



*United  
States  
Department  
of  
Agriculture*

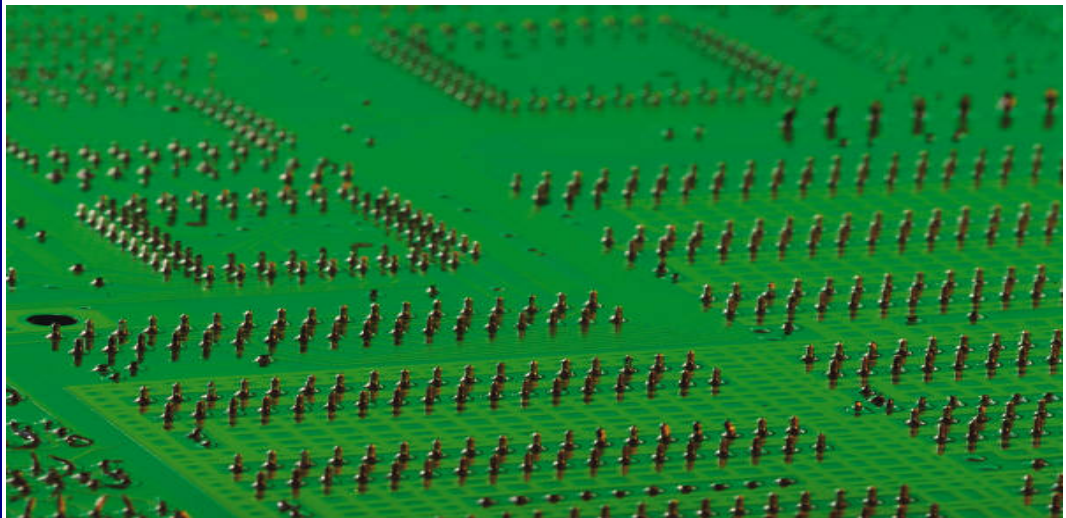
*Marketing  
and  
Regulatory  
Programs*

*Animal and  
Plant  
Health  
Inspection  
Service*

# ***National Animal Identification System (NAIS)***

**Technical Supplement to Draft Program Standards**

July 26, 2005



# TECHNICAL SUPPLEMENT TO DRAFT PROGRAM STANDARDS

<b>DATA STANDARDS .....</b>	<b>1</b>
ID NUMBER STANDARDS .....	1
PREMISES DATA STANDARDS – NATIONAL .....	1
PREMISES DATA STANDARDS – STATE/TRIBE .....	2
PREMISES FIELD DESCRIPTIONS .....	3
PREMISES BATCH FILE FORMAT .....	3
RECORD UPLOAD STANDARDS .....	4
<i>AIN/Animal Transaction Record</i> .....	4
<i>Group/Lot Movement Record</i> .....	6
<b>TECHNICAL STANDARDS.....</b>	<b>7</b>
CHECK DIGIT CALCULATION ALGORITHM – ISO 7064, MOD 37, 36 .....	7
<i>Formula for Calculating the Check Digit for an 18 Character Identifier</i> .....	7
RFID ISO STANDARDS .....	9
<i>ISO 11784 Radio Frequency Identification of Animals – Code Structure</i> .....	9
<i>ISO 11785 Radio Frequency Identification of Animals – Technical Concept</i> .....	10
<b>CODES .....</b>	<b>12</b>
OPERATION TYPE CODES .....	12
PREMISES RETIRED – REASON RETIRED CODES .....	12
SPECIES CODES .....	12
SEX CODES .....	13
ANIMAL EVENT CODES .....	13
GROUP/LOT EVENT CODES .....	14
BREED CODES (CAAB-NAAB) .....	14
<b>AIN WEB SERVICES.....</b>	<b>18</b>
AIN MANAGEMENT SYSTEM SCHEMA .....	18
WEB SERVICES JAVA CLASSES .....	23
<i>VerifyPremisesClient</i> .....	23
<i>RequestAinAllocationClient</i> .....	24
<i>ShipAinsClient</i> .....	25
<i>StateShipReportClient</i> .....	27

Date	Version	Changes
7/18/05	1.1	Removed AIN Allocation Data
7/26/05	1.2	Added AIN Web Services section

## DATA STANDARDS

### ID NUMBER STANDARDS

Premises ID Number			
Format	Length	Example	Comments
Alphanumeric	7 6 – Premises ID Number 1 – Check Digit	025B7HK	Right most character is a check digit <sup>1</sup> .

<sup>1</sup> See Technical Supplement B – Check Digit Calculation Algorithm

ID Numbers				
Data Element	Length	Format	Example	Comments
Premises ID Number	7	Alphanumeric	A123R69	Right most character is a check digit <sup>1</sup> .
Nonproducer Participant ID Number	7	Alphanumeric	H892345	Right most character is a check digit <sup>1</sup> .
Animal ID Number (AIN)	15	Numeric		
	[3]		840	First 3 digits are the country code (840 = USA)
	[12]		123456789012	Last 12 digits are animal number. Start number > 2,000,000,000
Group/Lot ID Number (GIN)	13	Alphanumeric		
	[7]		A234567	First 7 characters are the Premises ID Number.
	[6]		112205	Last 6 characters are the date the lot was established – MMDDYY.

<sup>1</sup> See Appendix B – Check Digit Calculation Algorithm

### PREMISES DATA STANDARDS – NATIONAL

Data Elements – National Premises Information Repository				
No.	Field Name	Format	Length	Example/Comments
1	Premises ID Number	Alphanumeric	7	025B7HK
2	Name of Entity	Alphanumeric	30	
3	Owner or Appropriate Contact Person <sup>1</sup>	Alphanumeric	30	
4	Street Address	Alphanumeric	30	

5	City	Alphanumeric	20	
6	State	Alphanumeric	2	
7	ZIP/Postal Code	Numeric	9	
8	Contact Phone Number	Alphanumeric	15	
9	Operation Type	Alphanumeric	1	<i>(See Technical Supplement C – Operation Type Codes)</i>
10	Date Activated	Date	8	YYYYMMDD
11	Date Retired	Date	8	YYYYMMDD
12	Reason Retired	Alphanumeric	1	<i>(See Technical Supplement C – Premises Retired – Reason Retired Codes)</i>

<sup>1</sup> The contact person should be the person the animal health official is to communicate with when performing a traceback (as determined by the entity).

## PREMISES DATA STANDARDS – STATE/TRIBE

Data Elements – State/Tribe Premises ID Database				
No.	Field Name	Format	Length	Example/Comments
<i>Data required to be uploaded to the NPIR</i>				
1	Premises ID Number	Alphanumeric	7	025B7HK
2	Name of Entity	Alphanumeric	30	
3	Owner or Appropriate Contact Person <sup>1</sup>	Alphanumeric	30	
4	Street Address	Alphanumeric	30	
5	City	Alphanumeric	20	
6	State	Alphanumeric	2	
7	ZIP/Postal Code	Numeric	9	
8	Contact Phone Number	Alphanumeric	15	
9	Operation Type	Alphanumeric	1	<i>(See Technical Supplement C – Operation Type Codes)</i>
10	Date Activated	Date	8	YYYYMMDD
11	Date Retired	Date	8	YYYYMMDD
12	Reason Retired	Alphanumeric	1	<i>(See Technical Supplement C – Premises Retired – Reason Retired Codes)</i>
<i>Data maintained in the State/Tribe Premises ID System</i>				
	Historic Data <sup>2</sup>			
13	Previous Contact Person	Alphanumeric	30	
14	Previous Contact Person Phone	Alphanumeric	15	
15	Previous Contact Person – Start Date	Date	8	YYYYMMDD

16	Previous Contact Person – End Date	Date	8	YYYYMMDD
	GPS			
17	Latitude	Numeric	2.6 <sup>3</sup>	
18	Longitude	Numeric	3.6 <sup>3</sup>	
19	Alternate Phone Number	Alphanumeric	15	

<sup>1</sup> The contact person should be the person with whom the Animal Health Official is to communicate when performing a traceback (as determined by the entity).

<sup>2</sup> Requires facility to store multiple records.

<sup>3</sup> GPS coordinates contain 2 (latitude) or 3 (longitude) digits to the left of the decimal point, and 6 digits to the right of the decimal point. They contain an optional sign character: - indicates south latitudes or west longitudes.

## PREMISES FIELD DESCRIPTIONS

Field Descriptions Submitted to Premises Number Allocator				
No.	Field Name	Format	Req'd	Example/Comments
1	Address Number	Alphanumeric	N	Primary address number Can contain 0-9, a-z/- and space
2	Pre Directional	Alphanumeric	N	Select S / N / E / W / SW / SE / NE / NW <sup>1</sup>
3	Street Name	Alphanumeric	Y	Can contain a-z and space
4	Street Suffix	Alphanumeric	N	Select ROAD / LANE / AVENUE / etc. <sup>1</sup>
5	Post Directional	Alphanumeric	N	Select S / N / E / W / SW / SE / NE / NW <sup>1</sup>
6	Secondary Address Identifier	Alphanumeric	N	Select APT / DEPT / FL / LOT / RM / STE / UNIT / # / etc. <sup>1</sup>
7	Secondary Address Range	Alphanumeric	N	Can contain 0-9, a-z/- and space
8	City	Alphanumeric	N	Can contain a-z and space
9	State	Alphanumeric	N	Select state code <sup>1</sup>
10	ZIP/Postal Code	Numeric	Y	Can contain 0-9 Secondary 4 digits are optional

<sup>1</sup> These fields are populated from drop-down menus. The selections are defined by USPS Publication 28: Postal Addressing Standards.

## PREMISES BATCH FILE FORMAT

Batch File Format for Obtaining Premises ID Numbers				
Description	Length	Begin	End	Comments
Unique Record Identifier	10	1	10	Unique identifier for record. Not used in batch processing. For client use only. (optional)
Address Line	60	11	70	Requested address line (required)
City	20	71	90	Requested city (required)

State	2	91	92	Requested state abbreviation ( <i>required</i> )
Zip5	5	93	97	Requested zip5 ( <i>required</i> )
Zip4	4	98	101	Requested zip4 ( <i>optional</i> )
<i>Once processed, the following will be appended to each line in the file</i>				
IsSuccessful	1	102	102	NOT successful = 0 Successful = 1
Short Message	30	103	132	Error message = PremisesNotFound <sup>1</sup>
Long Message	60	133	192	Error message description <sup>1</sup>
Address Line ( <i>normalized</i> ) <sup>3</sup>	60	193	252	Normalized address line <sup>2</sup>
City ( <i>normalized</i> ) <sup>3</sup>	20	253	272	Normalized city <sup>2</sup>
State ( <i>normalized</i> ) <sup>3</sup>	2	273	274	Normalized state abbreviation <sup>2</sup>
Zip5 ( <i>normalized</i> ) <sup>3</sup>	5	275	279	Normalized zip5 <sup>2</sup>
Zip4 ( <i>normalized</i> ) <sup>3</sup>	4	280	283	Normalized zip4 <sup>2</sup>
County ( <i>normalized</i> ) <sup>3</sup>	20	284	303	Normalized county <sup>2</sup>
County Code ( <i>normalized</i> ) <sup>3</sup>	5	304	308	Normalized county code <sup>2</sup>
Premises ID	6	309	314	Premises ID Number <sup>2</sup>
Check Sum	1	315	315	Premises ID check sum digit <sup>2</sup>
Is Premises New	1	316	316	Existing premises ID returned = 0 New premises ID created = 1 <sup>2</sup>
<sup>1</sup> Spaces if IsSuccessful = 1				
<sup>2</sup> Spaces if IsSuccessful = 0				
<sup>3</sup> For complete information on normalized addressing, refer to USPS Publication 28: Postal Addressing Standards.				

## RECORD UPLOAD STANDARDS

### AIN/Animal Transaction Record

The file naming convention for the AIN/Animal Transaction Record is defined as:

Nonproducer Participant ID Number + YYYYMMDDHHMMSS (time stamp) + .IND (file extension).  
For example: T234W6720050804112233.IND

Record format: Comma delimited, double quotes around fields that include a comma, terminate a record with an EOL character, terminate the file with an EOF character.

<b>AIN/Animal Transaction Record Format</b>				
<b>File: ID #1</b>				
<b>File Header Record (One Record)</b>				
No.	Field Name	Format	Size	Example
1	Nonproducer Participant ID Number	Alphanumeric	7	T234W67
2	Transmission Date	Numeric	12	YYYYMMDDHHMM

3	Record Count	Numeric	8	The number of records included in the file. Provides a check to verify that the file was sent in its entirety.
4	E-mail Address	Alphanumeric	60	E-mail address of the person to contact concerning errors when processing the file.

### Record Description (Multiple Records)

No.	Field Name	Format	Size	Req'd	Example
1	Event Type Code	Numeric	2	Y	<i>(See Technical Supplement C – Animal Event Codes)</i>
2	Sighting/Reporting Premises ID	Alphanumeric	7	Y	
3	Source/Destination Premises ID	Alphanumeric	7	N	
4	Event Date and Time	Numeric	12	Y	YYYYMMDDHHMM
5	AIN Used?	Boolean	1	Y	0 – False 1 – True (Field 6 required)
6	Animal ID Number	Numeric	15	Y	AIN <sup>1</sup>
7	Species	Alphanumeric	3	N	<i>(See Technical Supplement C – Species Codes)</i>
8	ID Electronically Read?	Boolean	1	Y	0 – False 1 – True
9	Animal Date of Birth	Alphanumeric	8	N	YYYYMMDD
10	Age of Animal	Alphanumeric	3	N	(D)ays, (M)onths, (Y)ears e.g. M11
11	Sex of Animal	Alphanumeric	1	N	<i>(See Technical Supplement C – Sex Codes)</i>
12	Breed of Animal	Alphanumeric	2	N	<i>(See Technical Supplement C – Breed Codes)</i>
13	Remarks	Alphanumeric	50	N	
14	Status	Alphanumeric	1	N	C – Correction
15	Alternate Animal ID 1	Alphanumeric	17	N	<sup>2</sup>
16	Alt Animal ID Type 1	Alphanumeric	1	N	<sup>3</sup>
17	Alternate Animal ID 2	Alphanumeric	17	N	<sup>4</sup>
18	Alt Animal ID Type 2	Alphanumeric	1	N	<sup>5</sup>

<sup>1</sup> Until AIN is the only approved animal ID identifier, other official ID numbers need to be reported in fields 15 through 18.

<sup>2</sup> Alternate pre-existing official ID number if AIN is not available; Group/Lot ID Number if animal has AIN and was moved out of a group/lot; old AIN if tag was replaced.

<sup>3</sup> A – American ID; B – Breed Registry Number; L – Lot Number; R – RFID; T – Tattoo; U – USDA eartag.  
E – Replacement AIN if event code 6 is used. This field is required if field 15 is used.

<sup>4</sup> Second alternate pre-existing official ID number if AIN is not available; Group/Lot ID Number if animal has AIN and was moved out of a group/lot.

<sup>5</sup> A – American ID; B – Breed Registry Number; L – Lot Number; R – RFID; T – Tattoo; U – USDA eartag. This field is required if field 17 is used.

**Group/Lot Movement Record**

The file naming convention for the Group/Lot Movement Record is defined as:

Nonproducer Participant ID Number + YYYYMMDDHHMMSS (time stamp) + .LOT (file extension).  
For example: T234W6720050804112233.LOT

Record format: Comma delimited, double quotes around fields that include a comma, terminate a record with an EOL character, terminate the file with an EOF character.

<b>Group/Lot Movement Record Format</b>					
<b>File: ID #2</b>					
<b>File Header Record (One Record)</b>					
<b>No.</b>	<b>Field Name</b>	<b>Format</b>	<b>Size</b>	<b>Example</b>	
1	Nonproducer Participant ID Number	Alphanumeric	7	T234W67	
2	Transmission Date	Numeric	12	YYYYMMDDHHMM	
3	Record Count	Numeric	8	The number of records included in the file. Provides a check to verify that the file was sent in its entirety.	
4	E-mail Address	Alphanumeric	60	E-mail address of the person to contact concerning errors when processing the file.	
<b>Record Description (Multiple Records)</b>					
<b>No.</b>	<b>Field Name</b>	<b>Format</b>	<b>Size</b>	<b>Req'd</b>	<b>Example</b>
1	Event Type Code	Numeric	2	Y	<i>(See Technical Supplement C – Animal Event Codes)</i>
2	Premises ID	Alphanumeric	7	Y	Required when Event Type Code is 2, 3, or 4.
3	Event Date and Time	Numeric	12	Y	YYYYMMDDHHMM
4	Group/Lot ID Number	Alphanumeric	13	Y	
5	Group/Lot Subset Identifier	Alphanumeric	30	N	Used to identify subset, such as barn.
6	Group Type	Alphanumeric	1	Y	S – Static D – Dynamic
7	Species	Alphanumeric	3	Y	<i>(See Technical Supplement C – Species Codes)</i>
8	Event Remark	Alphanumeric	50	N	
9	Status	Numeric	1	N	C – Correction

## TECHNICAL STANDARDS

### CHECK DIGIT CALCULATION ALGORITHM – ISO 7064, MOD 37, 36

The Check Digit algorithm referenced has been taken from ISO 7064:1983, Data Processing – Check Character Systems. It maps a string of alphanumeric characters to a single alphanumeric character.

#### Formula for Calculating the Check Digit for an 18 Character Identifier

Example: A1-2425G-ABC1234-002-x, where x is the check digit.

The characters of the identifier are processed character by character from left to right.

N=18 is defined as the number of characters including the check digit in the identifier. The characters of the identifier (including the check digit) are numbered from right to left: a<sub>1</sub> is the check digit and a<sub>2</sub> to a<sub>18</sub> are the characters of the identifier as follows.

A	1	2	4	2	5	G	A	B	C	1	2	3	4	0	0	2	x
a <sub>18</sub>	a <sub>17</sub>	a <sub>16</sub>	a <sub>15</sub>	a <sub>14</sub>	a <sub>13</sub>	a <sub>12</sub>	a <sub>11</sub>	a <sub>10</sub>	a <sub>9</sub>	a <sub>8</sub>	a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>

Table A-1

The algorithm then comprises five steps:

**Step 1:** Set a<sub>j</sub> for j = n...2 as follows:

a<sub>n</sub> is the value for the first character of the identifier. (See Table A-2)

a<sub>n-1</sub> is the value for the second character of the identifier

...

a<sub>2</sub> is the value for the last character of the identifier.

Char	Value	Char	Value	Char	Value	Char	Value
0	0	9	9	I	18	R	27
1	1	A	10	J	19	S	28
2	2	B	11	K	20	T	29
3	3	C	12	L	21	U	30
4	4	D	13	M	22	V	31
5	5	E	14	N	23	W	32
6	6	F	15	O	24	X	33
7	7	G	16	P	25	Y	34
8	8	H	17	Q	26	Z	35

Table A-2

**Step 2:** Set j=1 and P<sub>1</sub>=36

**Step 3:** Calculate

$$S_j = P_j \parallel_{37} + a_{(n-j+1)}$$

$$P_{(j+1)} = S_j \parallel_{36} \times 2$$

For  $j=1 \dots n$ , where

$\|_{36}$  is the remainder after division by 36. If the remainder equals zero, then  $\|_{36} = 36$ .

$|_{37}$  is the remainder after division by 37 (never equals to 0).

$a_{(n-j+1)}$  is value of a character in the string.

**Step 4:** The check digit  $a_1$  must be computed so that  $S_n \|_{36} = 1$ .

**Step 5:** Use Table A-2 to select the check character.

j	Char	$a_j$	$a_{(n-j+1)}$	$P_j$	$P_j _{37}$	$S_j$	$S_j \ _{36}$	$P_{j+1}$
1	A	10	10	36	36	46	10	20
2	1	1	1	20	20	21	21	42
3	2	2	2	42	5	7	7	14
4	4	4	4	14	14	18	18	36
5	2	2	2	36	36	38	2	4
6	5	5	5	4	4	9	9	18
7	G	16	16	18	18	34	34	68
8	A	10	10	68	31	41	5	10
9	B	11	11	10	10	21	21	42
10	C	12	12	42	5	17	17	34
11	1	1	1	34	34	35	35	70
12	2	2	2	70	33	35	35	70
13	3	3	3	70	33	36	36	72
14	4	4	4	72	35	39	3	6
15	0	0	0	6	6	6	6	12
16	0	0	0	12	12	12	12	24
17	2	2	2	24	24	26	26	52
18	M	22		52	15	37		

Table A-3

$S_{18}$  is defined as  $S_{18} = P_{18|37+a_1}$  ( $a_1$  being the check character). Hence, we must find an  $a_1$ , so that  $15 + a_1 - 1$  is dividable by 36 without rest.

This leads to  $a_1 = 22$ , which represents the character "M." Hence the complete "identifier" is A1-2425G-ABC1234-002-M.

The following table illustrates the generation of a check digit for a 6-character Premises ID Number (104G7M).

j	Char	$a_j$	$a_{(n-j+1)}$	$P_j$	$P_j _{37}$	$S_j$	$S_j _{36}$	$P_{j+1}$
1	1	1	1	36	36	37	1	2
2	0	0	0	2	2	2	2	4
3	4	4	4	4	4	8	8	16
4	G	16	16	16	16	32	32	64
5	7	7	7	64	27	34	34	68
6	M	22	22	68	31	53	17	34
7	3	3		34	34	37		

Table A-4

The check digit for this example is 3, resulting in a complete Premises ID Number of 104G7M3.

## RFID ISO STANDARDS

Radio Frequency Identification (RFID) is an electronic means of identification. The data gathering device is a transceiver, which may be stationary or portable. The transceiver transmits data to and receives data from a transponder, which is a device attached to the object being identified. The main components of the transponder are the antenna and the microchip. The antenna consists of a coil of wire or a coil etched on a substrate. The antenna serves to receive data from and send data to the transceiver, and functions with the transceiver to supply power to the transponder while data is gathered. Since the transponder is passive (contains no battery or other power source) the microchip is activated by energy from the transceiver. Upon activation, it receives data from the transceiver and responds by sending data, including the ID number embedded in the microchip's memory, to the transceiver.

RFID technology is used in the identification of a wide variety of objects. Its use in the identification of animals is governed by two international standards – ISO 11784 and ISO 11785. ISO 11784 defines the code structure of the ID data, which is embedded in the memory of the transponder's microchip. ISO 11785 defines the technical specifications of how the transceiver gathers data from the transponder.

### *ISO 11784 Radio Frequency Identification of Animals – Code Structure*

RF identification of animals requires that the bits transmitted by a transponder are interpretable by a transceiver. Usually the bit stream contains data bits, defining the identification code and a number of other bits to ensure correct reception of the data bits. ISO 11784 defines the data bits; ISO 11785 defines the data integrity bits.

#### *Definitions*

The following terms relate to RFID, and refer to RFID devices and the data contained in and communicated by them.

- Animal Code – Bit pattern to identify an animal.
- Bit Pattern – Sequence of binary digits or bits [0,1].
- Code Field – Group of bits in the identification code with a specific meaning.
- Country Code – Bit pattern to define the country where the transponder was issued.
- Data Block – Additional group of bits with a specific meaning.

- Flag – Single bit with a specific meaning.
- Identification Code – Part of the code that is used for identification (control codes such as header, trailer and checksum are excluded).
- Manufacturer’s Code – Bit pattern identifying the manufacturer of the transponder.
- National Identification Code – Code field with a unique number within a country.
- Transceiver – Device used to communicate with a transponder.
- Transponder – Device which transmits its stored information when activated by a transceiver and may be able to store new information.

#### *Description of Code Structure*

The code in the transponder is split up into a number of code fields, each with its own meaning. Each field is coded in natural binary with the high-order bit being leftmost. The structure of the code is specified in the table below. Bit number 1 in the code is the most significant bit (MSB), bit number 64 is the least significant bit (LSB).

The combination of country code and national identification code provides a unique worldwide identification number.

<b>Code Structure</b>			
<b>Bit No.</b>	<b>Information</b>	<b>Combinations</b>	<b>Description</b>
1	Flag for animal (1) or non-animal (0) application. <sup>1</sup>	2	This bit signals whether the transponder is used for animal identification or not. In all animal applications this bit shall be 1.
2 - 15	Reserved field.	16 384	Fourteen bits of code are reserved for future use.
16	Flag indicating the existence of a data block (1) or no data block (0).	2	This bit signals that additional data is to be received (e.g., physiological data, measured by a device which combines identification and monitoring). This bit shall be 1 if additional information is appended to the identification code, otherwise it shall be 0.
17 - 26	ISO 3166 numeric-3 country code.	1 024	Country codes from 900 to 998 may be used to refer to individual manufacturers of transponders. Country code 999 is used to indicate that the transponder is a test transponder and need not contain a unique identification number. The country code for the United States is 840.
27 - 64	National identification code. <sup>2</sup>	274 877 906 944	Unique number within a country. <sup>3</sup>

*Notes:*

<sup>1</sup> The method to distinguish between animal and non-animal applications using bit No. 1 allows the code structure to be recognized electronically. However, this requires that future standards on RF identification in other fields will adhere to this convention.

<sup>2</sup> The length of the national identification code was chosen to have enough combinations available for all animals in a large country. Moreover, the uniqueness of a code is expected to be maintained over thirty years.

<sup>3</sup> It is a national responsibility to ensure the uniqueness of the national identification code. If necessary, number series may be allocated to species and/or manufacturers, but this will not be standardized. Ideally every country should maintain a central database in which all issued codes are stored, together with a reference to the database where the information concerning the associated animal can be retrieved.

#### *ISO 11785 Radio Frequency Identification of Animals – Technical Concept*

ISO 11785 defines the technical aspects of how a transceiver communicates with a transponder, how a transponder is activated, and how the stored information is transferred to a transceiver. The standard

allows communication to be either half-duplex (permitting communication between transceiver and transponder in one direction only at a time) or full-duplex (permitting simultaneous communication between the transceiver and the transponder), and is designed to facilitate both method to be incorporated in a single transceiver.

The following table lists the applicable technical specifications for transceivers and transponders. In addition to the 64 data bits defined in ISO 11784, headers, trailers, error detection bits, and, for full-duplex, control bits are defined.

<b>Summary of the FDX and HDX Systems</b>		
<b>Parameter</b>	<b>FDX System</b>	<b>HDX System</b>
Activation frequency	134.2 kHz	134.2 kHz
Modulation	AM_PSK	FSK
Return frequencies	129.0 kHz to 133.2 kHz 135.2 kHz to 139.4 kHz	124.2 kHz 134.2 kHz (0)
Encoding	Modified DBP	NRZ
Bit rate	4 194 bit/s	7 762.5 bits/s (1) 8 387.5 bits/s (0)
Telegram structure:		
– Header	11	8
– Identification code	64	64
– Error detection code	16	16
– Trailer	24	24
– Control bits	13	–

## CODES

### OPERATION TYPE CODES

Operation Type	
Code	Description
B	Port of Entry
C	Clinic
E	Exhibition
L	Laboratory
M	Market/Collection Point
N	Nonproducer Participant
P	Production Unit <sup>1</sup>
Q	Quarantine Facility
R	Rendering
S	Abattoir
T	Tagging Site
<sup>1</sup> Includes Hunt Ranches, etc.	

### PREMISES RETIRED – REASON RETIRED CODES

Reason Premises Retired	
Code	Description
D	Developed (Operation terminated resulting from commercial development)
E	Error (Reported in error)
M	Merged
O	Sold
S	Split

### SPECIES CODES

Species			
Code	Description	Code	Description
AQU	Aquaculture	CER	Cervids
CLM	Clams	DEE	Deer
CRA	Crawfish	ELK	Elk

CTF	Catfish	EQU	Equine (Horses) <sup>1</sup>
MSL	Mussels	OVI	Ovine (Sheep)
OYS	Oysters	POR	Porcine (Swine)
SAL	Salmon	POU	Poultry
SBA	Striped Bass	CHI	Chickens
SHR	Shrimp	DUC	Ducks
SLP	Scallops	GEE	Geese
TIL	Tilapia	GUI	Guineas
TRO	Trout	PGN	Pigeon
BOV	Bovine (Bison and Cattle)	PHE	Pheasants
CAM	Camelid (Alpaca and Llama)	QUA	Quail
CAP	Caprine (Goats)	TUR	Turkeys

<sup>1</sup> Equine industry will expand as necessary

## SEX CODES

Animal Sex	
Code	Description
M	Male
F	Female
C	Neutered / castrated male
S	Neutered / spayed female
X	Mixed (used only in groups)

## ANIMAL EVENT CODES

Event Type	
Code	Description
1	Tag shipped – National AIN is shipped to a Nonproducer Participant or a premises.
2	Tag applied – National Animal ID tag is applied to an animal.
3	Moved in – Animal is moved into a premises.
4	Moved out – Animal is moved out of a premises.
5	Lost tag – New tag is applied to an animal that lost a tag and previous AIN is unknown.
6	Replaced tag or Re-tagged – New tag is applied to an animal that lost a tag and previous AIN is unknown.
7	Imported – Animal is imported into the U.S.
8	Exported – Animal is exported out of the U.S.
9	Sighting – Animal has a confirmed sighting at a location; no movement has occurred.

10	Slaughtered – Animal was sent to slaughter.
11	Died – Animal died of natural causes or euthanized at the farm/ranch.
12	Tag retired – Tag retired by producer, packing house, etc.
13	Animal missing – Lost, stolen, etc.
14	ICVI – Certificate of Veterinary Inspection

## GROUP/LOT EVENT CODES

Event Type	
Code	Description
1	Begin Group/Lot – Group/Lot of animals was established at a premises.
2	Moved Group/Lot in – Group/Lot of animals was moved into a premises.
3	Moved Group/Lot out – Group/Lot of animals was moved out of a premises.
4	Sighting – Group/Lot has a confirmed sighting at a location; no movement has occurred (i.e. vet sighting).
5	End Group-Lot – Group/Lot inventory is zero.

## BREED CODES (CAAB-NAAB)

Dairy Breeds					
Breed	Code	Breed	Code	Breed	Code
American Lineback	LD	Galloway	GD	Kerry	KY
Ayrshire	AY	Guernsey	GU	Red Holstein	WW
Brown Swiss	BS	Holstein	HO	Rouge Flamand	FM
Canadian Lineback	LK	Jersey	JE	Shorthorn	MS

Beef Breeds					
Breed	Code	Breed	Code	Breed	Code
Aberdeen Angus	AN	Dexter	DR	Nellore	NE
Abondance	AB	Dutch Belted	DL	Normande	NM
Africander	AF	East Flemish Red Pied	FP	Norwegian Red	NR
Alpine	AL	Eringer	ER	Parthenaise	PA
American Breed	AE	Flamande	FA	Pie Rouge	PR
Amerifax	AM	Fleckvieh	FL	Piedmontese	PI
Ankina	AK	Florida Cracker	FC	Pinzgauer	PZ
Ankole-Watusi	AW	Fribourg	FR	Ranger	RA
Aubrac	AU	Friesian (Belgium)	FB	Red Angus	AR

Barzona	BA	Friesian (Dutch)	DF	Red Brahman	RR
Beef Friesian	BF	Galloway	GA	Red Brangus	RB
Beefalo	BE	Gasconne	GS	Red Dane (Danish Red)	RD
Beefmaster	BM	Gelbray	GE	Red Poll	RP
Belgian Blue	BB	Gelbvieh	GV	Romagnola	RN
Belted Galloway	BG	Grauvieh	GI	Romosinuano	RS
Blonde d' Aquitaine	BD	Groningen	GR	Rotbunte	RO
Bonsmara	NS	Guzera	GZ	Rouge du Nord	DN
Braford	BO	Gyr (or Gir)	GY	Sahiwal	SW
Brahman	BR	Hays Converter	HC	Salers	SA
Brahmental	BH	Hereford (black)	HB	Santa Gertrudis	SG
Brahmousin	BI	Hereford (horned)	HH	Semepol	SL
Braler	BL	Hereford (polled)	HP	Senapol	SE
Brangus	BN	Highland (Scotch Highland)	SH	Shaver Beef Blend	SV
Braunveih	BU	Hybrid (Alberta Synthetic)	HY	Shothorn (beef-scotch)	SS
British White	BW	Indu Brazil	IB	Shothorn (Illawarra)	IS
Brown Swiss (beef/boeuf)	SB	Kerry	KY	Shorthorn (polled)	SP
Buelingo	BQ	Kobe (Wagyu)	KB	Simbrah	SI
Campine Red Pied	CP	Limousin	LM	Simmental	SM
Canadienne	CN	Lincoln Red	LR	South Devon	DS
Charbray	CB	Lowline (Loala)	LO	Sussex	SX
Charolais	CH	Luing	LU	Taba-pua	TB
Chi-Angus	CG	Maine-Anjou	MA	Tarentaise	TA
Chi-Maine	CM	Mashone	MH	Tasmanian Grey	TG
Chianina	CA	Mandalong Special	ML	Taurindicus	TN
Crossbreeds	XX	Marchiginana	MR	Texas Longhorn	TL
Crossbred Twinner	XT	Maremmana	ME	Tuli	TI
Cumberland	CU	Meuse-Rhine-Issel	MI	Welsh Black	WB
Danish Black and White	DB	Mexican Corriente	MC	West Flemish Red	WF
Danish Jersey	DJ	Montbeliard	MO	White Park	WP
Danish Red and White	RW	Murrah	MU	Yak	YA
Devon	DE	Murray Grey	MG		

## Swine Breeds

Breed	Code	Breed	Code	Breed	Code
-------	------	-------	------	-------	------

Berkshire	BK	Landrace	LA	Red Wattle	RW
Chester White	CW	Large Black (British)	LB	Spotted	SO
Duroc	DU	Large White	LW	Tamworth	TM
Hampshire	HA	Pietrain	PE	Wessex Saddleback	WS
Lacombe	LC	Poland China	PC	Yorkshire	YO

### Goat Breeds

Breed	Code	Breed	Code	Breed	Code
Alpine	AI	La Mancha	LN	Pygmy	PY
Angora	AG	Nigerian Dwarf	ND	Saanen	EN
Boer	BZ	Nubian	NU	Toggenburg	TO
Cashmere	CS	Oberhasli	OH		

### Sheep Breeds

Breed	Code	Breed	Code	Breed	Code
Arcott – Canadian	CD	Finnish Landrace	FN	Perendale	PE
Arcott – Outaouais	OU	Hampshire	HS	Polypay	PO
Arcott – Rideau	RI	Hybrid	HY	Rambouillet	RG
Barbados Black Belly	LY	Icelandic	IL	Romanov	RV
Black Face	FB	Ile de France	IF	Romnelet	RM
Black Welsh Mountain	BW	Jacob	JA	Romney	RY
Blue Faced Leister	BF	Karakul	KK	Rouge de l'Ouest	RO
Booroola	BO	Katahdin	KA	Ryeland	RL
Border Cheviot	BC	Kerry Hill	KH	St. Croix	SX
Charollais	CO	Lacaune Dairy Sheep	CU	Scottish Blackface	SC
Clun Forest	CF	Leicester – Border	BL	Shetland	SL
Columbia	CL	Leicester – English	LE	Shropshire	SR
Coopworth	CP	Leister – Hexam	HL	Southdown	ST
Corriedale	CR	Lincoln	LI	Suffolk	SU
Cottswold	CW	Merino	MM	Targhee	TA
Dorper	DO	Merino Polled	MP	Texel	TX
Dorset – Horned	DH	Montadale	MT	Tunis	TU
Dorset – Polled	DP	Newfoundland Loco	NL	Crossbred – Large	XL
DLS	DL	North Country Cheviot	NC	Crossbred – Medium	XM
Drysdale	DY	Oxford	OX	Crossbred – Small	XS
East Friesian	EF				

Bison Breeds					
Breed	Code	Breed	Code	Breed	Code
Wood Bison	WO	Plains Bison	PB		

Horse Breeds					
Breed	Code	Breed	Code	Breed	Code
American Bashkir Curly	AC	Friesian	FR	Peruvian	PV
American Saddlebred	AS	Gelderlander	GL	Pinto	PN
Andalusian	AA	German Warmblood	WG	Polish Warmblood	PW
Anglo-Arabian	AO	Hackney Horse	HN	Polo Pony	OL
Anglo-Normand	NO	Hackney Pony	HK	Quarter Horse	QH
Appaloosa	AP	Haflinger	HF	Rheinlander	RH
Arabian	AD	Hanovarian	HV	Rustic Pony	RU
Baden-Wurttemberg	BW	Hessen	HE	Shetland	SE
Bayerisches Warmblood	BY	Highland Pony	HG	Shire	SY
Belgian	GI	Holsteiner	HT	Standardbred	SN
Belgian Warmblood	GW	Hungarian Warmblood	HW	Suffolk Punch	SF
Buckskin	BU	Hunter (Sport Horse)	HU	Swedish Warmblood	WW
Canadian Horse	CI	Icelandic	IC	Swiss Horse	WI
Cheval de Selle Francais (French Saddle Horse)	FC	Lipizzaner	LZ	Swiss Warmblood	IW
Cleveland Bay	CV	Miniature Horse	MU	Tarpan	TP
Clydesdale	CY	Missouri Foxtrotting	MF	Tennessee Walking	TW
Connemara	CM	Morgan	MN	Thoroughbred	TH
Danish Warmblood	DW	New Forest	NF	Trakehner	TR
Dartmoor Pony	DT	Noriker	NK	Trotteur Francais	TF
Dutch Warmblood	DW	Oldenburg	OB	Viking	VK
Exmoor Pony	EX	Paint	PT	Welsh	WE
Fell Pony	FE	Palomino	PL	Westfalen	WF
Fjord	FJ	Paso Fino	PF	Wielkopolski (Polish Trakehner)	WR
French Horse	FH	Percheron	PH	Wurttemberg	WU

## AIN WEB SERVICES

There are four web services provided to State developers to access the AIN Management System:

- Validate PIN/NPN (ShipToInfoWS) – verify the validity of a Premises ID Number (PIN) or a Nonproducer Participant Number (NPN). The signature for this service is: **public ShipToInfoWS verifyPremises(String eAuthId, String premId) throws NAISException, RemoteException;**
- Request AIN Allocation (AinAllocationWS) – an AIN Tag Manufacturer request for AINs. The signature for this service is: **public AinAllocationWS requestAinAllocation(String eAuthId, String premId, Integer requestNum) throws RemoteException, NAISException;**
- Report AIN Shipment (AinShipmentWS) – Create an AIN Shipment record. The signature for this service is: **public void createAinShipmentEvent(String eAuthId, AinShipmentWS shipment, ShipToInfoWS shipTo, String shipDate) throws RemoteException, NAISException;**
- State Reports (StateProducerShipReportWS) – request a report of producers in a given state to which AINs have been shipped. The signature for this service is: **public StateProducerShipReportWS[] getStateProducerShipReport(String eAuthId, String stateName, Date startDate, Date endDate) throws RemoteException, NAISException;**

Client-side web services applications can be created by using either the client-side classes created by the application server or by generating your own classes using the WSDL file for the web services. Here are the links to obtain either (please note that these URLs point to our test instance):

- URL to get the test wsdl:

<http://ainmngt-test.aphis.usda.gov/ainmngmtwebservices/webServices?wsdl>

- URL to get the compiled client-side classes used to invoke the webservices:

[http://ainmngt-test.aphis.usda.gov/ainmngmtwebservices/webServices?proxy\\_jar](http://ainmngt-test.aphis.usda.gov/ainmngmtwebservices/webServices?proxy_jar)

The client-side code examples in the Web Services Java Classes section make use of the classes provided by this option. *Note: If you use this option, these other standard jar files will be required: soap.jar, mail.jar, and http\_client.jar.*

## AIN MANAGEMENT SYSTEM SCHEMA

The following code is the schema for accessing these web services.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions name="WSEntrySB"
  targetNamespace="http://gov.usda.aphis.nais.ws/WSEntrySB.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://gov.usda.aphis.nais.ws/WSEntrySB.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd1="http://gov.usda.aphis.nais.ws/WSEntrySB.xsd">

  <documentation>WSDL for Service: WSEntrySB, generated by Oracle WSDL
    toolkit (version: 1.1)</documentation>
```

```

<types>
  <schema targetNamespace="http://gov.usda.aphis.nais.ws/WSEntrySB.xsd"
    xmlns:tns="http://gov.usda.aphis.nais.ws/WSEntrySB.xsd"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/">

    <complexType
      name="gov_usda_aphis_nais_ws_model_AinAllocationWS">
      <all>
        <element name="endAin" type="xsd:long" />
        <element name="nppld" type="xsd:string" />
        <element name="startAin" type="xsd:long" />
      </all>
    </complexType>

    <complexType
      name="gov_usda_aphis_nais_ws_model_StateProducerShipReportWS">
      <all>
        <element name="ain" type="xsd:string" />
        <element name="producerName" type="xsd:string" />
        <element name="producerPrem" type="xsd:string" />
        <element name="shipDate" type="xsd:string" />
        <element name="sourceNPP" type="xsd:string" />
      </all>
    </complexType>

    <complexType
      name="ArrayOfGov_usda_aphis_nais_ws_model_StateProducerShipR
      eportWS">
      <complexContent>
        <restriction base="soapenc:Array">
          <attribute ref="soapenc:arrayType"
            wsdl:arrayType="tns:gov_usda_aphis_nais_ws_model_State
            ProducerShipReportWS[]" />
        </restriction>
      </complexContent>
    </complexType>

    <complexType
      name="gov_usda_aphis_nais_ws_model_AinShipmentWS">
      <all>
        <element name="mfgrBagNum" type="xsd:string" />
        <element name="numShipped" type="xsd:string" />
        <element name="prodCodes" type="tns:ArrayOfString" />
        <element name="sourceNPP" type="xsd:string" />
        <element name="startAin" type="xsd:string" />
      </all>
    </complexType>

    <complexType name="ArrayOfString">

```

```
<complexContent>
  <restriction base="soapenc:Array">
    <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]" />
  </restriction>
</complexContent>
</complexType>

<complexType name="gov_usda_aphis_nais_ws_model_ShipToInfoWS">
  <all>
    <element name="destCity" type="xsd:string" />
    <element name="destFirstName" type="xsd:string" />
    <element name="destLastName" type="xsd:string" />
    <element name="destNPP" type="xsd:string" />
    <element name="destPremisesName" type="xsd:string" />
    <element name="destState" type="xsd:string" />
    <element name="destStreet" type="xsd:string" />
    <element name="destZip4" type="xsd:string" />
    <element name="destZip5" type="xsd:string" />
  </all>
</complexType>

</schema>
</types>

<message name="requestAinAllocationInput">
  <part name="param0" type="xsd:string" />
  <part name="param1" type="xsd:string" />
  <part name="param2" type="xsd:int" />
</message>

<message name="requestAinAllocationOutput">
  <part name="output"
    type="xsd1:gov_usda_aphis_nais_ws_model_AinAllocationWS" />
</message>

<message name="verifyPremisesInput">
  <part name="param0" type="xsd:string" />
  <part name="param1" type="xsd:string" />
</message>

<message name="getStateProducerShipReportOutput">
  <part name="output"
    type="xsd1:ArrayOfGov_usda_aphis_nais_ws_model_StateProducerShip
    ReportWS" />
</message>

<message name="createAinShipmentEventInput">
  <part name="param0" type="xsd:string" />
  <part name="param1"
    type="xsd1:gov_usda_aphis_nais_ws_model_AinShipmentWS" />
  <part name="param2"
    type="xsd1:gov_usda_aphis_nais_ws_model_ShipToInfoWS" />
  <part name="param3" type="xsd:string" />
</message>
```

```

</message>

<message name="createAinShipmentEventOutput" />

<message name="verifyPremisesOutput">
  <part name="output"
    type="xsd:ArrayOfGov_usda_aphis_nais_ws_model_ShipToInfoWS" />
</message>

<message name="getStateProducerShipReportInput">
  <part name="param0" type="xsd:string" />
  <part name="param1" type="xsd:string" />
  <part name="param2" type="xsd:dateTime" />
  <part name="param3" type="xsd:dateTime" />
</message>

<portType name="WSEntrySBPortType">
  <operation name="requestAinAllocation">
    <input message="tns:requestAinAllocationInput" />
    <output message="tns:requestAinAllocationOutput" />
  </operation>
  <operation name="createAinShipmentEvent">
    <input message="tns:createAinShipmentEventInput" />
    <output message="tns:createAinShipmentEventOutput" />
  </operation>
  <operation name="verifyPremises">
    <input message="tns:verifyPremisesInput" />
    <output message="tns:verifyPremisesOutput" />
  </operation>
  <operation name="getStateProducerShipReport">
    <input message="tns:getStateProducerShipReportInput" />
    <output message="tns:getStateProducerShipReportOutput" />
  </operation>
</portType>

<binding name="WSEntrySBBinding" type="tns:WSEntrySBPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />

  <operation name="requestAinAllocation">
    <soap:operation soapAction="urn:gov-usda-aphis-nais-ws-
      WSEntrySB/requestAinAllocation" />
    <input>
      <soap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:gov-usda-aphis-nais-ws-WSEntrySB" />
    </input>
    <output>
      <soap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:gov-usda-aphis-nais-ws-WSEntrySB" />
    </output>
  </operation>

```

```
<operation name="createAinShipmentEvent">
  <soap:operation soapAction="urn:gov-usda-aphis-nais-ws-
    WSEntrySB/createAinShipmentEvent" />
  <input>
    <soap:body use="encoded"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:gov-usda-aphis-nais-ws-WSEntrySB" />
  </input>
  <output>
    <soap:body use="encoded"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:gov-usda-aphis-nais-ws-WSEntrySB" />
  </output>
</operation>

<operation name="verifyPremises">
  <soap:operation soapAction="urn:gov-usda-aphis-nais-ws-
    WSEntrySB/verifyPremises" />
  <input>
    <soap:body use="encoded"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:gov-usda-aphis-nais-ws-WSEntrySB" />
  </input>
  <output>
    <soap:body use="encoded"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:gov-usda-aphis-nais-ws-WSEntrySB" />
  </output>
</operation>

<operation name="getStateProducerShipReport">
  <soap:operation soapAction="urn:gov-usda-aphis-nais-ws-
    WSEntrySB/getStateProducerShipReport" />
  <input>
    <soap:body use="encoded"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:gov-usda-aphis-nais-ws-WSEntrySB" />
  </input>
  <output>
    <soap:body use="encoded"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:gov-usda-aphis-nais-ws-WSEntrySB" />
  </output>
</operation>
</binding>

<service name="WSEntrySB">
  <port name="WSEntrySBPort" binding="tns:WSEntrySBBinding">
    <soap:address location="http://ainmngt-
      test.aphis.usda.gov/ainmngmtwebservice/webServices" />
  </port>
</service>
</definitions>
```

## WEB SERVICES JAVA CLASSES

The following clients are offered as test examples for the web services.

### *VerifyPremisesClient*

```
package gov.usda.aphis.nais.ws.client;

import gov.usda.aphis.nais.ws.proxy.WSEntrySBProxy;
import gov.usda.aphis.nais.ws.proxy.gov_usda_aphis_nais_ws_model_ShipToInfoWS;

import java.io.BufferedReader;
import java.io.InputStreamReader;

/**
 * @author Scott Queen
 *
 * TODO Provide detail regarding this class.
 */
public class VerifyPremisesClient {

    public static void main(String[] args) {

        InputStreamReader inputStreamReader = new InputStreamReader ( System.in );
        BufferedReader stdin = new BufferedReader ( inputStreamReader );
        gov_usda_aphis_nais_ws_model_ShipToInfoWS premInfo;

        try {
            WSEntrySBProxy proxy = new WSEntrySBProxy();
            System.out.println("");
            System.out.println("What is your eAuth id? ");
            String eAuthId = stdin.readLine();
            System.out.println("What is the Non-producer participant/Producer id you want to verify? ");
            String premisesId = stdin.readLine();
            System.out.println("calling the webService verifyPremises....");
            premInfo = proxy.verifyPremises(eAuthId,premisesId);
            System.out.println("Premises Info:");
            System.out.println("NPN/PIN: " + premInfo.getDestNPP());
            System.out.println("Premises Name: " + premInfo.getDestPremisesName());
            System.out.println("Premises Contact: " + premInfo.getDestFirstName() + " " +
            premInfo.getDestLastName());
            System.out.println("Premises Street Address: " + premInfo.getDestStreet());
            if("".equals(premInfo.getDestZip4())) {
                System.out.println("Premises City/Zip: " + premInfo.getDestCity() + " " +
            premInfo.getDestZip5());
            } else {
                System.out.println("Premises City/Zip: " + premInfo.getDestCity() + " " +
            premInfo.getDestZip5() + " - " + premInfo.getDestZip4());
            }
        }
    }
}
```

```
    } catch (Exception e) {
        System.out.println("We caught a " + e.getClass() + " exception");
        e.printStackTrace();
    }
}
}
```

### *RequestAinAllocationClient*

```
package gov.usda.aphis.nais.ws.client;

import java.io.BufferedReader;
import java.io.InputStreamReader;

import gov.usda.aphis.nais.ws.proxy.WSEntrySBProxy;
import gov.usda.aphis.nais.ws.proxy.gov_usda_aphis_nais_ws_model_AinAllocationWS;
/**
 * @author Scott Queen
 *
 * Client code for testing the "Request AIN Allocation" web service.
 */
public class RequestAinAllocationClient {

    public static void main(String[] args) {

        InputStreamReader inputStreamReader = new InputStreamReader ( System.in );
        BufferedReader stdin = new BufferedReader ( inputStreamReader );

        try {
            WSEntrySBProxy proxy = new WSEntrySBProxy();
            System.out.println("");
            System.out.println("What is your eAuth id? ");
            String eAuthId = stdin.readLine();
            System.out.println("What is your Non-producer participant id? ");
            String nppId = stdin.readLine();
            System.out.println("How many AINs are you requesting? ");
            String num = stdin.readLine();
            Integer numberWanted = new Integer(num);
            System.out.println("calling the webService requestAinAllocation....");
            gov_usda_aphis_nais_ws_model_AinAllocationWS ain = proxy.requestAinAllocation(eAuthId,
nppId, numberWanted);
            System.out.println("The allocation was successful: ");
            System.out.println("Starting AIN: " + ain.getStartAin() + "; Ending AIN: " + ain.getEndAin() +
"; Requesting npp: " + ain.getNppId());
        } catch (Exception e) {
            System.out.println("We caught a " + e.getClass() + " exception");
            e.printStackTrace();
        }
    }
}
```

```
}
```

### *ShipAinsClient*

```
package gov.usda.aphis.nais.ws.client;

import gov.usda.aphis.nais.ws.proxy.WSEntrySBProxy;
import gov.usda.aphis.nais.ws.proxy.gov_usda_aphis_nais_ws_model_AinShipmentWS;
import gov.usda.aphis.nais.ws.proxy.gov_usda_aphis_nais_ws_model_ShipToInfoWS;

import java.io.BufferedReader;
import java.io.InputStreamReader;

/**
 * @author Scott Queen
 *
 * Client code for testing the "Ship Ains" web service.
 */
public class ShipAinsClient {

    public static void main(String[] args) {

        InputStreamReader inputStreamReader = new InputStreamReader ( System.in );
        BufferedReader stdin = new BufferedReader ( inputStreamReader );
        String shipDate = "";

        try {
            WSEntrySBProxy proxy = new WSEntrySBProxy();
            gov_usda_aphis_nais_ws_model_AinShipmentWS shipInfo = new
gov_usda_aphis_nais_ws_model_AinShipmentWS();
            gov_usda_aphis_nais_ws_model_ShipToInfoWS destInfo = new
gov_usda_aphis_nais_ws_model_ShipToInfoWS();
            System.out.println("");
            System.out.println("What is your eAuth id? ");
            String eAuthId = stdin.readLine();
            System.out.println("What is your Non-producer participant id? ");
            shipInfo.setSourceNPP(stdin.readLine());
            System.out.println("What was the ship date? (DD-MMM-YYYY): ");
            shipDate = stdin.readLine();
            System.out.println("What is the destination Non-producer participant/Producer id? ");
            String premisesId = stdin.readLine();
            destInfo.setDestNPP(premisesId);
            System.out.println("You can specify the ship-to information (e.g. contact, address) or you can
have that information populated by a web service.");
            System.out.println("Enter S to specify the ship-to info; enter W to use the web service to pre-
populate the ship-to info.");
            String method = stdin.readLine();
            if ("W".equals(method)) {
                System.out.println("calling the webService verifyPremises....");
            }
        }
    }
}
```

```
gov_usda_aphis_nais_ws_model_ShipToInfoWS premInfo = new
gov_usda_aphis_nais_ws_model_ShipToInfoWS();
    premInfo = proxy.verifyPremises(eAuthId,premisesId);
    destInfo.setDestFirstName(premInfo.getDestFirstName());
    destInfo.setDestLastName(premInfo.getDestLastName());
    destInfo.setDestStreet(premInfo.getDestStreet());
    destInfo.setDestCity(premInfo.getDestCity());
    destInfo.setDestState(premInfo.getDestState());
    destInfo.setDestZip5(premInfo.getDestZip5());
    destInfo.setDestZip4(premInfo.getDestZip4());
} else {
    System.out.println("What is the destination premises name? ");
    //destInfo.setDestPremisesName(stdin.readLine());
    System.out.println("What is the first name of the destination contact? ");
    destInfo.setDestFirstName(stdin.readLine());
    System.out.println("What is the last name of the destination contact? ");
    destInfo.setDestLastName(stdin.readLine());
    System.out.println("What is the destination street address? ");
    destInfo.setDestStreet(stdin.readLine());
    System.out.println("What is the destination city? ");
    destInfo.setDestCity(stdin.readLine());
    System.out.println("What is the destination state (two-letter code)? ");
    destInfo.setDestState(stdin.readLine());
    System.out.println("What is the destination Zip5 value? ");
    destInfo.setDestZip5(stdin.readLine());
    System.out.println("What is the destination Zip4 value? ");
    destInfo.setDestZip4(stdin.readLine());
}
while(true) {
    System.out.println("Did you ship individual AINs or a bag? (Ind | Bag)");
    String indOrBag = stdin.readLine();
    if("Ind".equals(indOrBag)) {
        System.out.println("What is the starting AIN? ");
        shipInfo.setStartAin(stdin.readLine());
        System.out.println("How many did you ship? ");
        shipInfo.setNumShipped(stdin.readLine());
        break;
    } else if("Bag".equals(indOrBag)) {
        System.out.println("What is the Manufacturer's Bag number? ");
        shipInfo.setMfgrBagNum(stdin.readLine());
        break;
    } else {
        System.out.println("I don't understand \"" + indOrBag + "\". I only understand Ind or Bag.
Please try again.");
    }
}
String[] prodCodes = new String[1];
System.out.println("Enter product code (can be found via the 'List ID Devices' link on the web
app; use the 'NAIS Device Code#' column): ");
prodCodes[0] = stdin.readLine();
shipInfo.setProdCodes(prodCodes);
```

```
        System.out.println("calling the webService createAinShipmentEvent...");
        proxy.createAinShipmentEvent(eAuthId, shipInfo, destInfo, shipDate);
        System.out.println("The shipment event was successfully recorded.");
    } catch (Exception e) {
        System.out.println("We caught a " + e.getClass() + " exception");
        e.printStackTrace();
    }
}
}
```

### *StateShipReportClient*

```
package gov.usda.aphis.nais.ws.client;

/*
 * NOTE: the WSEntrySBProxy class is generated by the Oracle 10g Application Server and is found
 * in the webServices.zip file provided by that application server. The WSEntrySBProxy class contains
 * the SOAP address location so it's not necessary to define it in this code. As the web services move
 * to other servers (e.g. production) a new webServices.zip/WSEntrySBProxy class will have to
 * be generated.
 */

import gov.usda.aphis.nais.ws.proxy.WSEntrySBProxy;
import gov.usda.aphis.nais.ws.proxy.gov_usda_aphis_nais_ws_model_StateProducerShipReportWS;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Date;

/**
 * @author Scott Queen
 *
 * Client code for testing the "State Ship Report" web service.
 *
 * This application prompts the user for the information required to generate a report
 * of AINs shipped to the state during a given time period.
 */
public class StateShipReportClient {

    public static void main(String[] args) {
        InputStreamReader inputStreamReader = new InputStreamReader ( System.in );
        BufferedReader stdin = new BufferedReader ( inputStreamReader );

        try {
            WSEntrySBProxy proxy = new WSEntrySBProxy();
            System.out.println("");
            System.out.println("What is your eAuth id? ");
        }
    }
}
```

```
String eAuthId = stdin.readLine();
System.out.println("For what state do you want a report? (use two-letter code) ");
String stateName = stdin.readLine();
System.out.println("Report start date (DD-MMM-YYYY): ");
Date startDate = new Date(stdin.readLine());
System.out.println("Report end date (DD-MMM-YYYY): ");
Date endDate = new Date(stdin.readLine());
System.out.println("calling the webService getStateProducerShipReport....");

/*
 * Note: the gov_usda_aphis_nais_ws_model_StateProducerShipReportWS class is generated by
the Oracle 10g Application
 * Server and is bundled in the webServices.zip file produced by the application server. As you
can see from the print
 * statement, the class contains the following getter methods for obtaining the information
returned from the web services:
 * getAin(), getShipDate(), and getProducerPrem().
 */
gov_usda_aphis_nais_ws_model_StateProducerShipReportWS[] events =
proxy.getStateProducerShipReport(eAuthId, stateName, startDate, endDate);
System.out.println("Results:");
for(int i = 0; i < events.length; i++) {
    System.out.println("AIN: " + events[i].getAin() + "; Date: " + events[i].getShipDate() + ";
Prod. Prem: " + events[i].getProducerPrem());
}
} catch (Exception e) {
    System.out.println("We caught a " + e.getClass() + " exception");
    e.printStackTrace();
}
}
}
```