_head_count="620,741" sows_head_count="null" boars_head_count="null"/>
      </report>
      <report label="Barrows/Gilts">
        <record purchase_type="Prod. Sold Negotiated"
          head_count="30,442" base_price="66.43" avg_net_price="68.41" lowest_net_price="54.45"
          highest_net_price="77.33" avg_live_weight="267.43" avg_carcass_weight="202.16"
          avg_sort_loss="1.40" avg_backfat=".74" avg_loin_depth="2.33" loineye_area="6.99"
          avg_lean_percent="53.63"/>
        <record purchase_type="Prod. Sold Other Market Formula"
          head_count="58,347" base_price="67.03" avg_net_price="70.23"
          lowest_net_price="61.65" highest_net_price="77.78" avg_live_weight="281.96"
          avg_carcass_weight="211.72" avg_sort_loss="-1.83"
          avg_backfat=".74" avg_loin_depth="2.60" loineye_area="7.82" avg_lean_percent="54.99"/>
      </report>
      <report label="Carcass Measurements">
        <record avg_carcass_weight="208.93" avg_backfat=".74"
          avg_loin_area="7.40" avg_loin_depth="2.47" standard_lean_yield="51.80"
          wtd_avg_base="68.11" wtd_avg_net_price="70.22"/>
      </report>
      <report label="Sows/Boars">
        <record purchase_type="Negotiated"
          head_count="null" base_price_300="null" base_price_450="null" base_price_500="null"
          avg_live_wt_300="null" avg_live_wt_450="null" avg_live_wt_500="null"/>
        <record purchase_type="All Purchase Types"
          head_count="null" base_price_300="null" base_price_450="null" base_price_500="null"
          avg_live_wt_300="null" avg_live_wt_450="null" avg_live_wt_500="null"/>
      </report>
      <report label="14-Day Scheduled Swine">
        <record delivery_date="December 20, 2010" head_count="409,854"/>
        <record delivery_date="December 21, 2010" head_count="382,392"/>
        <record delivery_date="December 22, 2010" head_count="378,518"/>
        <record delivery_date="December 23, 2010" head_count="345,397"/>
        <record delivery_date="December 24, 2010" head_count="98,782"/>
        <record delivery_date="December 25, 2010" head_count="2,480"/>
        <record delivery_date="December 26, 2010" head_count="16,632"/>
        <record delivery_date="December 27, 2010" head_count="213,942"/>
        <record delivery_date="December 28, 2010" head_count="138,476"/>
        <record delivery_date="December 29, 2010" head_count="119,137"/>
        <record delivery_date="December 30, 2010" head_count="103,476"/>
        <record delivery_date="December 31, 2010" head_count="65,164"/>
        <record delivery_date="January 01, 2011" head_count="1,000"/>
        <record delivery_date="January 02, 2011" head_count="8,609"/>
      </report>
    </record>
  </report>
</results>
```

```
</report>
</results>
```

## 4.4 Error Handling

In addition to the standard HTTP errors or Page Not Found, there are some standard common errors that may be encountered:

Exhibit 3 Error and Their Causes

Message	Cause
Unknown report "AA_BB999".	The URL contained an invalid report name. e.g.: "LM_CT999"
Filter not found	The part trailing the '?' was not read. This part is required.
Failed to parse filter string	For example an incomplete filter was read. e.g.: <i>filter={"filters":[</i>

## 4.5 Consuming XML Data

### 4.5.1 XSD Schema

XML Schema Definition (XSD) is a file that defines the potential element, attribute and other data structures allowed and available in a given XML document. XSD is not published for the LMR Web Service, but can be generated with a W3C compliant XSD reverse engineering tool, such as Altova's XMLSpy, XMLFox, Liquid XML Studio, etc. Using a schema is not necessary for using the LMR Web Service, but is possible to generate for those who require one. The following is a guide for how to create an XSD for the LMR Web Service. Some of these reverse engineering programs may not allow repeated nesting of elements, so XSLT transformation may be a desired preprocessing step beforehand as different client platforms may require adherence to these extra constraints. The table below captures some of the rules needed for generating an XSD document describing a report from a web service request. This includes the top level tags and sub-tags, and how they are related to one another.

## Exhibit 4 Common LMR XML Elements

Tag	Quantity	Required Attributes	Sub-Elements
results	1	exportTime	report <sup>1</sup>
report <sup>1</sup>	1-*	label, slug	record
record	0-*		report <sup>2</sup>
report <sup>2</sup>	0-*	Label	record

\* - many

Note: superscripts denote the level in nested depth.

Below is a sample of the XSD template. Each downloaded form will contain different specific elements, which are not listed below.:

```
<?xml version="1.0" encoding="utf-16"?>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="results">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="report">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="record">
              . . . .
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute name="label" type="xsd:string" />
          <xsd:attribute name="slug" type="xsd:string" />
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="exportTime" type="xsd:string" />
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Please see the XSD reference here: <http>

for more validation options that you can add to your XSD document.

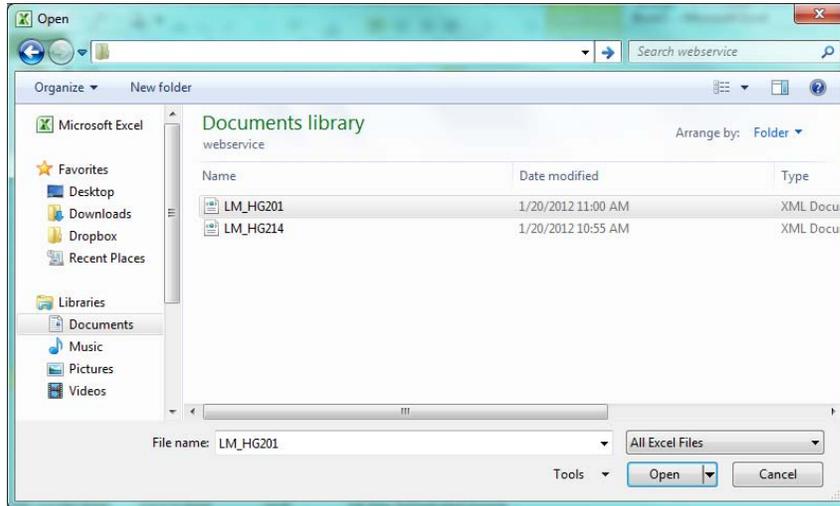
#### 4.5.2 Example of XML being consumed by Microsoft Excel 2010

The XML can be easily consumed by Microsoft Excel spreadsheet for very easy reading and converting to other formats and further analysis.

To do this, follow these steps:

1. Save the XML output to a file with an .xml file extension in a directory.

Exhibit 5 Saving and Opening the XML from MS Excel



2. Launch Microsoft Excel 2010 and open the file as seen above, Select “As an XML table” when the dialogue below pops up, and then click OK on the next dialogue, as we can allow Excel to create a schema if needed later:

Exhibit 6 MS Excel Open XML dialogue

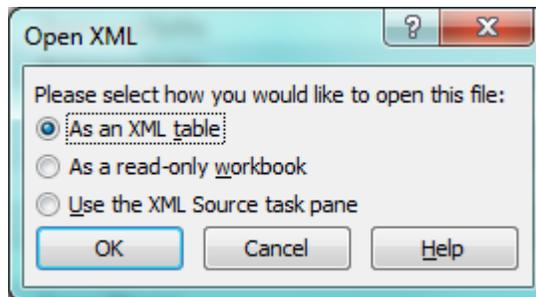
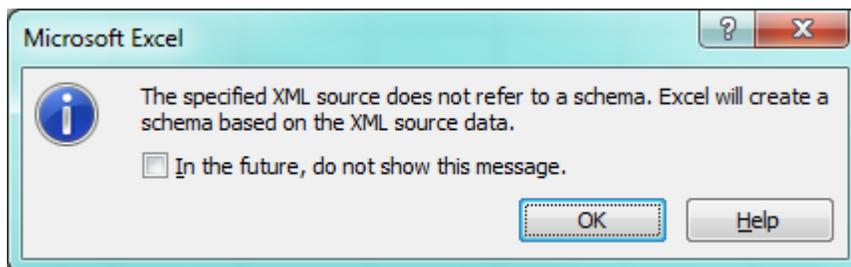


Exhibit 7 MS Excel schema creation prompt



3. You should now be able to view the results, pick a theme, sort or filter, export and do other analysis.

Exhibit 8 Example XML table in MS Excel showing sortable and filterable columns

	A	B	C	D	E	F	G	H	I
1	exportTime	label	slug	report_date	for_date_begin	narrative	label2	barrows_head_count	sows_head
2	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Summary	620,741	null
3	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Barrows/Gilts		
4	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Barrows/Gilts		
5	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Barrows/Gilts		
6	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Barrows/Gilts		
7	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Barrows/Gilts		
8	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Barrows/Gilts		
9	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Barrows/Gilts		
10	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Carcass Measurements		
11	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Sows/Boars		
12	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	Sows/Boars		
13	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		
14	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		
15	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		
16	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		
17	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		
18	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		
19	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		
20	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		
21	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		
22	2012-01-20 09:59:57 CST	National Daily Direct Hog Prior Day - Slaughtered Swine	LM_HG201	12/20/2010	12/17/2010	null	14-Day Scheduled Swine		

To save as CSV for example, you can use “Save As” to choose that format. (Due to the structure of the data, there can be empty sections and repeating sections so to prevent this, you may need to pre-process the data in Excel or select portions that you require).

Exhibit 9 MS Excel Save As dialogue with CSV selected

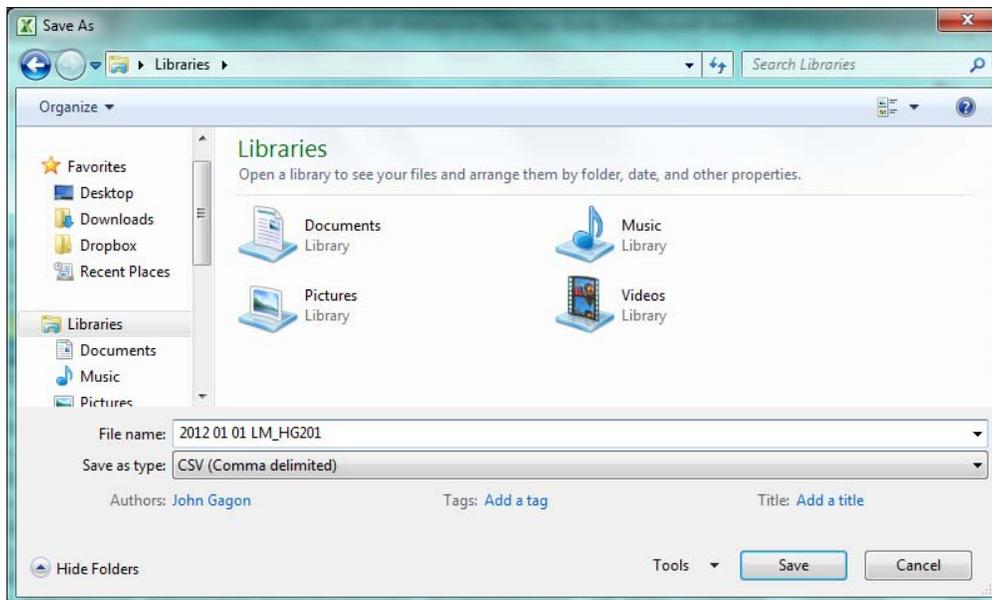
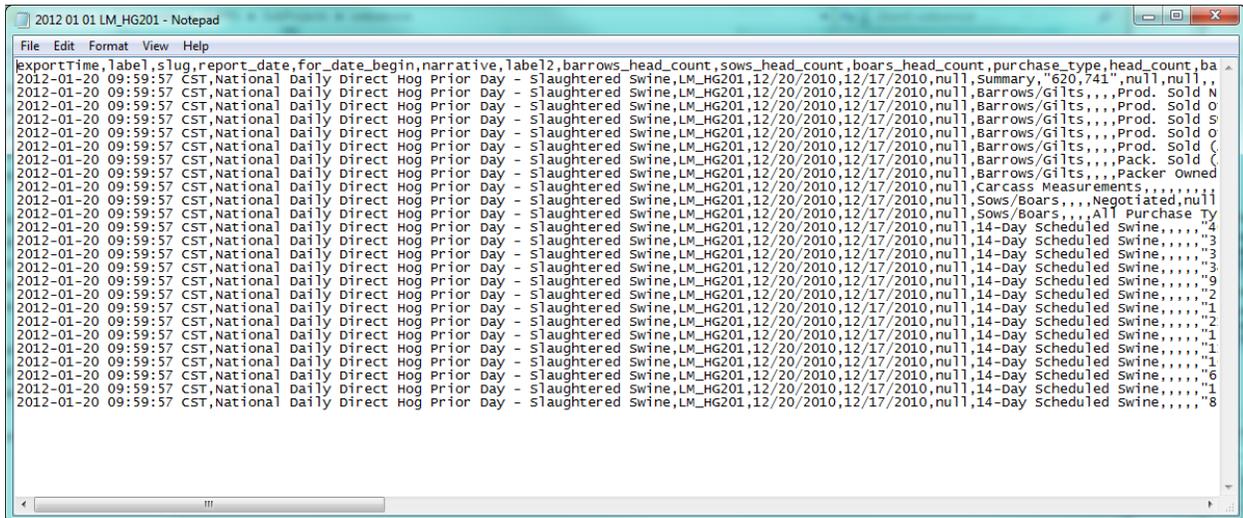
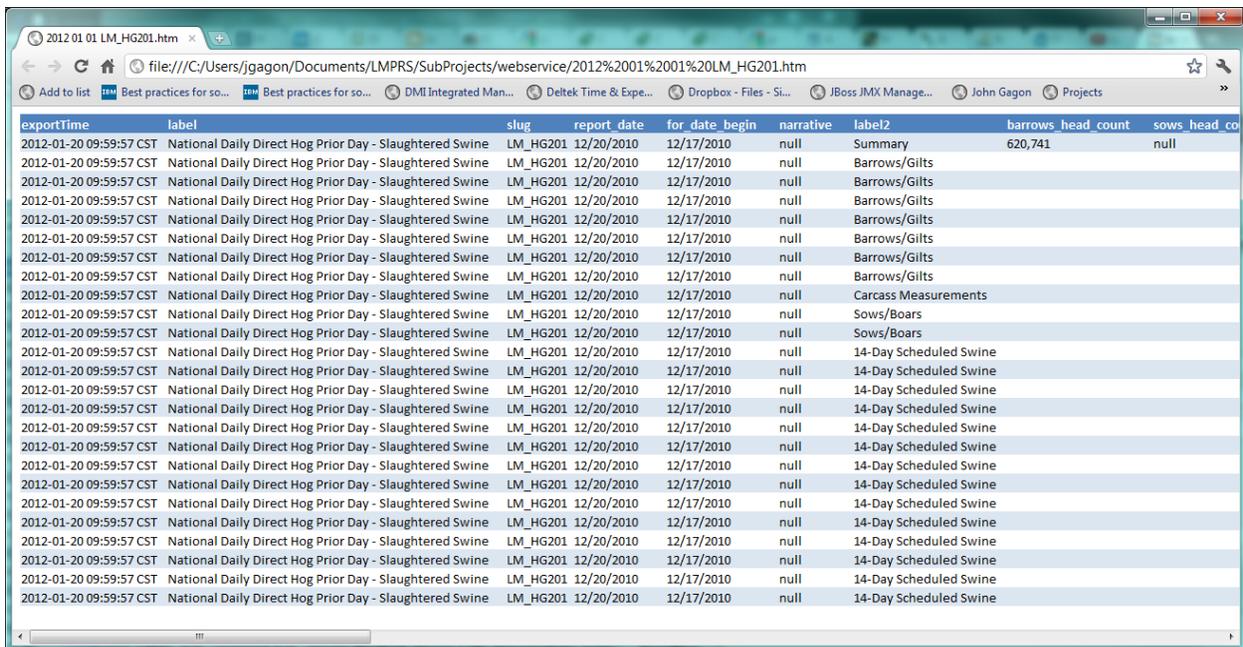


Exhibit 10 Sample CSV output in Notepad



You can also save the data to HTML while preserving the format:

Exhibit 11 Sample HTML output in Chrome



## 4.6 Sample Code for Querying the Service (Java)

In the following example, we use HttpClient from Apache Commons as well as Java's `javax.xml.stream.*` package. More examples on how to use the client are available on their website:

<http://hc.apache.org/httpcomponents-client-ga/>

<http://docs.oracle.com/javase/6/docs/api/javax/xml/stream/package-summary.html>

We go through a series of constructors to set values that will go into our request. We make the client a GET http method object that will send a GET request with query string. We add a query string to it with `NameValuePair`, but we could just as easily use a JSON library. The `%22`'s are just the double quotes in JSON, so we don't have to escape them all the time. You can use other means to URL encode with Java's standard encoders or formatting libraries as well. For simplicity, we show the final result in a simple string.

Then we make the call to execute this URL with `executeMethod(method)`, where the client will do its own network protocol to connect to a web server, and send the URL like a programmatic browser. We check for bad URL responses and then move on.

We want to handle the XML coming back, so here we start using the XML stream classes to get an instance of the `XMLEventReader`. We create some indentation so we can view our XML output more easily. For the sake of brevity, we are making this client to merely proof the URLs we want to test, and simply output the resulting response from the server. The reader uses a `hasNext()` to iterate through `XMLEvent` objects. Each `XMLEvent` will contain an element start tag, content or end tag. Our XML doesn't have content, so we simply loop through start and end tags and print attributes for start tags. We indent according to start tags and reduce indent when the end tag is encountered.

In the code sample above, the last thing we do is to close the network connection we have opened up to the web server. Using a `finally{...}` block to do so is highly recommended if you are performing significant work that could cause an exception or error to be thrown from the main block of code (`throws Exception`) to ensure this fairly safe but critical step is performed despite any errors. █

```
//Create an instance of the client
HttpClient client = new HttpClient();

//Initialize a GetMethod instance with the request.
GetMethod method = new GetMethod(
    "http://mpr.datamart.ams.usda.gov/ws/report/v1/cattle/LM_CT115");
method.getParams().setParameter(HttpMethodParams.RETRY_HANDLER,
    new DefaultHttpClientRetryHandler( 3, false));
method.setQueryString(new NameValuePair[] {new NameValuePair("filter",
    +"{\"fieldName\":\"Report Date\", \"
    +\"operatorType\":\"EQUAL\", \"
    +\"values\":[\"12/06/2010\"]}]"}));

//Execute the method passing it into the client and get the status code and print out any errors.
int statusCode = client.executeMethod(method);
if (statusCode != HttpStatus.SC_OK) {
```

```
        System.err.println("Method failed: " + method.getStatusLine());
        return;
    }

    //From a factory, get an Xml reader which will take the response from the method.
    //The executing of this method populates it with an Xml response.
    XMLInputFactory factory = XMLInputFactory.newInstance();
    XMLEventReader reader = factory.createXMLEventReader(method.getResponseBodyAsStream());
    String indent = "";

    //Loop events off the reader in the form of XMLEvent objects.
    while(reader.hasNext()) {

        XMLEvent event = reader.nextEvent();

        //The event object can be a start element or end element but a start element can have
        //attributes to read. We are just grabbing the objects and printing out their content to
        //System.out below in a format that shows a name and bracket list of name:value pairs.
        if (event.isStartElement()) {
            StartElement startElement = event.asStartElement();
            System.out.print(indent + startElement.getName().getLocalPart());

            //An element can have multiple attributes so we loop through those.
            Iterator<Attribute> attributes = startElement.getAttributes();
            while (attributes.hasNext()) {
                Attribute attribute = attributes.next();
                System.out.print(" ["
                    + attribute.getName() + ": "
                    + attribute.getValue() + "], ");
            }

            //Indent to form a tree structure for each start element and carriage return.
            indent += " ";
            System.out.println("\n");
        }else if(event.isEndElement()){
            //Indent back out when we are finished with one or more start elements.
            indent = indent.substring(2);
        }
    }

    //Clean up resources by releasing the connection when finished.
    method.releaseConnection();
```

## 4.7 Report Holidays

There are six national holidays that are usually observed when reports are not issued. Reports resume following these holidays. The observed dates do not follow actual dates for the holiday, but are a subset of observed holidays derived from the [U.S. OPM Federal Holiday schedule](#).

The six holidays normally selected from this schedule are as follows:

1. New Year's Day
2. Memorial Day
3. Independence Day
4. Labor Day
5. Thanksgiving Day
6. Christmas Day

Besides the holidays above, extenuating circumstances may also impact the dates when reports are issued.