INTRODUCTION TO ERDAS APOLLO REST API
# Contents

Using This Manual .......................................................................................................................... 1

Section 1: Basics ............................................................................................................................. 3  
  Exercise 1: Basic Setup .................................................................................................................. 4

Section 2: Basic Catalog Operations ............................................................................................... 8  
  Exercise 1: Crawling and Querying .............................................................................................. 9  
  Exercise 2: Attributes and Properties ......................................................................................... 16  
  Exercise 3: Delete ......................................................................................................................... 19

Section 3: Jobs Management .......................................................................................................... 21  
  Exercise 1: Jobs Management ...................................................................................................... 22

Section 4: Pyramids and Thumbnails .............................................................................................. 24  
  Exercise 1: Pyramids and Thumbnails .......................................................................................... 25

Section 5: OGC and Legacy Vector Services .................................................................................... 27  
  Exercise 1: Services ..................................................................................................................... 28  
  Exercise 2: Services-Vector .......................................................................................................... 30

About Us ......................................................................................................................................... 32

Copyright and Terms of Use ........................................................................................................... 33
Using This Manual

This manual contains step-by-step instructions on how to perform certain processes. You should be aware that each exercise provides a single path through the application's tools. In most cases, there are various ways to maximize tool usage, depending on the individual project.

This exercise manual is provided to the student, along with all images used by the instructor. Copies of the presentation slides are available upon request. This provides the capability for recreating the processes performed in class at a later date, as well as the key points on any theory involved.

Exercise Conventions

Section Title Page  States the objective of the exercises and lists the application tools to be used within the various tasks.

Exercise Tasks  Each exercise is split into a set of tasks. After the course, these tasks will help you locate within the manual where you performed a certain set of steps.

Questions  The instructor may quiz and/or review with you, following each exercise.

Notational Conventions

Bold Text  Any text that is bold indicates buttons, tabs, group names, dialogs, and field names that are visible in the workspace.

Monospace Text  Any text that is monospace indicates a file name, text entered by you, or code such as HTML, XML, JavaScript.

Graphics  To help you locate buttons and objects used in the exercises, the button icons will be next to the button name in the text.

Diagrams  Optional diagrams can show you how to use some of the application's tools.

This is a tip, describing a different way you can enter information into the software or giving relevant information about the software.

This is a reference book, listing an external location (website, Help document) where you can read or see more information.
Section 1: Basics

Section Objective

Learn the basics of the ERDAS APOLLO REST API.

Tools Used

Swagger UI

Swagger user interface is designed to allow users to easily formulate and execute REST API queries against a target endpoint.

Notepad / Notepad++

Simple text editing programs. In these exercises they are used only to display blocks of JSON so that they may be conveniently copied to the Swagger UI.
Exercise 1: Basic Setup

Objective
Get familiar with Swagger UI Tool.
- Start the Virtual Machine for the exercises.
- Open Swagger UI and authenticate.
- Set up root exercise aggregate using Swagger and the REST API.

Task 1: Start the Virtual Machine

Get the virtual machine up and running that you will use for the exercises.

1. On your machine, find and open VirtualBox. You should see a machine named HxGNLiveApollo. Start this VM by double clicking on it, or by selecting it and one of the Run options.
2. Log in to the machine using the username HxgnUser and the password hxgnLive2017. Allow the machine a few seconds to get through its startup processes.
3. While waiting for the startup process to complete, look at the bottom right end of the toolbar, specifically for the Microsoft Action Center flag. If the flag has a tiny clock emblem under it, then maintenance is in progress.
4. Click the flag to open a small options window. A much larger flag and clock opens at the top with the words Maintenance in progress next to it. Click that message.
5. This opens the Action Center. Click Maintenance, and then click Stop maintenance.

Task 2: Open Swagger UI and Authenticate

Use a browser to access the Swagger UI.

1. Open the web browser of your choice. Google Chrome is recommended, but Firefox is also available.
2. Go to the URL http://localhost/erdas-apollo/api-docs. If you are using Chrome, your default page has been set to this URL.
3. Verify that the URL in the text field at the top is set to http://localhost/erdas-apollo/api-docs/api-docs.json.
4. In the upper right next to the Explore button, type admin in the first field and apollo123 in the second.

5. Click the Explore button.

There is not much feedback for completing this action successfully. You will see much of the UI reload and a brief message about fetching the resource list. This form of authentication is a temporary workaround that will be removed later as authentication is streamlined.

Task 3: Set Up Root Exercise Aggregate

Create an aggregate using the Aggregate CRUD endpoint.

1. Find the line in the Catalog Service API that reads Aggregates: APIs specific to catalog aggregates.
2. Click either the word Aggregates at the beginning or Show/Hide to expand the Aggregates API operations.
3. Click the top GET operation, the one next to /content/catalog/aggregates.json.
4. Scroll to the bottom of the operation block, and click the Try it out! button.
5. Find the Response Body and scroll to the results block. There should be two results, ROOT and IMG, that have an id field. You need to copy id for ROOT for use in the next steps.
6. Click the **POST** operation to expand it.

7. Click the example block to the right of the text field. This adds a template POST body to the text field. If the text field is too small, click and drag from the bottom right corner to expand it.
Exercise 1: Basic Setup

8. Replace the string placeholders with the root id that you saved from step 5 and the name you want to use for your new aggregate, respectively. Remember to keep the quotation marks around each value.

![JSON Body Example]

9. Scroll down and click the POST operation’s Try it out button.

10. Verify that you received a Response Code of 201 and a response body containing the name and id of your new aggregate.
Section 2: Basic Catalog Operations

Section Objective

Learn how to crawl files, query data, update, and delete items using the ERDAS APOLLO REST API.

Tools Used

Swagger UI

Swagger user interface is designed to allow users to easily formulate and execute REST API queries against a target endpoint.

Notepad / Notepad++

Simple text editing programs. In these exercises they are used only to display blocks of JSON so that they may be conveniently copied to the Swagger UI.
Objective
Become familiar with the basics of using the ERDAS APOLLO REST API to crawl data.

- Crawl sample data using the Swagger UI.
- Query against sample data using various filters and options.
- Run an advanced query.
- Query from Image CRUD API.
- Crawl data from a database.

Task 1: Crawl Data

Crawl data using the Swagger UI.

1. Find the line in the Catalog Service API that reads **Crawl Operations: APIs to crawl specific data into the APOLLO Catalog.**
2. Click **Crawl Operations** or Show/Hide to open this section of the Catalog Service API.
3. There is only one available operation. Click the **POST** operation to expand it.

   This POST Operation looks similar to the one performed in basic setup. However, the POST body of a crawl operation is significantly larger and more complex than the one for creating an aggregate. Expand the body text window to make it easier to read the json that will be pasted there.

4. It is possible to click on the example model block and fill in all of the placeholder values in a manner similar to the aggregate POST in basic setup. However, a JSON block that should be easier to manage has been provided for you. Open a file explorer and navigate to C:\DemoMaterials\Crawl and open the file called **CrawlDirJSONBody.json**, preferably in Notepad++.

5. Copy the contents of the file into the body text field under the **POST** operation. There are two critical placeholders that must be filled in. The first is the one in the **directory** field, the second is the **parentId** field.

6. For **directory** replace placeholder with **C:/DemoMaterials/Data/Raster**.

   Be sure to use / instead of \\, as \ is handled as an escape character.
Exercise 1: Crawling and Querying

7. For `parent_id`, you need to retrieve the ID of the parent aggregate you created in basic setup. That ID should still be in the response body of the operation that created it, but if you have lost that response body (for example, by reloading the page), then you can find the id again by performing an Aggregate GET operation and finding the id associated with your generated aggregate.

8. Scroll to the bottom of the POST operation and click **Try it out!** button.

9. Verify that the **Response Code** is 202 and that the response body does not contain an error message.

---

**Task 2: Query Against Sample Data**

Perform basic querying using the Catalog endpoint.

1. Find the line in the Catalog Service API that reads **Entire Catalog: APIs specific to all catalog items**.

2. Click **Entire Catalog** or **Show/Hide** to expand the Entire Catalog API operations.
3. Click the GET option next to /content/catalog/items.json.

4. This is the primary GET all items catalog search. There are a number of filtering options. First, find the keywords text field and enter sample.

5. Click Try it out!
   This performs a get and collects only results containing the keyword sample. The Response body should contain only a single result, the sample.ecw file crawled previously in the exercise.

6. Remove sample from keywords field. Type name asc in orderby field.

7. In classes field, click com.erdas.rsp.babel.model.ResourceAggregate. Click Try it out!

8. Verify that the results of the response body now contains the aggregates in the catalog.
   Assuming you have been following these instructions exactly, there will be four items in alphabetical order by name:
   - ROOT
   - Raster
   - IMG
   - [your root exercise aggregate name]
Exercise 1: Crawling and Querying

The Raster directory was created because the `replicateDirectory` option in the crawl POST body was set to true, creating a child aggregate under the one you created in basic setup.

The IMG directory was created prior to HxGN LIVE to assist with the Metadata training and is otherwise unrelated to these exercises.

9. The start and maxresults options are explained adequately on the Swagger UI page itself. Instead, change the profile option from brief to full and click Try it out! The response returned should be significantly more verbose!

10. In the intersect field, enter the text:
    
    "POLYGON((-89.37233984599706 31.198089395911598, -89.372454399355231.200175750918177, -89.37002721557404 31.200274201105486, -89.36991271503733 31.19818783805115))"

11. Set the profile field back to brief(default) and clear the classes filter. Click Try it out!
    
    The results block should show you the 1955555 dataset, along with other information, such as its parent Raster aggregate.

Task 3: Advanced Search

Perform an advanced catalog search.

1. Open the POST operation under the Entire Catalog endpoint at /content/catalog/search.json.

2. In file explorer, locate the file under C:\DemoMaterials\Crawl called AdvancedSearch.json and open it.

3. Paste its contents into the payload block of the operation. This JSON block details many of the same filters from the previous task, but allows advanced users to specify additional fields.
Exercise 1: Crawling and Querying

4. Click **Try it out**!

You should see the file 19155555 in the response body.

Task 4: Image CRUD Search

Perform a search using the Image CRUD endpoint.

1. Find the line in the Catalog Service API that reads **Images: APIs specific to image catalog items.** Click Images or Show/Hide to open this section of the Catalog Service API.

2. Click the GET operation at /content/catalog/datasets/images.json.

   A number of the parameters from the items query display, though the **class** filter is missing, as you can search only for images from this endpoint.

3. Click **Try it out**!

   The images crawled into the catalog are listed in the response body.

   The Images endpoint is one of several that trades flexibility for ease of use, alongside Aggregates, Vectors, Point Clouds, and Generics. These endpoints also only manipulate the catalog.
Task 5: Database Crawl

Perform a database crawl, rather than a directory crawl.

1. Find the line in the Catalog Service API that reads **Crawl Operations: APIs to crawl specific data into the APOLLO Catalog. Click Crawl Operations or Show/Hide to open this section of the Catalog Service API.**

2. There is only one available operation. Click the **POST** operation to expand it.

3. It is possible to click on the example model block and fill in all of the placeholder values in a manner similar to basic setup. However, a JSON block that should be easier to manage has been provided for you. Open a file explorer and navigate to C:\DemoMaterials\Crawl and open the file called **CrawlDatabaseJSONBody.json**, preferably in Notepad++.

4. Copy the contents of the file into the text field under the **POST** operation. In this case, the only placeholder to fill in is the **parentId**.

   ![](image)

   The necessary fields under **database** change depending on the type of database used. In this case, a postgres/postgis database has been provided on the test vm (localhost).

   - **Username** - postgres
   - **Password** - apollo123
   - **Database name** - CherokeeShape
   - **SRS** – 2240
   - **Accessible on Port 5432**
5. Click Try it out!

The job should report a status of 202. Further queries, such as a GET under the vectors or items endpoint, should reveal the presence of your new database data in the catalog.

The `connectionName` TestAPI creates a sub-aggregate in the catalog.
The data that you have crawled from your database is contained within this.
Exercise 2: Attributes and Properties

Objective
Become familiar with manipulating Attributes and Properties using the REST API.

- Update an attribute.
- Add a property to an item.

Task 1: Update an attribute

Update an attribute on a catalog item.

1. Ensure that you have crawled data into your catalog.
   
   If you have been performing these tasks in order, this should be done already, but if not, you may need to perform an additional crawl.

2. Retrieve the id of the aggregate you want to update. For example, one of the GET queries detailed earlier in these exercises. This task uses the Raster aggregate.

3. Find the PATCH operation under the Aggregates endpoint.

4. Copy the id of the item to be updated into the ID field.

5. Open the file at C:\DemoMaterials\Attribute\Attribute.json. Copy the contents into the text field next to catalogAttributePatchPay.
6. Click **Try it out!**
   
   Though the response body yields no content, the **Response Code 204** indicates success.

7. However, to verify that it succeeded, search for aggregates again using the GET operation under the correct endpoint (aggregates, if you used the Raster aggregate as your target item) and look for your chosen aggregate. Verify the name and title fields are set to **Raster2**.

---

**Task 2: Add a Property to an Item**

Use two different methods to add a property to a catalog item.

1. Find the line in the Catalog Service API that reads **Entire Catalog: APIs specific to all catalog items**.

2. Click **Entire Catalog at the beginning** or **Show/Hide** to expand the **Entire Catalog API** operations.

3. Click the **POST** option next to `/content/catalog/items/{ID}/{propertyname}`.

4. In the **ID** field, enter the **ID** of the item to which you want to add a property. You can use the GET operation in the general catalog items section to retrieve all catalog items in order to get an ID.

5. In **propertyName**, type a name for your property. This task uses **WhereAmI**.

6. In ‘type’ field, enter a **1** (do not include quotations) to give the property a type of string.

7. In ‘value’ field, enter the value of your new property. This task uses **VegasBaby**.

8. Click **Try it out!**
   
   This should generate a 204 response code with no response body. There will be a more rigorous verification that the property was created at the end of the task, after you have explored a second method of adding properties.

   As an alternative, you can add a property using a **propertyPostPayload**. In this use, you must first remove the values from the ‘type’ and ‘value’ fields.

9. Type a new property name. In this task, the new property name in the **propertyName** field is **WhereAmINow**.

10. In the **propertyPostPayload**, type the following:

    ```json
    {
        "type": "1",
        "value": "StillVegas"
    }
    ```

11. Change the parameter content type to **application/json**.

12. Click **Try it out!**
13. Verify the results by searching under the GET operation associated with
/content/catalog/items/{ID}/{propertyName}.json.

Alternatively, you can use a different search, such as the one at
/content/catalog/items/{ID}.json with the profile set to ‘full’ rather than ‘brief.’ The
properties block will show the newly added properties.

```
"properties": {
  "generateAggregateThumbnails": "N/A",
  "aggregateDirty": "false",
  "generateAggregateFootprints": "true",
  "generateAggregateISOFilename": "true",
  "whereAmINow": "StillVegas",
  "generateAggregatePyramids": "N/A",
  "WhereAMI": "VegasBaby"
}
```
Exercise 3: Delete

Objective
Become familiar with the REST API’s delete functions.

- Delete from the catalog using the Delete endpoint.
- Delete from the catalog using the items DELETE endpoint.

Task 1: Delete from the Catalog using the Delete endpoint

Delete from the Catalog using the designated endpoint.

1. Find the line in the Catalog Service API that reads Delete Operations: APIs to delete single or bulk items from the APOLLO Catalog. Click Delete Operations or Show/Hide to open this section of the Catalog Service API.

2. There is only one available operation. Click the POST operation to expand it.

3. There is only one field to fill in – the POST body of the delete operation. However, due to the nature of the contents, there is no easy-to-copy block of JSON for this one. Find the ids of your two crawled ecw files. Do this by using a properly filtered GET items command, or the GET all images command under the Images endpoint.

4. Click the model Example Value block to the right to auto-fill the template for the post body into the text field. Replace string with the id values of your ecw files. All of these values must be in quotes and comma separated.
Exercise 3: Delete

5. Click **Try it out!** Now when you search for those items, you should get an empty result block.

---

Task 2: Delete from the Catalog Using the Items DELETE

This alternative delete method uses the Catalog Items DELETE operation. This is essentially a redirect to the previous method that does not allow for multiple deletes, but is more structured and easier to use.

1. Find the line in the Catalog Service API that reads **Entire Catalog: APIs specific to all catalog items.**
2. Click **Entire Catalog at the beginning or Show/Hide** to expand the Entire Catalog API operations.
3. Click **DELETE** option next to `/content/catalog/items/{ID}.json`.

4. There is only a single available field here. Enter the ID of the item that you want to delete. This task assumes the aggregate that was renamed Raster2 earlier in the exercises.

5. Click **Try it out!** Now when you search for the deleted item, you should get an empty result or error.
Section 3: Jobs Management

Section Objective

Learn how to manage jobs using the ERDAS APOLLO REST API.

Tools Used

**Swagger UI**

Swagger user interface is designed to allow users to easily formulate and execute REST API queries against a target endpoint.

**Notepad / Notepad++**

Simple text editing programs. In these exercises they are used only to display blocks of JSON so that they may be conveniently copied to the Swagger UI.
Exercise 1: Jobs Management

Objective
Become familiar with ERDAS APOLLO REST API's Job Management functionality.

- View jobs in progress.
- Cancel a running job.
- Delete a job.

Task 1: View Jobs
Find existing jobs.

1. Find the line in the Catalog Service API that reads Jobs Management: APIs specific to job management services.
2. Click Jobs Management or Show/Hide to expand the Entire Catalog API operations.
3. Open the GET command for /jobs.
4. Click Try it out to view a list of completed jobs. If any jobs are running when this command is run, the results will report those as well.

Task 2: Cancel a Running Job

This task will NOT be performed in this training. The data stored on the machines will not crawl slowly enough to warrant cancellation. However, for your reference, follow these steps to cancel a job that is currently running.

1. Open the PUT command at /jobs/{jobId}/cancel while a lengthy job is in progress.
2. Enter the target jobId into the jobId text field and click Try it Out! This cancels the job in progress.
Task 3: Delete a Job

Delete a job.

1. Open the DELETE operation at /jobs/{jobId}.
2. Fill the text field next to jobId with the id of the job you want to delete.
3. Click Try it Out! You can verify the result by viewing jobs again using the GET command.
Section 4: Pyramids and Thumbnails

Section Objective

Learn how to manage pyramids and thumbnails using the ERDAS APOLLO REST API.

Tools Used

**Swagger UI**

Swagger user interface is designed to allow users to easily formulate and execute REST API queries against a target endpoint.

**Notepad / Notepad++**

Simple text editing programs. In these exercises they are used only to display blocks of JSON so that they may be conveniently copied to the Swagger UI.
Exercise 1: Pyramids and Thumbnails

Objective
Become familiar with managing pyramid and thumbnail metadata using the REST API.
- Create a pyramid file and delete it.
- Generate a thumbnail.

Task 1: Create and Delete a Pyramid File

Create a pyramid metadata file and later delete it.

1. Initiate a crawl operation using the post body found at C:\DemoMaterials\Pyramid-Thumbnail\CrawlPyrJSONBody.json, replacing the parentId with a valid id and the directory value with C:/DemoMaterials/Data/Pyramid.
   
   ![Tip]
   Be sure to use / instead of \, as \ is handled as an escape character.

2. Find the line in the Catalog Service API that reads Pyramids: APIs specific to pyramids for catalog items.

3. Get the Id of the new image, which is sp28gtif.img.

4. Click Pyramids at the beginning or Show/Hide to expand the Entire Catalog API operations.

5. Open the POST operation at /pyramids.

6. Click the example block to fill the body text field with the template. Replace the id placeholder with the id of the sp28gtif dataset.

7. Click Try it out!
   
   Optional: You can verify that the pyramids were created by performing a GET operation with the profile set to full. Search the response body (CTRL+F will work) for pyramidDescriptor and find the one under sp28gtif. You can also verify the result using Data Manager.

8. Open the DELETE operation at /pyramids/{ID}.

9. In the ID text field, type the sp28gtif dataset ID.

10. Click Try it out!
   
    Optional: You can verify that the pyramids were deleted by performing a GET operation with the profile set to full. Search the response body (press CTRL+F) for pyramidDescriptor and find the one under sp28gtif. It should say null instead of having the values it did before.
Task 2: Generate a Thumbnail

Generate a thumbnail for a dataset.

1. Find the line in the Catalog Service API that reads Thumbnails: APIs specific to thumbnails for catalog items.

2. There is only one operation. Open the POST operation.

3. Click the Example block to fill the body text field with the template.

4. Replace the string placeholder with the id of sp28gtif.

5. Click Try it out!

   You can verify the result by performing a GET request with the profile set to full. You should be able to locate thumbnail data that corresponds to sp28gtif.img by pressing CTRL+F. You can also verify the result with Data Manager.
Section 5: OGC and Legacy Vector Services

Section Objective

Learn how to create and manipulate services using the ERDAS APOLLO REST API.

Tools Used

Swagger UI

Swagger user interface is designed to allow users to easily formulate and execute REST API queries against a target endpoint.

Notepad / Notepad++

Simple text editing programs. In these exercises they are used only to display blocks of JSON so that they may be conveniently copied to the Swagger UI.
Exercise 1: Services

Objective
Become familiar with publishing and unpublishing OGC services using the REST API.

- Publish an OGC Service
- View OGC Services
- Delete an OGC Service

Task 1: Publish an OGC Service

Publish an OGC Service.

1. Find the line in the Catalog Service API that reads **Services: APIs specific to services catalog items.**
2. Click **Services** or **Show/Hide** to expand the Services Catalog API operations.
3. While not strictly necessary for the exercise, using the GET command to get all OGC services will reveal that none are currently published.
4. Open the **POST** command for `/content/catalog/services.json`.
5. In the **resourceURL** field, type `http://MACHINENAME:80/erdas-apollo/vector/ATLANTA_VECTOR`, substituting the machine’s actual name for `MACHINENAME`. This name should be `HxGNLiveApollo`, but if you run into difficulties, check to be sure.
6. In the **basicauth** field, type `YWRtaW46YXBvbGxvMTIz` to provide the basic authentication encoded value. Do not alter the **reharvest** field.
7. Click **Try it out!**

Task 2: View OGC Services

View existing OGC Services.

1. Open the **GET** command for `/content/catalog/services.json` if you have not already.
2. Click **Try it out!**

The results in the response body should now include the ATLANTA VECTOR service that was published in the previous step.
3. In the results block of the Response Body, locate OGC:WFS. It will likely be toward the bottom of the Response Body. Copy its id. You will need it for the next task.

![Response Body](image)

**Task 3: Delete an OGC Service**

Unpublish an OGC Service.

1. Open the DELETE command under the Services endpoint.
2. Paste the id of the Service you intend to unpublish into the text field next to ID.
3. Click Try it out!

   This will unpublish the service. The 204 response is the only indication from this operation that it worked, however for further verification you can repeat task 2. You should see an empty results block in the response body.
Exercise 2: Services-Vector

Objective
Become familiar with managing vector services using the REST API.

- View all legacy vector services.
- Add a legacy vector service.
- Update a legacy vector service.
- Delete a legacy vector service.

Task 1: View All Legacy Vector Services

View legacy vector services.

1. Find the line in the Catalog Service API that reads Services-Vector: APIs specific to managing legacy vector services.
2. Click Services or Show/Hide to expand the Services Catalog API operations.
3. Click the GET operation at /services/vector to open it.
4. Click Try it out!
   The response body should include some values like ATLANTA_GML and WORLDWIDE.

Task 2: Add a Legacy Vector Service

Create a vector service.

1. Open the POST operation under Services-Vector.
2. Open the file on your machine at C:\DemoMaterials\Services-Vector\AddVectorService.json, preferably in Notepad++.
3. Copy the contents of that file into the POST body text field.
4. Click Try it out!
   In this case, you have created a service based on data in the file system, but a properly formatted POST body could also construct one from a database.
5. Repeat step 1. There should be a service named SV_SAMPLE.
Task 3: Update a Legacy Vector Service

Update a vector service.

1. Open the PATCH operation under Services-Vector.
2. In providerName, type the name of the newly created service SV_SAMPLE, unless you modified the post body in the previous task.
3. Click the Example block on the right to fill the Map text field with the template. Replace the title value of The new title with your preferred new title.
4. Click Try it out!

   The Response Body reports a successful update.

   You can verify further by repeating the GET operation under the Services endpoint, NOT the Services Vector endpoint. See Task 1 of the Services exercise. The new OGC Services endpoint can see this new service because we published it as part of Task 2 with the registerServiceInCatalog option set to true.

Task 4: Delete a Legacy Vector Service

Delete a vector service.

1. Open the DELETE operation under Services-Vector.
2. In providerName, type the name of the created Service.
3. Click Try it out!

   The 204 result reports success. You can verify further by repeating Task 1.
About Us

Hexagon Geospatial helps you make sense of the dynamically changing world. We enable you to envision, experience and communicate geographic information. Our technology provides you the form to design, develop and deliver solutions that solve complex, real-world challenges. Ultimately, this is realized through our creative software products and platforms.

CUSTOMERS. Globally, a wide variety of organizations rely on our products daily including local, state and national mapping agencies, transportation departments, defense organizations, engineering and utility companies, and businesses serving agriculture and natural resource needs. Our portfolio enables these organizations to holistically understand change and make clear, reliable decisions.

TECHNOLOGY. Our priority is to deliver products, platforms and solutions that make our customers successful. Hexagon Geospatial is focused on developing technology that displays and interprets information in a personalized, meaningful way. We enable you to transform location-based content into dynamic and useable business information that creatively conveys the answers you need.

PARTNERS. As an organization, we are partner-focused, working alongside our channel to ensure we succeed together. We provide the right platforms, products, and support to our business partners so that they may successfully deliver sophisticated solutions for their customers. We recognize that we greatly extend our reach and influence by cultivating channel partner relationships both inside and outside of Hexagon.

TEAM. As an employer, we recognize that the success of our business is the result of our highly motivated and collaborative staff. At Hexagon Geospatial, we celebrate a diverse set of people and talents, and we respect people for who they are and the wealth of knowledge they bring to the table. We retain talent by fostering individual development and ensuring frequent opportunities to learn and grow.

HEXAGON. Hexagon’s solutions integrate sensors, software, domain knowledge and customer workflows into intelligent information ecosystems that deliver actionable information. They are used in a broad range of vital industries.

Hexagon (Nasdaq Stockholm: HEXA B) has approximately 17,000 employees in 46 countries and net sales of approximately 3.3bn USD. Learn more at hexagon.com and follow us @HexagonAB.
Copyright and Terms of Use

Copyright

© 2017 Hexagon AB and/or its subsidiaries and affiliates. All rights reserved. Hexagon has registered trademarks in many countries throughout the world. Visit the Trademarks Page for information about the countries in which the trademarks are registered. See Product Page and Acknowledgments for more information.

Product Documentation Terms of Use

PLEASE READ THESE TERMS CAREFULLY BEFORE USING HEXAGON GEOSPATIAL’S DOCUMENTATION (“DOCUMENT”). USE OF THIS DOCUMENT INDICATES ACCEPTANCE OF THIS AGREEMENT WITHOUT MODIFICATION. IF YOU DO NOT AGREE TO THE TERMS HEREOF (“TERMS”), DO NOT USE THIS DOCUMENT.

Use Of This Document

All materials in this Document are copyrighted and any unauthorized use may violate worldwide copyright, trademark, and other laws. Subject to the terms of this Agreement, Hexagon Geospatial (a Division of Intergraph Corporation) and Intergraph’s subsidiaries (“Intergraph”) hereby authorize you to reproduce this Document solely for your personal, non-commercial use. In consideration of this authorization, you agree to retain all copyright and other proprietary notices contained therein. You may not modify the Materials in any way or reproduce or publicly display, perform, or distribute or otherwise use them for any public or commercial purpose, except as specifically authorized in a separate agreement with Hexagon Geospatial.

The foregoing authorization specifically excludes content or material bearing a copyright notice or attribution of rights of a third party. Except as expressly provided above, nothing contained herein shall be construed as conferring by implication, estoppel or otherwise any license or right under any copyright, patent or trademark of Hexagon Geospatial or Intergraph or any third party.

If you breach any of these Terms, your authorization to use this Document automatically terminates. Upon termination, you will immediately destroy any downloaded or printed Materials in your possession or control.

Disclaimers

ALL MATERIALS SUPPLIED HEREUNDER ARE PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. Hexagon Geospatial does not warrant that the content of this Document will be error-free, that defects will be corrected, or that any Hexagon Geospatial Website or the services that make Materials available are free of viruses or other harmful components.

Hexagon Geospatial does not warrant the accuracy and completeness of this Document. Hexagon Geospatial may make changes to this Document at any time without notice.
Limitation Of Liability

IN NO EVENT SHALL HEXAGON GEOSPATIAL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY YOU OR ANY THIRD PARTY, WHETHER IN AN ACTION IN CONTRACT OR TORT, ARISING FROM YOUR ACCESS TO, OR USE OF, THIS DOCUMENT.

Indemnification

You agree to defend, indemnify, and hold harmless Hexagon Geospatial, its officers, directors, employees, and agents from and against any and all claims, liabilities, damages, losses or expense, including reasonable attorneys’ fees and costs, arising out of or in any way connected with your access to or use of this Document.

Use Of Software

Use of software described in this Document is subject to the terms of the end user license agreement that accompanies the software, if any. You may not download or install any software that is accompanied by or includes an end user license agreement unless you have read and accepted the terms of such license agreement. Any such software is the copyrighted work of Hexagon Geospatial, Intergraph or its licensors. Portions of the user interface copyright 2012-2017 Telerik AD.

Links To Third Party Websites

This Document may provide links to third party websites for your convenience and information. Third party websites will be governed by their own terms and conditions. Hexagon Geospatial does not endorse companies or products to which it links.

Third party websites are owned and operated by independent parties over which Hexagon Geospatial has no control. Hexagon Geospatial shall not have any liability resulting from your use of the third party website. Any link you make to or from the third party website will be at your own risk and any information you share with the third party website will be subject to the terms of the third party website, including those relating to confidentiality, data privacy, and security.

Trademarks

The trademarks, logos and service marks ("Marks") displayed in this Document are the property of Hexagon Geospatial, Intergraph or other third parties. Users are not permitted to use Marks without the prior written consent of Hexagon Geospatial, Intergraph or the third party that owns the Mark. "Intergraph" is a registered trademark of Intergraph Corporation in the United States and in other countries. Other brands and product names are trademarks of their respective owners.

Find additional trademark information.

Procedure For Making Claims Of Copyright Infringement

Notifications of claimed copyright infringement should be sent to Hexagon Geospatial by mail at the following address: Intergraph Corporation, Attn: Intergraph Legal Department, P.O. Box 240000, Huntsville, Alabama 35824.
US Government Restricted Right

Materials are provided with "RESTRICTED RIGHTS." Use, duplication, or disclosure of Materials by the U.S. Government is subject to restrictions as set forth in FAR 52.227-14 and DFARS 252.227-7013 et seq. or successor provisions thereto. Use of Materials by the Government constitutes acknowledgment of Hexagon Geospatial or Intergraph's proprietary rights therein.

International Use

You may not use or export Materials in violation of U.S. export laws and regulations. Hexagon Geospatial makes no representation that Materials are appropriate or available for use in every country, and access to them from territories where their content is illegal is prohibited.

Hexagon Geospatial provides access to Hexagon Geospatial international data and, therefore, may contain references or cross references to Hexagon Geospatial products, programs and services that are not announced in your country. These references do not imply that Hexagon Geospatial intends to announce such products, programs or services in your country.

The Materials are subject to U.S. export control and economic sanctions laws and regulations and you agree to comply strictly with all such laws and regulations. In addition, you represent and warrant that you are not a national of, or otherwise located within, a country subject to U.S. economic sanctions (including without limitation Iran, Syria, Sudan, Cuba, and North Korea) and that you are not otherwise prohibited from receiving or accessing the Materials under U.S. export control and economic sanctions laws and regulations. Hexagon Geospatial makes no representation that the Materials are appropriate or available for use in every country, and access to them from territories where their content is illegal is prohibited. All rights to use the Materials are granted on condition that such rights are forfeited if you fail to comply with the terms of this agreement.

Revisions

Hexagon Geospatial reserves the right to revise these Terms at any time. You are responsible for regularly reviewing these Terms. Your continued use of this Document after the effective date of such changes constitutes your acceptance of and agreement to such changes.

Applicable Law

This Document is created and controlled by Hexagon Geospatial in the State of Alabama. As such, the laws of the State of Alabama will govern these Terms, without giving effect to any principles of conflicts of law. You hereby irrevocably and unconditionally consent to submit to the exclusive jurisdiction of the United States District Court for the Northern District of Alabama, Northeastern Division, or the Circuit Court for Madison County, Alabama for any litigation arising out of or relating to use of this Document (and agree not to commence any litigation relating thereto except in such courts), waive any objection to the laying of venue of any such litigation in such Courts and agree not to plead or claim in any such Courts that such litigation brought therein has been brought in an inconvenient forum. Some jurisdictions do not allow the exclusions or limitations set forth in these Terms. Such exclusions or limitations shall apply in all jurisdictions to the maximum extent allowed by applicable law.

Questions

Contact us with any questions regarding these Terms.