Voyager Search

Spatially Enabled Enterprise Search
Agenda

• Introductions and Overview
• Voyager Search – Company Background
• Voyager Architecture including System Overview, APIs and Data Delivery / ETL Geo-process and Federation across multiple locations
• Voyager Product Demonstration
• Voyager License Model and Service offering
• Contract Vehicles
• Q&A / Next Steps
Company Background

- Voyager was founded in June 2008
- Marriage of modern search and geospatial technologies
- Built on open-source components
- Focused on Enterprise Data Search across an organizations data holdings
Why Voyager?

› Voyager natively understands spatial – proven solution across large Federal, Defense/Intel agencies as well as the world’s largest commercial organizations in petroleum and mining

› Tightly aligned with Esri stack from the Desktop to ArcGIS Server to AGOL/Portal for AGOL

› Out-of-the-box, powerful indexing and catalog of over 1,700 spatial and non-spatial formats including vector, raster, LiDAR, video, jpegs, MS Office

› Thoroughly indexes file servers, databases and web service content

› Connector framework allows for indexing custom and 3rd party repositories

› Robust API / custom development framework for developers

› Federate content from remote physical locations and networks

› Adds spatial context to non-spatial documents via Geo-tagging

› Creates and maintains the relationship of data to layers and layers to maps or maps to services

› Single sign on / role based security

› Tremendous ROI including identification of duplicate data and broken links
Built on and Extends Apache Lucene

› Voyager is built on Apache Lucene — so are Amazon, Kayak, Twitter, Netflix and Apple and every other thing you search on the web
› High-performance, full-featured search engine
› We’ve extended it to support spatial
› We’ve made it accessible to the non-developers for query management, tuning etc.
› We’ve given it a UI so you can search, sort and filter immediately
› We’ve extended it to support data discovery
› We’ve added tools for enriching your content
Architecture Overview
Voyager Architecture Overview

- **Connectors**
  - Geospatial
  - Office
  - Web Services
  - Database
  - Doc Mgt Sys
  - Open Search
  - CDR Broker
  - Autonomy
  - Custom

- **Pipeline**
  - Normalization
  - Geo-Tagging
  - Gazetteer
  - Transform
  - Custom Biz Logic
  - Open Sextant
  - Other...

- **Index**
  - Analysis
  - Text Search
  - Faceting
  - Caching
  - Query parsing
  - Clustering
  - Synonyms
  - Suggestions
  - Authentication
  - Authorization

- **Web Framework**
  - Browsers
  - Subscribe Services
  - Applications (Web, Desktop & Other)

- **Content & Applications**
  - Data Delivery (ETL)
  - Analytics (run models)
  - DNI CDR Brokered Search
  - Custom Apps

- **Discovery**
  - REST, RSS, XML, OpenSearch, CSW

- **Index**
  - Integrated Security – LDAP, Windows Authentication, Voyager Internal, SAML, HTTP Headers, GeoAxis, Custom

- **Application**
  - Open Search
  - Office
  - Web Services
  - Database
  - Doc Mgt Sys

Voyager System Overview

- Open Search
- Office
- Web Services
- Database
- Doc Mgt Sys

- Pipeline
  - Geospatial
  - Office
  - Web Services
  - Database
  - Doc Mgt Sys
  - Open Search

- Index
  - REST, RSS, XML, OpenSearch, CSW
From Confusion to Clarity and Control with Voyager

**Data Discovery**
Voyager can read 1600+ file formats, and if that's not what you need, then we can always add one more.

**Enrichment**
During indexing, content is run through Voyager's indexing pipeline to accomplish many tasks.

**Search Services**
Once indexed, the contents can be searched, sorted and explored in a variety of ways.

**Access**
Use Voyager's user interfaces or build your own.

**Workflows**
Voyager processing framework can improve ROI because it is a fast iteration engine.

**Unclassified**

### Databases
- Oracle, SQL Server, and more...

### File Systems
- Desktops, Laptops, Servers, and FTPs

### Document Management Systems
- SharePoint, Dropbox, and Google Drive

### Web Services
- ArcGIS Online, ArcGIS Server, WMS, WFS, FME Server, Geocortex and CSW

### Stand Alone Metadata XML Documents
- CSV, SVG, OGG, CAD, TGA, FLV, KMZ, ISO, CPP, SWF, PNG, WMA, XLS, XML
UNCLASSIFIED

The image shows a pie chart with the following categories:

- Reporting
- Connectors & Extractors
- Catalog Creation
- Content Enrichment
- Search Services
- UI
- Extraction and Transformation
- Exposing Catalog

The chart is part of a document related to Voyager, an application or service, indicating various functionalities and features.
Connectors & Extractors

- Built-in tools for connecting to and “ingesting” content
- Connectors to servers, Web services, databases, catalogs, CMS and other search engines.
- Extractors “read” 1,770 file formats
- Understanding of complex GIS formats — Shapefiles, for example
Catalog Creation

- Voyager creates a catalog as the index is created
- Content update = catalog updates
- The catalog is a set of “library cards” for everything — Web services, spatial or non-spatial content, content with or without metadata
- The Lucene index is a great Metadata Catalog
Content Enrichment

- Transformation and augmentation of existing metadata during the indexing process
- Creation of thumbnails
- Duplicate content detection
- Metadata normalization or adding fields to the metadata
- 3rd-party rules service (content type and form taxonomy)
Search Services & Index Management

› Keyword, place name and spatial query search
› Tagging, rating, nomination (GOLD, Crisis)
› Integration with enterprise security
› Index Management, tuning, federation, query elevation and more
User Interfaces

- Classic Voyager UI for index management and configuration
- Voyager Navigator for end users
- Voyager widget for ArcGIS Online/Portal
- Voyager’s ArcGIS desktop toolbar
Content Extraction and Transformation

- Delivery and transformation tools
  - Clip and ship
  - Transform

- Data management tools
  - Clean up content
  - Facilitate data consolidation
  - Fix Metadata
  - Remove duplicates

- Modeling tools
  - JEMA integration
Process Search Results

› New Processing Framework
› 20 + tasks (out-of-the-box)
› Tasks can easily be extended
› Written entirely in Python
Creating Custom Tasks

› Task files are placed in well-known folders
› Parameters are defined in a file using JSON (i.e. zip_files.info.json)
› Source code is Python (i.e. zip_files.py)
› Written entirely in Python
› Helper functions exist to help with accessing data, reporting status, etc.
› Tasks are open sourced on github: https://github.com/voyagersearch/voyager-py/tree/master/processing
Expose, Publish and Use

- Exposed through industry standards
- APIs
  - RSS/Atom Feeds (notification services)
  - Open Search (search engine & Globe integration)
  - Solr/Lucene (best of breed, open source search)
- Integration with other systems
  - Voyager Collect Service
  - Voyager-to-Voyager federation
  - Voyager-to-Autonomy federation
Reporting

- Business analytics
  - Logging of user activity (search queries and content access)

- Visualization of content catalog
  - What’s in these catalogs?
  - Metrics by content type
  - Graphs and chats of content composition
  - Field usage reports
APIs

• We have APIs!
  • http://<voyager>/solr/select?
    • Directly query and manipulate the Lucene index
  • http://<voyager>/api
    • Query and manipulate the Voyager application (and parts of the index)
  • Allows us (and others) to build custom UI / UX over the Voyager core
API Methods

- ~/system/
  - Provides system status information and settings
- ~/discovery/
  - Provides information about discovery settings and state, and control over indexing
- ~/process
  - Provides information and control over our processing framework
API Demo ...
GeoProcessing Framework

- Adds processing to discovery process and search workflows
- Once you’ve found data in Voyager …
- Our processing FRAMEWORK lets you do things with that data
Geo-Processing Framework

- Processing is applied to a set of one or more search results
- General order of operations:
  - Collect Data > Process …
  - Select a task to run …
  - Enter and confirm parameters …
  - Execute … (Wait) … Enjoy!
# GeoProcessing Framework

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add to Geodatabase</td>
<td>Add input data to an existing geodatabase</td>
</tr>
<tr>
<td>Build Raster Pyramids</td>
<td>Builds pyramids for input raster datasets. Pyramids improve display performance.</td>
</tr>
<tr>
<td>Calculate Raster Statistics</td>
<td>Calculates statistics for raster datasets and mosaic datasets</td>
</tr>
<tr>
<td>Clip Data</td>
<td>Clips selected search results using the clip geometry</td>
</tr>
<tr>
<td>Convert to KML</td>
<td>Converts feature or raster data into Keyhole Markup Language</td>
</tr>
<tr>
<td>Copy Files</td>
<td>Copies files to a target folder</td>
</tr>
<tr>
<td>Create an ArcMap or Layer Package</td>
<td>Package data into a single compressed file (mpk or .pk)</td>
</tr>
<tr>
<td>Create GeoPDF</td>
<td>Add selected data to a GeoPDF document</td>
</tr>
<tr>
<td>Delete Files</td>
<td>Permanently delete files</td>
</tr>
<tr>
<td>Mosaic</td>
<td>Mosaic input rasters to a new raster dataset where the result is to be downloaded</td>
</tr>
<tr>
<td>Mosaic to Workspace</td>
<td>Mosaic input rasters to a new dataset in an existing workspace</td>
</tr>
<tr>
<td>Move Files</td>
<td>Moves files to a target folder</td>
</tr>
<tr>
<td>Publish to ArcGIS Server</td>
<td>Publish results as a map service on ArcGIS for Server</td>
</tr>
<tr>
<td>Publish to Portal for ArcGIS</td>
<td>Publish results as a map service to a portal for ArcGIS</td>
</tr>
<tr>
<td>Replace Data Source</td>
<td>Replaces a layer's data source</td>
</tr>
<tr>
<td>Replace Workspace Path</td>
<td>Replace the workspace path for layer files and map document layers</td>
</tr>
<tr>
<td>Write Metadata</td>
<td>Create or update existing metadata</td>
</tr>
<tr>
<td>Zip Files</td>
<td>Compress input files into a single zip file</td>
</tr>
</tbody>
</table>
GeoProcessing Framework

- FRAMEWORK that is easy(~ish) to extend - Each process is simply underpinned by:
  - A configuration file: defines names and types of fields in the UI
  - A python file: defines processing logic
  - Processing tasks are open source (https://github.com/voyagersearch/voyager-py)
  - Use ours as templates / inspiration
  - Edit / add your own process with well-documented conventions
    - <taskname>.info.json to define UI fields and field types
    - <taskname>.py to do things with those field values
GeoProcessing Framework

- Documented Conventions / Example
- Configuration file (Voyager Results, Other Field Types, etc.)
- Python script (Execute Method, Get Parameters, Other Logic, etc.)
- Utility classes
  - Get Results / Field Values
  - Status Writer

```python
def execute(request):
    
    """Clips selected search results using the clip geometry.
    :param request: json as a dict.
    """
    clipped = 0
    errors = 0
    skipped = 0
    parameters = request['params']
```
GeoProcessing Demo ...
Web Applications and Developer APIs
Indexing Architecture
Distributed Architecture

Scale indexing to support a variety of environments
- Remote Locations
- Compartmentalized Information
- Organizational structure
- Large Volumes (index shards)

Search all data, all of the time, at scale
Implementation Examples

Content Repository

1. Small
2. Large
3. Distributed
BLM Data Exchange Architecture (high-level)

Voyager’s Data Delivery and ETL processes are leveraged to transfer content from one location to another:
- Internally users can access data from other systems
- Public can order data on demand in a format of their choice
Voyager Product Demonstration