Logistical Operations Service Tracker
(LOST)

Product Requirements and Design Document

February 2, 2017
Executive Summary

OSNAP has a significant number of declassified and classified assets that must be accounted for and audited in accordance with federal laws and regulations. The current practice of directorate level spreadsheets and databases has resulted in failed audits and assets silently going missing while in transit. The Logistical Operations Service Tracker (LOST) will unify asset tracking and management across all OSNAP divisions, reducing incidents of property loss and asset management overhead.

LOST provides a single point of truth for where OSNAP assets are located and associated ownership. Assets will be tracked at rest and periodically while in motion. Due to the classified nature of some OSNAP assets, LOST will implement access controls to limit data visibility based on the nature of the data and the requesting user.
Document Versioning

12/28/2016 DE - Initial version
01/09/2017 DE - Updates for prerelease to development team
01/16/2017 DE - Initial datamodel added
01/27/2017 DE - Added missing lines in ERD
02/01/2017 DE - Added configuration file documentation
02/02/2017 DE - Significant updates to webservice API section
Project Description

Asset management across a large geographically distributed organization presents a significant management challenge. In the past, individual divisions have been independently responsible for asset management and tracking. Independent asset tracking will no longer scale for the needs of OSNAP and thus a large scale centralized asset management system is required. The Logistical Operations Service Tracker (LOST) will fill this requirement and enable OSNAP to safely continue to scale out.

OSNAP personnel needing access to LOST will have LOST access added to their WHO account. WHO will provide LOST with user information including username, person name, and division. LOST will internally maintain other user information as needed to support LOST functions.

Many OSNAP assets are classified and visibility into the type and location of those assets must be tightly controlled. LOST will support mandatory access control (MAC) to protect asset information (e.g. asset type and location). Since many assets must be hidden in plain sight (e.g. shipments on public highways), plausible substitute information in some cases will be shown to users with insufficient clearance. The special user role of 'classifier' will be able to change the classification of assets within LOST. The special user role of 'assigner' will be able to change the clearance of users. LOST must enforce that no user is both a classifier and an assigner.

Assets are instances of products. Product information can be used to look up plausible substitutions for an asset within a report. An asset may also include an explicit substitution. Each asset carries a classification, since an otherwise declassified product (e.g. a note pad) may become classified based on how it is used. Products also carry a classification and an asset may not carry a classification that is incompatible with the backing product.

Assets will be associated with a facility while at rest. While in transit, an asset will be associated with a travel request. The travel request will include the starting, ending, and last known location of the asset as well as information regarding the convoy. In addition to the current asset location, the location history of an asset can be reported using LOST.

Existing OSNAP asset data exists in a plethora of division specific data sources. The existing asset data will need to be in LOST at the time each division is cut over. Operators will use data migration scripts to support this activity.
Workflows

This section describes several of the workflows LOST will help to automate.

Travel Management

LOST will provide visibility into when and where assets are moved through the Travel Management workflow. This workflow will enforce OSNAP asset management policy.

Workflow Steps

1. A user in logistics creates a travel request and includes the assets to be transferred, the request is in the ‘unapproved’ state.

2. The request is then forwarded to management for approval. All managers must approve for the travel request to move forward, the request moves to the ‘approved’ state. Logistics and facilities managers may not approve their own requests.

   (a) Logistics manager at the source
   (b) Logistics manager at the destination
   (c) Facilities manager at the source
   (d) Facilities manager at the destination

3. Logistics manager at the source facility adds the convoy transporting the assets to the travel request. The asset is now associated with both the facility and the convoy. The request is now in the ‘ready’ state.

4. After loading onto the convoy, a user in logistics updates the request state to ‘in transit’ and the asset is disassociated from the source facility and associated with the destination facility.

5. While in transit, logistics users associated with the convoy periodically update the request with current location and other notes.

6. When the asset is received at the destination facility, a facilities manager associated with the destination facility moves the travel request into the ‘received’ state. The asset is disassociated with the convoy.

Asset Location History

LOST will provide features to reconstruct a time ordered location history of an asset.
Workflow Steps

1. A Logistics Manager or Facility Manager selects an asset and make a request for asset history.

2. The report request is logged for auditing.

3. Logistics Manager approves the request. Logistics managers may not approve their own requests.

4. The approved request is logged for auditing.

5. The requesting logistics or facilities manager opens the report in LOST. Opening the report is logged.

Additional Notes

An asset history report may only be viewed once. If the request approval causes LOST to generate and store the report internally, the report must be deleted once opened.

A logistics and facilities managers should only be able to request history for assets currently located at facilities they are associated with.

Logistics and facilities managers should only be able to see the facilities they associated with. Facilities that a logistics or facilities manager is not associated with should be omitted in the report. For example:

An asset starts at $A$, is moved to $B$, and ends at $C$, $A \rightarrow B \rightarrow C$. A Facilities manager associated only with facilities $A$ and $C$ would get a report indicating $A \rightarrow C$. Similarly, a facilities manager associated only with facilities $B$ and $C$ would get a report indicating $B \rightarrow C$. A facilities manager not associated with $C$ should not be able to see the asset to request the report on.

Asset Inventory Report

LOST will provide features to report on currently housed assets.

Workflow Steps

1. A Logistics Manager or Facility Manager selects a facility and requests a report on assets currently located there.

2. The report request is logged for auditing.

3. Facilities Manager approves the request. Facilities managers may not approve their own requests.

4. The approved request is logged for auditing.

5. The requesting logistics or facilities manager opens the report in LOST. Opening the report is logged.
Additional Notes

An asset inventory report may only be viewed once. If the request approval causes LOST to generate and store the report internally, the report must be deleted once opened.

Logistics and facilities managers should only be able to request asset inventories for facilities they are associated with.

Logistics and facilities managers should only be able to see assets for which they have appropriate privileges. Assets the user does not have sufficient privilege for should be omitted if a substitution has not been given for the asset. For example: An asset $I$ is in security compartments $A_{ts}$, $B_{ts}$, and $C_u$. A user with $A_{ts}$, $B_{ts}$, and $C_u$ will see the asset in the report since they have the correct compartments and sufficient clearance. A user with $A_{ts}$, $B_{ts}$, and $C_{ts}$ will see the asset in the report since they have the correct compartments and sufficient clearance. A user with compartments $A_s$, $B_u$, and $C_s$ should not see the asset $I$ since $A_{ts}$ exceeds their clearance. A user with compartments $A_{ts}$, $B_{ts}$, and $D_{ts}$ will not be able to see the asset since they are missing a required compartment.
User Stories

This section describes several of the user stories that are expected to be covered by the Logistical Operations Service Tracker (Lost). User stories are ordered by user type and high level task. Each user story provides a high level user objective and possible user interface process to complete the task.

Classifier

Change Asset Compartments

Objective
Classifier is changing the compartments associated with an asset.

Experience
Classifier navigates to the asset using the LOST UI. After selecting the asset, Classifier chooses to edit the asset properties. Classifier is able to change (add and remove) compartments the asset is associated with. Classifier is prompted to save changes before they take effect.

Change Product Compartments

Objective
Classifier is changing the compartments associated with a product.

Experience
Classifier navigates to the product using the LOST UI. After selecting the product, Classifier chooses to edit the product properties. Classifier is able to change (add and remove) compartments the product is associated with. If the Classifier assigns compartments that are incompatibly with an existing asset, a warning is displayed. Classifier is prompted to save changes before they take effect.
Assigner

Change User Compartments

Objective
Assigner is changing the compartments a user has access to.

Experience
Assigner navigates to the user using the LOST UI. After selecting the user, the assigner is able to change (add and remove) compartments the user is associated with. Assigner is prompted to save changes before they take effect.

Change User Facilities

Objective
Assigner is changing the facilities a user has access to.

Experience
Assigner navigates to the user using the LOST UI. After selecting the user, the assigner is able to change facilities and vehicles the user is associated with. Assigner is prompted to save changes before they take effect.

Logistics User

View Travel Requests

Objective
Logistics User wants to view travel requests needing service.

Experience
Logistics logistics dashboard lists the travel requests this user might be able to assist with. The user selects a travel request and is shown the details. User updates the request.

Request Transfer

Objective
Logistics User is moving an asset.

Experience
Logistics User selects ‘new travel request’ from the logistics dashboard. The user selects the source facility and assets to be transported then saves the request.
Start Transfer

Objective
Logistics User is noting that an asset has started motion.

Experience
After opening the travel request, the Logistics user changes the travel request state to loaded.

Update Transfer

Objective
Logistics User is noting an intermediate change in asset location.

Experience
After opening the travel request, the Logistics user updates the location to note that the asset has moved to a particular intermediate location.

Receive Transfer

Objective
Logistics User is noting arrival of an asset.

Experience
After opening the travel request, the Logistics user clicks ‘transfer complete’ to note that the asset has been received.

Logistics Manager

LM Approve Travel Requests

Objective
Logistics approves a new travel requests.

Experience
Logistics dashboard lists the new travel requests. The user selects a travel request and is shown the details. User selects approve. User then uses the UI to assemble the convoy to move the asset.
Facility Manager

FM Approve Travel Requests

Objective
Facilities approves a new travel requests.

Experience
Facilities dashboard lists the new travel requests. The user selects a travel request and is shown the details. User selects approve.
LOST concerns itself with three main ideas: assets, users, and security tags. Assets are the entities that LOST has been designed to track. Users represent OSNAP staff that are involved in managing assets. Security tags are used to provide mandatory access control since there are limitations on which security compartments users are allowed to view or manage.

Several join tables appear in the model. These join tables are needed to handle the many-to-many relationships between entities.

### Asset Tables

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>product_pk</td>
<td>integer</td>
<td>primary key for a product instance</td>
</tr>
<tr>
<td>vendor</td>
<td>text</td>
<td>who sells this product</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>description of the asset</td>
</tr>
<tr>
<td>alt_description</td>
<td>text</td>
<td>alternate description for the asset</td>
</tr>
</tbody>
</table>
assets

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>asset_pk</td>
<td>integer</td>
<td>primary key for an asset instance</td>
</tr>
<tr>
<td>product_fk</td>
<td>integer</td>
<td>id for the product instance the asset was spawned from</td>
</tr>
<tr>
<td>asset_tag</td>
<td>text</td>
<td>stick or engraved id used for inventory tracking</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>description of the asset</td>
</tr>
<tr>
<td>alt_description</td>
<td>text</td>
<td>alternate description for the asset</td>
</tr>
</tbody>
</table>

vehicles

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>vehicle_pk</td>
<td>integer</td>
<td>primary key for an asset instance</td>
</tr>
<tr>
<td>asset_fk</td>
<td>integer</td>
<td>id for the associated asset record</td>
</tr>
</tbody>
</table>

Vehicles are a type of asset but are special since they can be used to transport other assets. Table inheritance in Postgres could be used to do this a little more gracefully but that is an out of scope feature.

facilities

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>facility_pk</td>
<td>integer</td>
<td>primary key for a facility instance</td>
</tr>
<tr>
<td>fcode</td>
<td>text</td>
<td>facilities code used to identify the facility (6 or less characters)</td>
</tr>
<tr>
<td>common_name</td>
<td>text</td>
<td>common name for the facility</td>
</tr>
<tr>
<td>location</td>
<td>text</td>
<td>addressing information for the facility</td>
</tr>
</tbody>
</table>

asset_at

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>asset_fk</td>
<td>integer</td>
<td>asset at a facility</td>
</tr>
<tr>
<td>facility_fk</td>
<td>integer</td>
<td>facility the asset is at</td>
</tr>
<tr>
<td>arrive_dt</td>
<td>timestamp</td>
<td>when the asset arrived</td>
</tr>
<tr>
<td>depart_dt</td>
<td>timestamp</td>
<td>when the asset left</td>
</tr>
</tbody>
</table>

convoys

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>convoy_pk</td>
<td>integer</td>
<td>primary key for a convoy instance</td>
</tr>
<tr>
<td>request</td>
<td>text</td>
<td>request identifier for the convoy</td>
</tr>
<tr>
<td>source_fk</td>
<td>integer</td>
<td>source facility</td>
</tr>
<tr>
<td>dest_fk</td>
<td>integer</td>
<td>destination facility</td>
</tr>
<tr>
<td>depart_dt</td>
<td>timestamp</td>
<td>when the asset departed</td>
</tr>
<tr>
<td>arrive_dt</td>
<td>timestamp</td>
<td>when the asset arrived</td>
</tr>
</tbody>
</table>

used_by

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>vehicle_fk</td>
<td>integer</td>
<td>vehicle participating in a convoy</td>
</tr>
<tr>
<td>convoy_fk</td>
<td>integer</td>
<td>convoy vehicle participates in</td>
</tr>
</tbody>
</table>

asset_on

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>asset_fk</td>
<td>integer</td>
<td>asset at a facility</td>
</tr>
<tr>
<td>convoy_fk</td>
<td>integer</td>
<td>convoy the asset is on</td>
</tr>
<tr>
<td>load_dt</td>
<td>timestamp</td>
<td>when the asset was loaded</td>
</tr>
<tr>
<td>unload_dt</td>
<td>timestamp</td>
<td>when the asset was unloaded</td>
</tr>
</tbody>
</table>
User Tables

users

<table>
<thead>
<tr>
<th>user_pk</th>
<th>integer</th>
<th>primary key for a user instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>text</td>
<td>login name used by the user</td>
</tr>
<tr>
<td>active</td>
<td>boolean</td>
<td>Is the user active?</td>
</tr>
</tbody>
</table>

tables

<table>
<thead>
<tr>
<th>role_pk</th>
<th>integer</th>
<th>primary key for a role instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>text</td>
<td>short textual name for the role</td>
</tr>
</tbody>
</table>

user_is

<table>
<thead>
<tr>
<th>user_fk</th>
<th>integer</th>
<th>id for the user instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>role_fk</td>
<td>integer</td>
<td>id for the role instance</td>
</tr>
</tbody>
</table>

user_supports

<table>
<thead>
<tr>
<th>user_fk</th>
<th>integer</th>
<th>id for the user instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>facility_fk</td>
<td>integer</td>
<td>id for the facility instance</td>
</tr>
</tbody>
</table>

Security Tables

levels

<table>
<thead>
<tr>
<th>level_pk</th>
<th>integer</th>
<th>primary key for security level lookups</th>
</tr>
</thead>
<tbody>
<tr>
<td>abbrv</td>
<td>text</td>
<td>abbreviation for the security level</td>
</tr>
<tr>
<td>comment</td>
<td>text</td>
<td>comment, if any</td>
</tr>
</tbody>
</table>

compartments

<table>
<thead>
<tr>
<th>compartment_pk</th>
<th>integer</th>
<th>primary key for compartment lookups</th>
</tr>
</thead>
<tbody>
<tr>
<td>abbrv</td>
<td>text</td>
<td>abbreviation for the security compartment</td>
</tr>
<tr>
<td>comment</td>
<td>text</td>
<td>comment, if any</td>
</tr>
</tbody>
</table>

security_tags

<table>
<thead>
<tr>
<th>tag_pk</th>
<th>integer</th>
<th>primary key for security tag instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>level_fk</td>
<td>integer</td>
<td>id for the tag level</td>
</tr>
<tr>
<td>compartment_fk</td>
<td>integer</td>
<td>id for the tag compartment</td>
</tr>
<tr>
<td>user_fk</td>
<td>integer</td>
<td>user the tag is applied to or NULL</td>
</tr>
<tr>
<td>product_fk</td>
<td>integer</td>
<td>product the tag is applied to or NULL</td>
</tr>
<tr>
<td>asset_fk</td>
<td>integer</td>
<td>asset the tag is applied to or NULL</td>
</tr>
</tbody>
</table>

Security tags must have both level and compartment. Security tags must also have a user xor product xor asset.
Webservice Specification

This section describes the Webservice API used to interact with LOST.

Overview

LOST will provide a RESTful interface for automated interaction with other OSNAP systems. Human resources systems are expected to access and modify LOST user account information. The procurements department will also integrate with LOST to add and change the classification of various products. Only specific application instances should be allowed to use the REST interface to make changes to LOST data.

A strategy involving public/private cryptography will be used to authenticate interactions via the API. Public and private key files will be configured in the LOST configuration file. Most API calls will take in two arguments:

1. A blob containing the needed arguments, encrypted using the LOST public key
2. A cryptographic signature for the blob, signed using the sender’s private key

The signature is checked to verify a correct application is the sender. The argument blob is then decrypted. To provide some protection against replay, the request argument blob includes a timestamp. Timestamps will be given in UTC time and requests differing from the current time by more than 5 seconds should cause the API call to fail the request. LOST responses will include the timestamp provided in the request rather than the timestamp of when the response was generated.

The LOST web services will be located under /rest in the LOST URI space. When /rest is queried directly, it should return an HTML document describing the provided REST functions. Each API call expects to be called using the POST method with two named fields; the ‘arguments’ field will contain the request JSON encrypted using the LOST public key and the ‘signature’ field will contain a cryptographic signature of the encrypted message using the requester’s private key. The response will be encrypted with the receiver’s public key.

API Calls

lost_key

This is the only lost call that does not authenticate the requester and uses plaintext input and output.
Request -

The request does not provide values for any of the arguments though the data and signature fields must be present in the request.

Response -

result ‘OK’ or ‘FAIL’

key The LOST public key

Example:

{  
  "timestamp": "2017-02-02 06:15:13",
  "result": "OK",
  "key": "bksaoudu......aoelchsauh"
}

activate_user

Reactivates LOST access for a user or generates a new user account if needed.

Request -

username OSNAP username to activate

Example:

{  
  "timestamp": "2017-02-02 06:15:13",
  "username": "carter"
}

Response -

result ‘OK’, ‘NEW’, or ‘FAIL’

Example:

{  
  "timestamp": "2017-02-02 06:15:13",
  "result": "OK"
}

suspend_user

Revokes access for a user. If the user does not exist or access has already been revoked, this call has no effect.
Request -

**username** OSNAP username to revoke

Example:

```json
{
    "timestamp": "2017-02-02 06:15:13",
    "username": "jackson"
}
```

Response -

**result** ‘OK’

Example:

```json
{
    "timestamp": "2017-02-02 06:15:13",
    "result": "OK"
}
```

**list_products**

Requests a listing of all products in LOST based on a filter criteria. Additional filters may be provided and additional data may be returned. The requests of the minimum form documented in the example must be accepted and the produced result must have at least the structure and fields of example provided. Asset information may not be provided by this call (e.g. no returning number of assets that are of product types).

Request -

**vendor** Case insensitive string to match against vendor name

**description** Case insensitive string to match against product description

**compartments** Security tags for the asset as a json array, all tags must be matched for compartment and level.

Example:

```json
{
    "timestamp": "2017-02-02 06:15:13",
    "vendor": "",
    "description": "notepad",
    "compartments": []
}
```
Response -

**listing** a json list containing the product information entries

Example:

```json
{
    "timestamp": "2017-02-02 06:15:13",
    "listing": [
        {
            "vendor": "Dunder Mifflin",
            "description": "LOST legal size notepad",
            "compartments": []
        },
        {
            "vendor": "big n large",
            "description": "LOST legal size notepad",
            "compartments": []
        }
    ]
}
```

**add_products**

Adds products to LOST. Requests should be atomic (all products are added or no products are added) and attempting to add a duplicate product should cause the request to fail. A product is duplicated if at the end of the call the database would have two products with the same vendor and description.

Request -

**new_products** a json list describing the new products

Example:

```json
{
    "timestamp": "2017-02-02 06:15:13",
    "new_products": [
        {
            "vendor": "Dunder Mifflin",
            "description": "LOST letter size notepad",
            "alt_description": "Children’s storybook",
            "compartments": ["adm:s"]
        },
        {
            "vendor": "Stark Industries",
            "description": "Micronized Arc Reactor",
            "alt_description": "Baseball",
            "compartments": ["nrg:ts", "wpn:s"]
        }
    ]
}
```
Response -
result ‘OK’, or ‘FAIL’

Example:
{
    "timestamp": "2017-02-02 06:15:13",
    "result": "OK"
}

add_asset

Adds a new asset to LOST.

Request -

vendor Vendor of the product
description Description of the product
compartments Additional compartments the asset should be classified under
facility Facility code the asset is initially housed at

Example:
{
    "timestamp": "2017-02-02 06:15:13",
    "vendor": "Dunder Mifflin",
    "description": "LOST letter size notepad",
    "compartments": ["wpn:ts"],
    "facility": "HQ"
}

Response -
result ‘OK’, or ‘FAIL’

Example:
{
    "timestamp": "2017-02-02 06:15:13",
    "result": "OK"
}
Tech Stack

The Logistical Operations Service Tracker (LOST) product will use OSNAP’s standard web application technology stack. Deviations from the standard technology stack are not expected to support LOST. Exceptions must be approved by the OSNAP Chief Information Security Officer (CISO) prior to deployment.

Standard Technologies

**Apache httpd** The Apache http daemon will be used to host the web application.

  *mod_wsgi* mod_wsgi will be used as the gateway between Apache and the application.

**Python** Python 3 will be used as the application development language.

**PyCryptodom** Provides cryptographic services for Python

  *Flask* The Flask framework will support development efforts.

  *WHO* User authentication will be done using the OSNAP WHO service.

**Postgres** The Postgres RDBMs will be used for persistent storage.

**PGSQL** If needed, stored procedures will be written using the default procedure language PGSQL.
Configuration

Overview

Operations personal will require a way to configure LOST in the OSNAP datacenter environment. To support this, LOST will use a JSON based configuration file. The configuration file will be named ‘lost_config.json’ and will be deployed in the same directory as the LOST application.

Configuration Items

This section enumerates the support configuration items with a short description of their intended meaning or use. In the JSON file, configuration options are grouped similarly to the following subsections.

Database Items - database

dbname The name of the database holding the LOST data
dbhost The name of the host the database is running on
dbport The port to connect to on the database host

Encryption Keys - crypto

lost_priv Path to the private key for the LOST instance
lost_pub Path to the public key for the LOST instance
user_pub Path to the public key for the system authorized to use the user webservice calls
prod_pub Path to the public key for the system authorized to use the product webservice calls

WHO Coupling - who

who_pub Public key for the WHO instance
who_url URL for the WHO instance
Configuration Sample

{
    "database": {
        "dbname": "lost",
        "dbhost": "/tmp",
        "dbport": 5432
    },
    "crypto": {
        "lost_priv": "/osnap/keys/lost.priv",
        "lost_pub": "/osnap/keys/lost.pub",
        "user_pub": "/osnap/keys/user.pub",
        "prod_pub": "/osnap/keys/prod.pub"
    },
    "who": {
        "who_pub": "/osnap/keys/who.pub",
        "who_url": "http://petunia.cs.uoregon.edu:8080/who"
    }
}
Software Design

To Do