Product Distribution

Version History

Date	Version	Description
Apr 18, 2008	1	Created
May 1, 2008	2	Added Indexing section, revised requirements
May 16, 2008	3	Changed Delivery Options to Architecture Added Product Format and Notification
Jun 9, 2008	4	Added impact to applications to Delivery, Notification, and Indexing sections.
Sep 28, 2009	5	Removed delivery, indexing, notification and proposed architectures sections; updated scope and product format; added class models and activity diagrams

Table of Contents

Problem
Each product currently distributed through the EHP web site uses separate delivery, indexing, and notification methods. The lack of a standardized product distribution process is a barrier to integration for new products and a maintenance nightmare for existing products.

Background
A product is any content generated in response to an earthquake, information about an earthquake, or another product. After a producer generates a product, it must be delivered to consumers. Product distribution is the process of transferring products from a producer to a consumer.

The primary consumer of products is the EHP web site. The EHP web site displays products directly, and links to products from event pages. There are also other product consumers that may use one product to produce another product. An example of this is PAGER using the ShakeMap grid as its primary input.

There are several methods currently used to deliver products to the EHP web site. Several producers transfer their products directly into the filesystem. Other processes fetch products from outside servers. Each time a new product is added, new user accounts must be created and new firewall exceptions added to allow access.

By creating a distribution system outside the EHP web site, new types of products may be sent and received without special EHP web site customization.

Scope
Product distribution is focused primarily on product delivery. How products are consumed by consumers is beyond the scope of this development.

Stakeholders
Product Producers - Any person or application that creates products. Both internal USGS and external non-USGS producers exist, particularly regional networks.

Product Consumers - Any person or application that uses products. The public is usually considered the consumer, but products are also used internally.

Existing Systems

Both QDDS and EIDS are message delivery systems. EIDS was intended to solve the problems of QDDS, but requires significant system resources to handle product sized messages. The delivery system is being designed to provide this functionality, but not necessarily replace either existing system. Additionally, EIDS pushes all messages to its consumers, regardless of content.

EHP Web Consumers

Indexers process between delivery and display on the EHP web site. This currently includes detecting new, redundant, and changed products, in addition to generating web pages. Detecting new, updated, and deleted products is repetitive. Many products change infrequently, but must be checked frequently to minimize overall latency.

Modifying indexing to be consumers of product messages instead of scanning files for changes eliminates this unnecessary process. Indexers will receive product messages, store product content, update the product index, and generate web pages.

QDM

QDM is the consumer for event messages. It detects redundant events, chooses authoritative versions, maintains an index (the catalog), and notifies consumers when the index changes. A similar tool is needed to detect redundant products, choose authoritative versions, and, potentially, produce a product catalog.

When QDM updates its catalog, it notifies downstream consumers of the change. Currently this notification is file based and requires secondary consumers to monitor a directory for changes.

Other Consumers

Some product consumers monitor the QDM catalog for updates and check expected product URLs, or monitor a product feed directly. These are typically self-notifications and require development and maintenance by product consumers on a product by product basis.

ShakeMap and PAGER also directly notify their consumers. ENS provides event notifications, but, by receiving products directly, could be expanded to support product notifications for these users and other products. Additionally, critical external consumers could be configured as direct consumers. By removing this burden from generators, more time can be spent improving products.

Prototype

Producers use the client prototype to build and/or send products. Consumers use the client prototype to receive products. Consumers currently create wrapper scripts which receive a product's type, code, source, version, and directory where product contents are stored as command line arguments.

A prototype QDM update product generator was also developed that sends catalog update products that describe changes to a catalog, events being added, updated, replaced, and deleted. This may eliminate the need to run a local instance of QDM.

Create and Send a Product

```
java -jar ProductClient.jar --build --type=shakemap --code=us2008abcd --source=us
      --version=1 --directory=/path/to/shakemap/us2008abcd/files
```

Receive a Product

```
/path/to/wrapper/script --type=shakemap --code=us2008abcd --source=us --version=1
      --directory=/path/to/shakemap/us2008abcd/files
```

Improvements Needed

Error handling is very rough: exceptions are logged, but not always reported via exit codes. Delivery usually only works when consumers are already running, unless a polling directory is used. If errors occur, no re-delivery is attempted without resending a product.

The prototype product format does not include tracking urls, signatures, and only partially supports status. Additionally, the format requires producers to assign a version instead of relying only on update time. While the format supports properties and links, the client does not have an interface for this data.

Producers configure the products consumers receive, and consumers receive and store all products they are sent. Consumers only process the products they are interested in, but have already downloaded the product. Consumers should receive a notification and have the option to download a product before actually receiving the product.

Similarly, storage space grows unless an outside process cleans it up. Consumers should be responsible for saving interesting files outside product distribution, and product distribution should only store received products long enough for consumers to process them.

Requirements

- Products have arbitrary size.
  A large ShakeMap is approximately 10mb. Any new system should scale with growing file sizes for products.

- Products have text and/or binary content.
- Products have one or more files.
  Map based products often include multiple text and binary files.

- Products may be related to one another.
  A PAGER product is derived from ShakeMap, which is derived from an origin and, optionally, DYFI responses. Products used during generation provide additional information that is useful during indexing and also provides a log of events.

- A version of a Product is immutable.
- Producers can add new versions of a product.
- A new version of a Product may indicate that previous versions are no longer valid.
  Detecting changes in products is a very product specific process. To simplify the indexing process, a change to a product must also change its version.

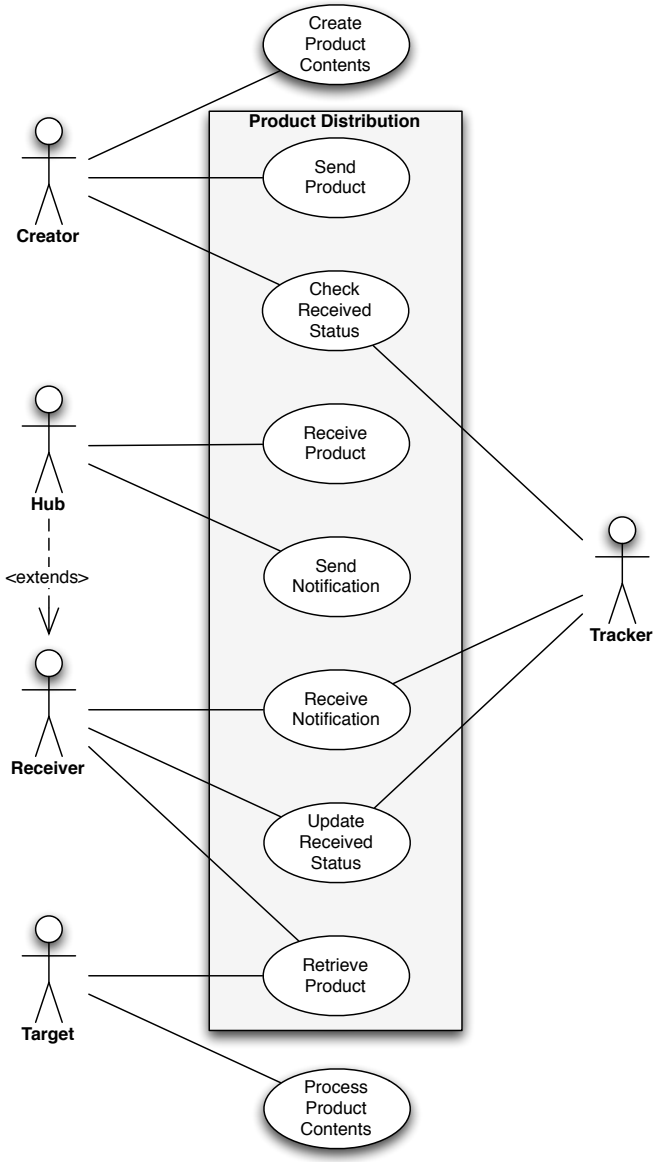
- Producers do not need to know their consumers.
- Producers do not need special access to web servers.
  Requiring producers to track their products' consumers adds repetitive work to the product generation process and does not scale to a large number of consumers.

- Delivery must be redundant.
- Delivery should not be repetitive.
- Consumers may filter products by type and producer.
  It is important products reach their consumers in a timely manner. However, products are relatively large and delivering an unwanted or already received product wastes bandwidth.

- Producers may check the distribution status of their product.
  While producers do not need to know their consumers, certain producers are interested in who receives their products, and when they were received.

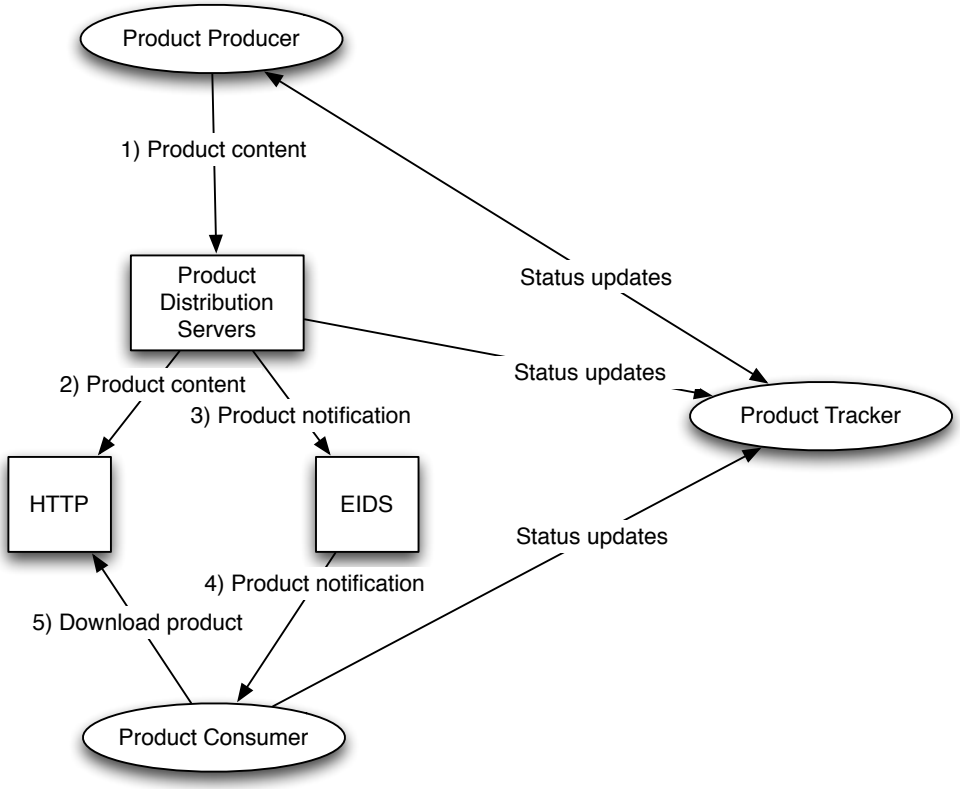
- Only authorized producers should be able to send products.
- Consumers should be able to verify a product was sent by its producer.

High Level Use Cases



Architecture

Producers transfer products to each product distribution server, which store products to a HTTP accessible URL, and notify consumers of its availability. Interested consumers download products. All send updates to an external product tracker, and Producers can check the product tracker for updates.



Latency	Internal consumers requires up to three transfers: from the producer to the product distribution server, from the server to HTTP storage (if not local), and download by the consumer.
Robustness	There is one path for each primary Product Distribution Server.
Scalability	Secondary Product Distribution Servers may be added to support additional clients.
Security	Producers have no access to consumers. All consumers connect to EIDS servers to receive notifications, and use HTTP to download products.

Product Attributes

Encapsulating the variety of product formats into a standard format simplifies product handling.

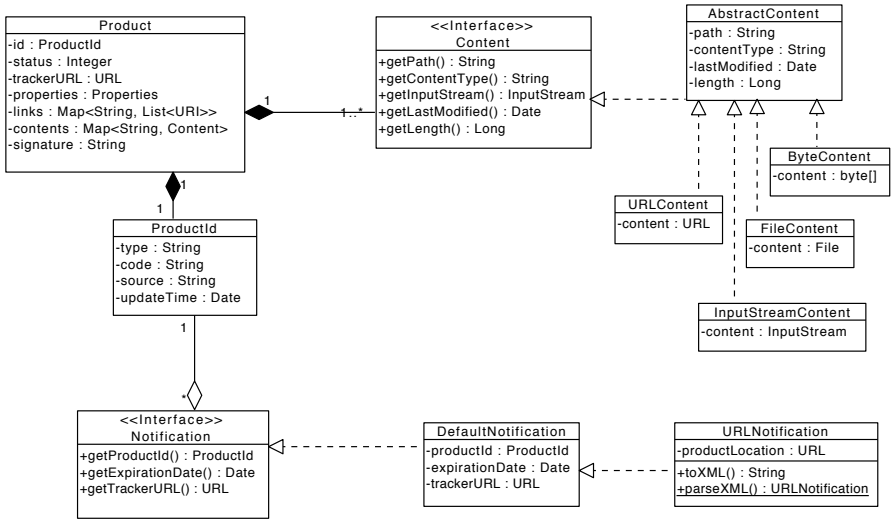
A product is a specific version of any content produced in response to an earthquake, information about an earthquake, or another product. It is possible for multiple producers to create the same type of product using the same code.

The type, code, and producer uniquely identify a product and its associated content. Update times indicate versions of the product, and products with more recent update times supersede products with less recent update times.

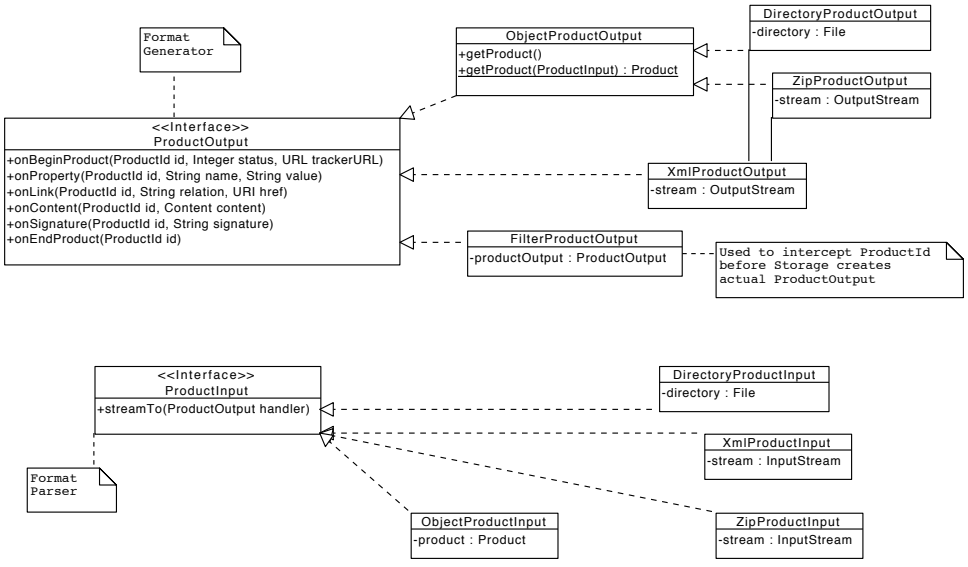
Attribute	Description
type	shakemap, pager
code	us2008abcd
producer	us, nc (network code)
~~version~~	replaced by update time.
update time	The date and time when this version of product contents was created.
status	A brief status message, like “update” or “delete”. Each type of product must use the same semantics for its status messages.
properties	Key value pairs. Each property may only have one value. Properties help simplify consumer processing, by avoiding parsing.
links	Links to other products and resources. Links have a relation, which specifies how the resource is related to the product. There may be multiple links for each type of relation.
tracker url	A url that accepts updates about a product’s distribution and processing status.
signature	Producers generate a signature which may be verified by both hubs and consumers. The signature consists of a digest of all product contents and attributes, which is encrypted using a private key and verified using a public key.

Class Models

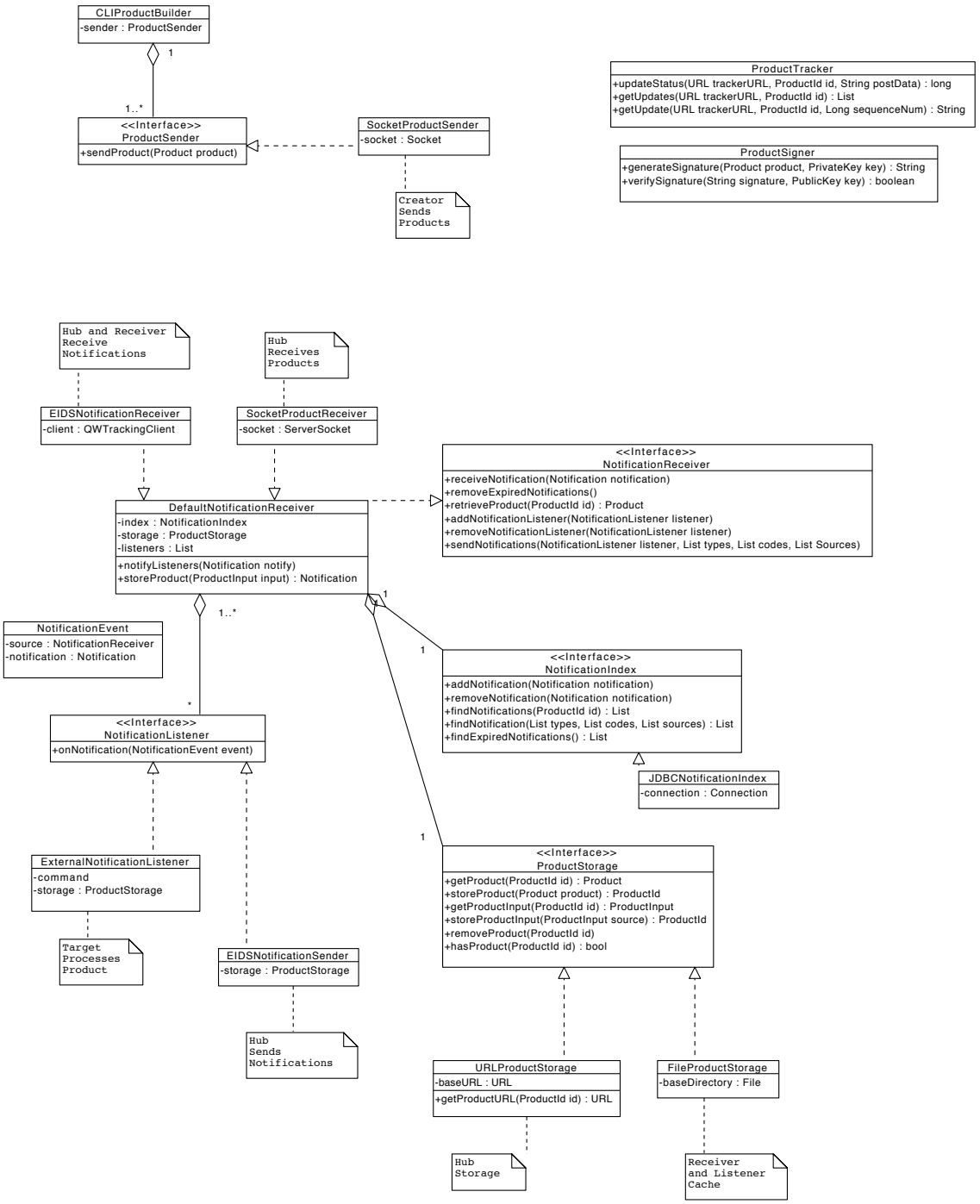
Product and Notification



Product IO

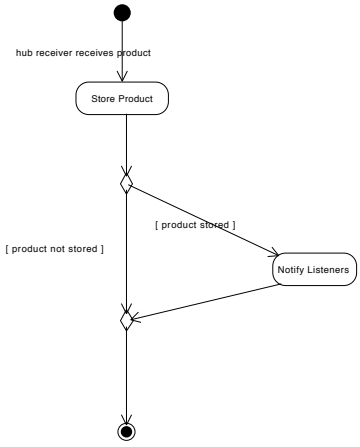


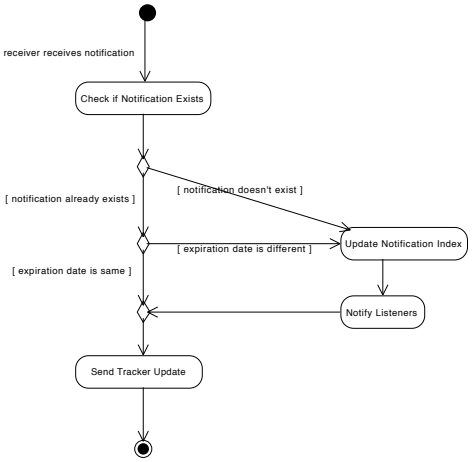
Producer and Consumer



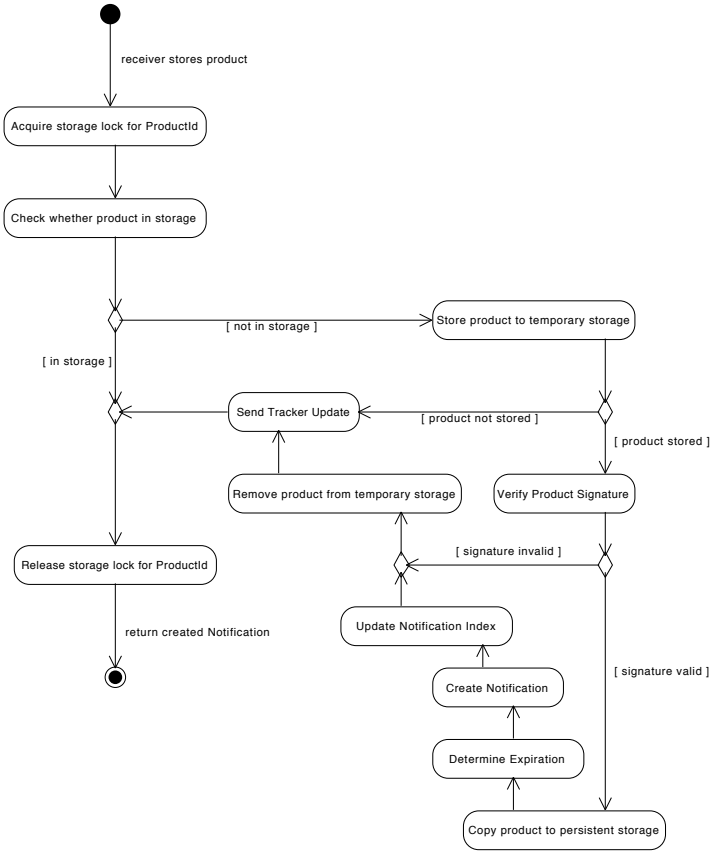
Activity Diagrams

Receive Product or Notification

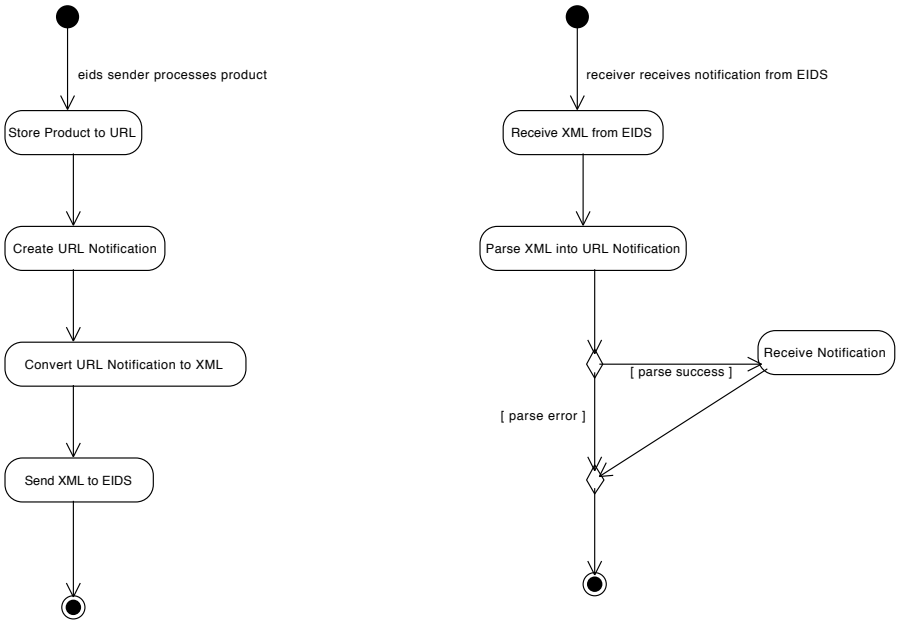




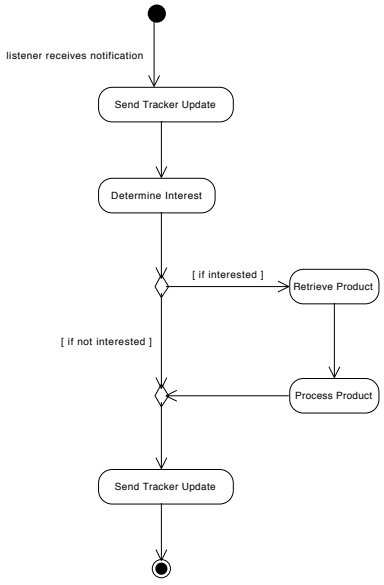
Store Product



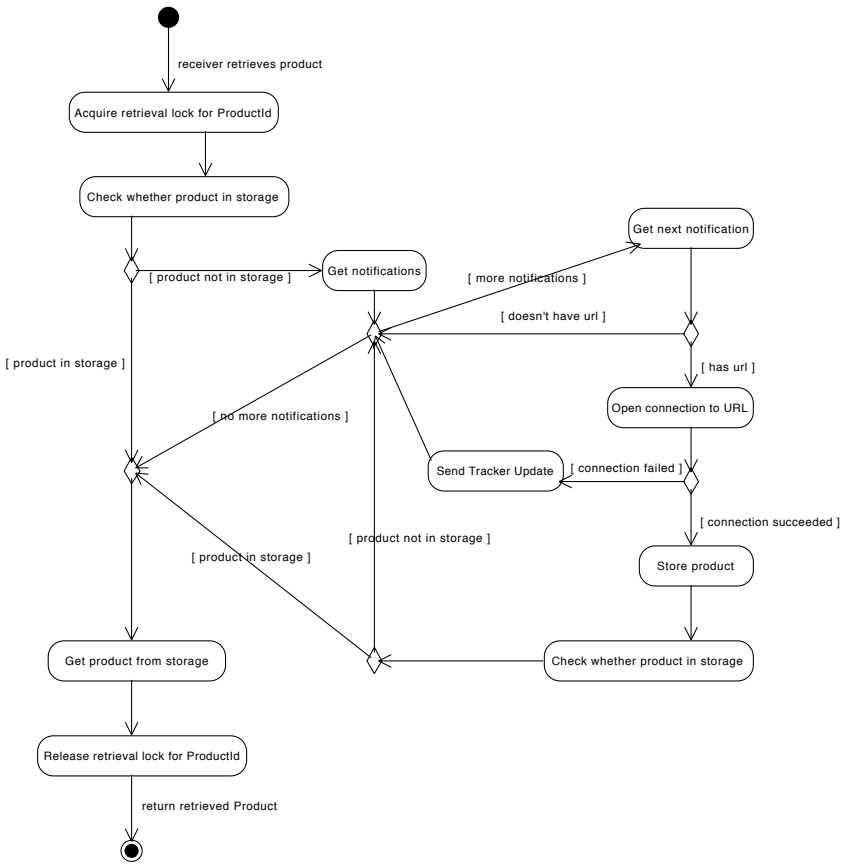
Send and Receive EIDS Notification



Listener Notified



Retrieve Product



Steps

1) Define updated command line interfaces, both for producer and consumer.
    version is being removed (could convert to optional argument that sets property)
    update time, tracking url, properties, and links need to be added
    status should be handled differently (delete could become a status value)

    ShakeMap conference next week would like these specs.

2) Create/update Product and Notification classes
    All other classes depend on these.
    ProductId will be different
    Product is adding additional fields

3) Create Product IO interfaces
    Producer and Consumer interfaces depend on these interfaces.

4) Create Producer and Consumer interfaces

5) Begin implementation

    - Implement Product IO classes
        Can reuse much of prototype.
        Need to update XML format for additional fields and product id format.

    - Implement Producer and Consumer classes
        ProductTracker and ProductSigner first, can be stubs for now
        NotificationIndex and ProductStorage classes first
        NotificationReceiver and NotificationListeners next

Glossary

Consumer - A person or application that receives products

Delivery - The process of transferring a product to an EHP web server.

Distribution - The process of transferring a product from a producer to a consumer.

Indexing - Processing that occurs between delivery and notification.

Latency - The delay between product production and consumption.

Notification - Alerting consumers to the availability of a product.

Producer - An application that generates products.

Product - A specific version of any content produced in response to an earthquake, information about an earthquake, or another product.

Pull - When referring to distribution, consumers must check if messages are available before requesting them.

Push - When referring to distribution, consumers either receive messages or notifications that messages are available.

Robustness - The ability to recover from network outages.

Scalability - The ease of adding new web servers.

Security - Any user accounts and firewall rule implications.

Stakeholder - Any person, organization, or application that is affected by this proposal.

Uniform Resource Identifier (URI) - Either a URL, URN or both.

Uniform Resource Locator (URL) - a location where a resource can be found.

Uniform Resource Name (URN) - a name that is not necessarily tied to a specific location.

xxDS - An earthquake data distribution system, either EIDS or QDDS.