Release guide

LuciadCPillar 2023.1

14 December 2023

Contents

About this release

The 2023.1 release of LuciadCPillar adds a set of tools to enrich your operational view and visualize data as required for your application domain. New raster, line and point styling options allow you to create the required presentation of your business data. Moreover, LuciadCPillar now also supports the visualization of Longitude-Latitude grids, and of the Military Grid Reference System (MGRS), including support for parsing and formatting coordinates in the both formats. Note that all new features work on desktop and Android and are available via all supported programming languages (C++, C# and Java for Android/Kotlin). This release also includes some format-specific improvements.

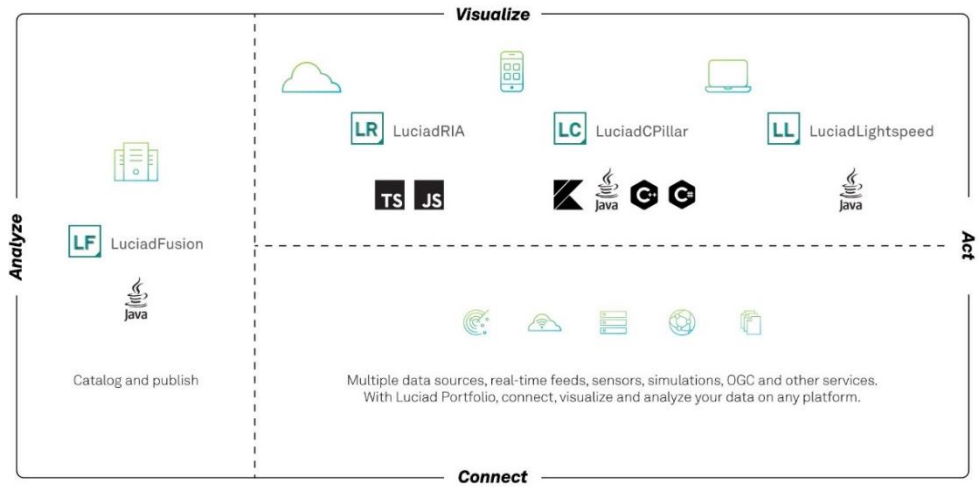


Figure 1 The Luciad portfolio

Benefits of new features

This section provides an overview and introduction of new features that are available in LuciadCPillar 2023.1. The new features are available for all the supported platforms and programming languages currently offered for LuciadCPillar.

Offer an accurate and rich common operational view

This release brings exciting extra possibilities for styling your feature and raster data. Apply your business-specific symbology and use 3D icons to offer a better understanding of the situation.

Apply any stroke and stroke pattern to your feature data

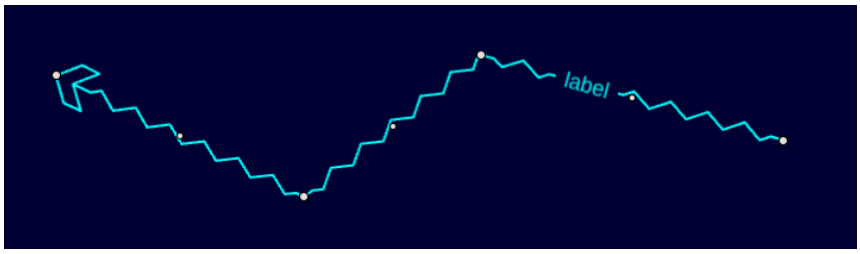


Figure 2: An illustration of a line stroked with a repeating pattern and with a label as decoration

The complex stroking capability makes it possible to visualize lines by stroking repeating graphical patterns and decorations along the line. An example is a zigzag line with an arrow (See Figure 2) and

a text decoration. LuciadCPillar offers a new API for this purpose. Using this powerful API, you can visualize a line in virtually any way you want. This also applies to the outline of polygons.

Sample code to get you started

A new sample has been added specifically for the Complex Strokes capability. In the Complex Strokes sample, you will see various shapes and lines that are stroked using the Complex Stroke API, grouped into themes. You can see how the shape is styled in the sample code.

You can select an object to start editing it. See Figure 3 and 4 for an illustration of this capability.

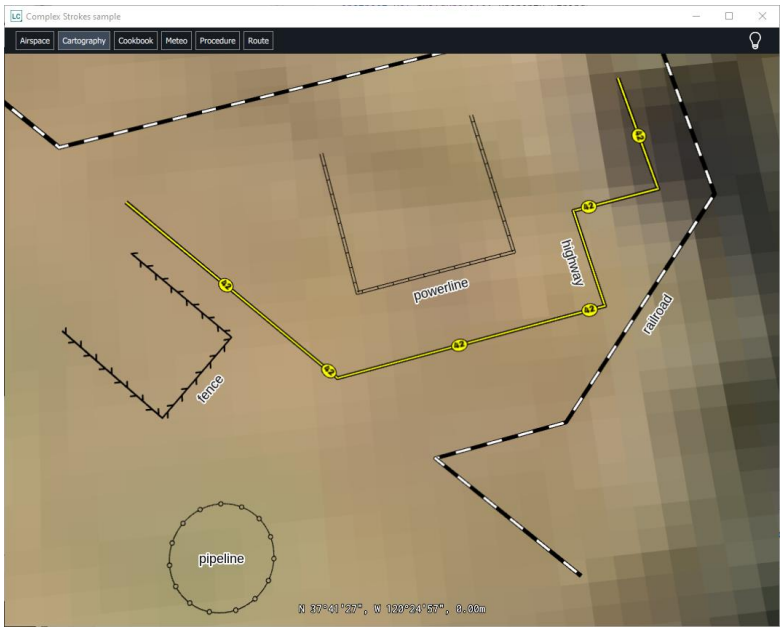


Figure 3: The complex strokes sample illustrates various line patterns.

Figure 4: LuciadCPillar offers complex line strokes on all platforms, including Android.

Documentation

The “Step-by-step guide to complex strokes” explains how complex stroking works and shows how you can build a complicated complex stroke in a sequence of steps. It applies the main complex stroking principles along the way.

GETTING STARTED | DOCUMENTATION | SAMPLES | API REFERENCE | RELEASE NOTES | PREREQUISITES | CONTACT | C++

LuciadCPillar / Maps / Visualizing feature data / A step-by-step guide to complex strokes

A step-by-step guide to complex strokes

What are complex strokes?

The complex stroking capability makes it possible to visualize lines constructed from repeating patterns and decorations. An example is a zigzag line with an arrow, or with text decorations. LuciadCPillar offers the `ComplexStrokeLineStyle` API for this purpose. Using this powerful API, you can create a line in virtually any way you want.

This article explains how complex stroking works, and shows how you can build a complicated complex stroke in a sequence of steps. It applies the main complex stroking principles along the way. The end result is the following line:

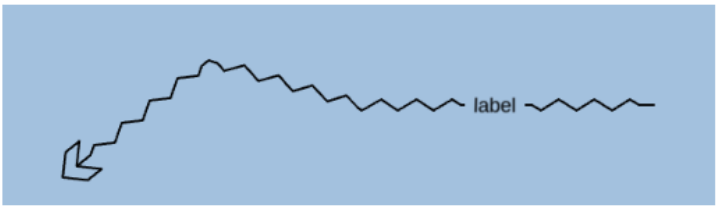


Figure 1. End result stroke

How complex strokes work

You can compose a complex stroke of the following elements:

- **Decorations** Small non-repeating elements that you add at a specific location on a line, such as arrow heads, text decorations, and icons.
- **Regular stroke pattern** A pattern that you repeat along the entire line, zigzag lines, wavy lines or dash patterns, for example.
- **Fallback stroke pattern**: like the regular stroke pattern, but you use it in places where you can't paint decorations or regular stroke patterns. It's typically a simple pattern, like a plain line.

Sometimes, a complex stroke is interrupted, and partly skipped:

- **Obstacles** When a decoration is already present somewhere along the line, it forms an obstacle for all other decorations, regular strokes or fallback stroke. At the location of the obstacle, the decoration, or the obstructed part of the regular/fallback stroke pattern, is dropped. It isn't painted. Note that you can tweak this behavior. We also show how to do that in the following sections.
- **Corners** When a decoration or regular stroke patterns crosses a sharp corner of the line, it gets dropped.

The next sections show how to create and compose decorations, regular stroke patterns and fallback stroke patterns in an example.

Figure 5: The “Step-by-step guide to complex strokes” guides you step by step.

Use 3D models as icons, for the representation of static or moving units

You can now use 3D icons encoded as glTF to style points.

You can configure the following aspects of the 3D icon:

- Orientation
- Rotation
- Scale
- Translation
- Transparency

Sample code to get you started

The tutorial “Visualize glTF data” includes example code for a 3D icon painter.

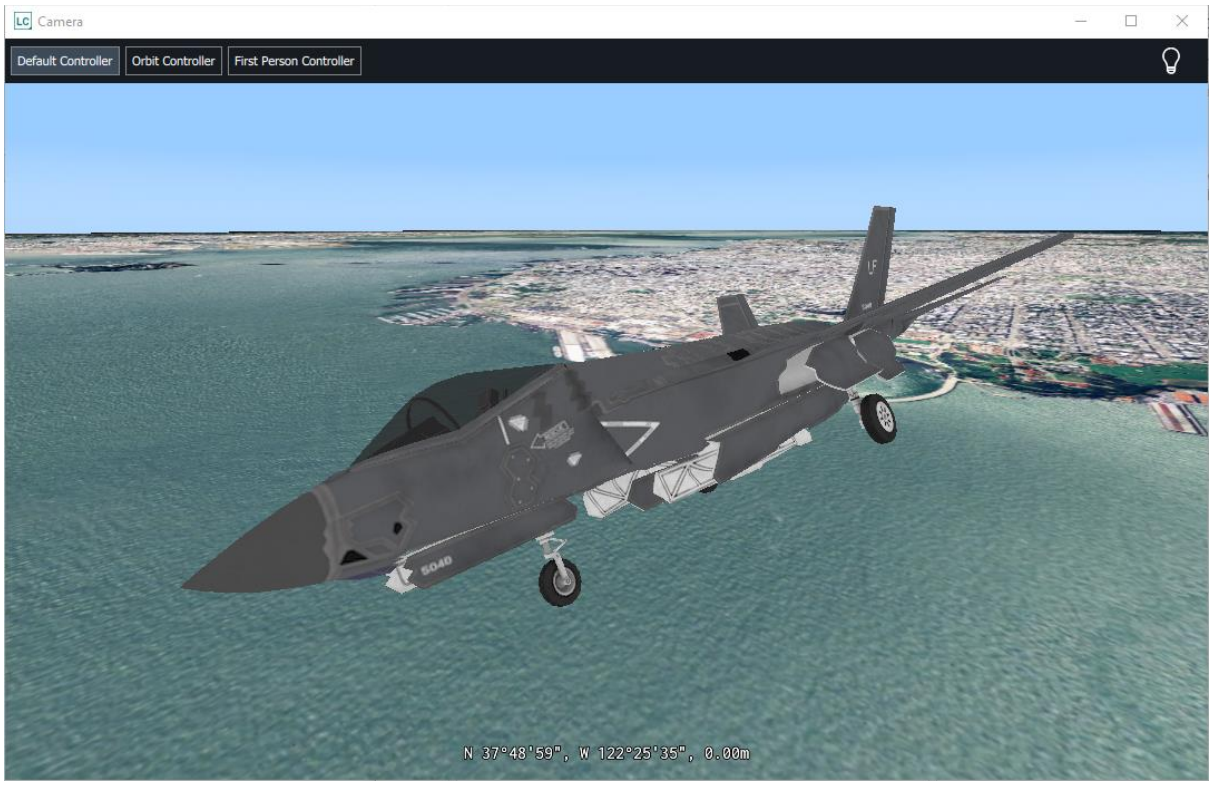


Figure 6: A 3D icon depicting an airplane is used to style a moving track.

Documentation

There is a new tutorial, “Visualize glTF data,” explaining how to work with 3D models in the glTF data format. This tutorial and an accompanying FAQ can be found under the subsection “Data Formats: glTF.”

Finetune the appearance of your raster layers

Imagery data is typically brought into the operational picture as background or reference data not to be altered by users. However, depending on the other information in the view, the combined picture can be enhanced by slightly adapting the rendering of the image. With a different contrast or opacity, business data overlaid on the image may be clearer for the human observer. The LuciadCPillar RasterStyle object groups a set of styling settings allowing you to alter contrast and brightness opacity and set a modulation color.

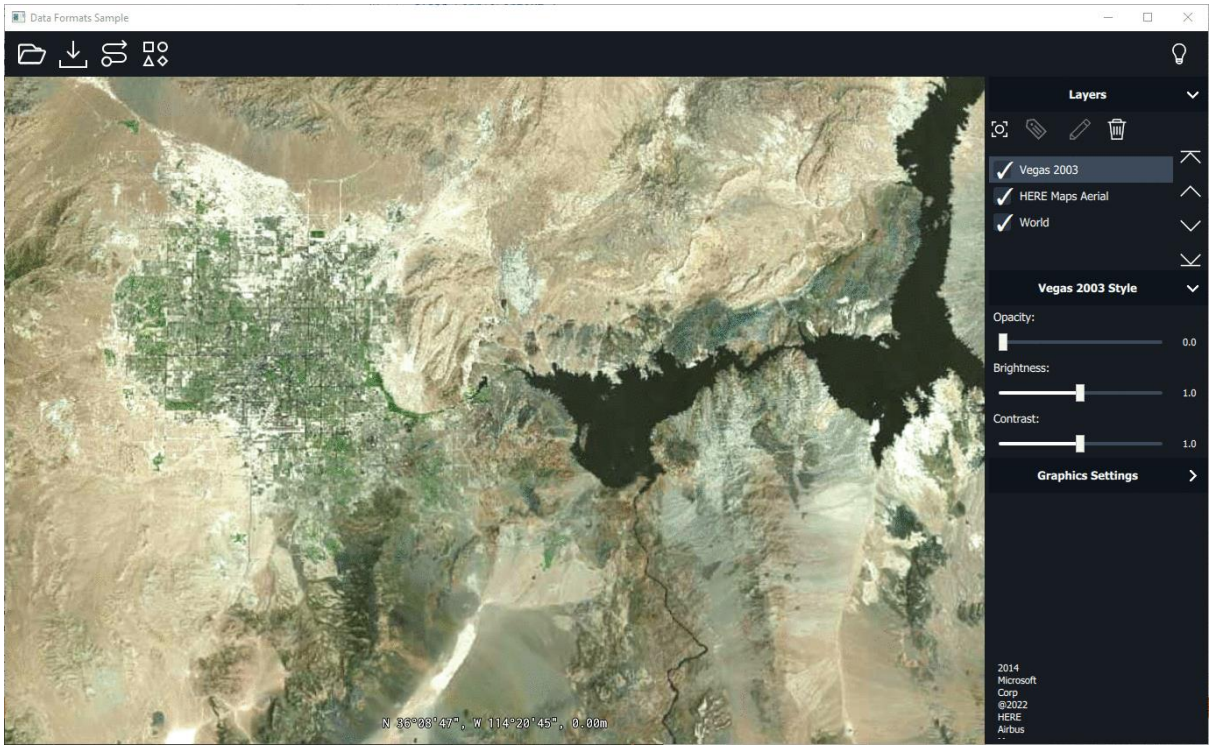


Figure 7: The opacity of image data can be adapted using the new layer control in the LuciadCPillar samples.

The layer control in the LuciadCPillar samples has been extended with raster styling UI to illustrate how the raster styling API can be used in an interactive way.

Documentation

The article “Introduction to styling raster data” has been extended to reflect the new raster styling options.

Overlay your 2D or 3D view with a grid for improved situational awareness

You can now add Grid layers to the map. Grid layers don’t contain actual data, their purpose is to provide a visual, usually uniform reference that makes it easier to locate data. We included two types of grid layers: the traditional longitude-latitude (lon-lat) grid and the military MGRS grid.

Support for longitude-latitude grids

You can add a lon-lat grid as a separate layer to your map and have control over the styling of the grid lines and the labeling of the lines. Additionally, API was added to format longitude-latitude coordinates in a readable format.

Sample code to get you started

Most samples in the LuciadCPillar release now display a lon-lat grid by default. We feel it really makes a difference to orient yourself on the map!

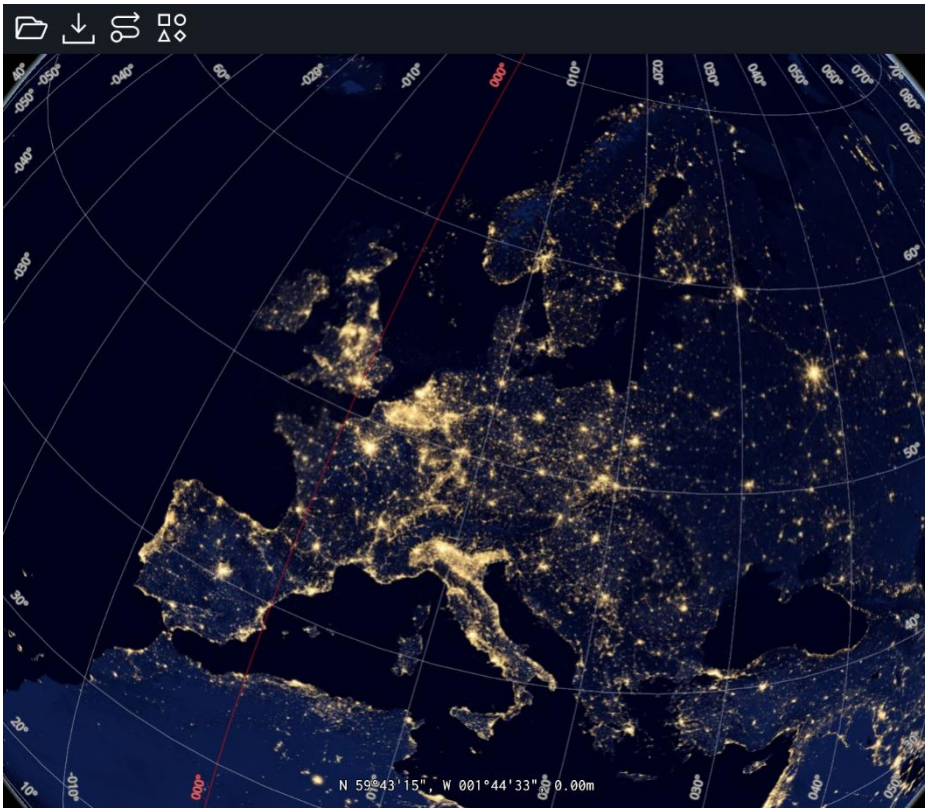


Figure 88: An illustration of a lon-lat grid overlaid on the map

Documentation

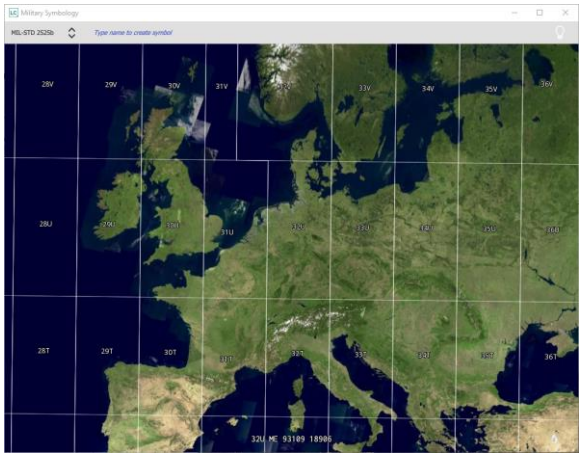
We added the “Visualizing a lon-lat grid” article that explains how to add a lon-lat grid layer to the map and how to style the grid. The “Formatting points locations as coordinates” article explains how to format lon-lat coordinates into a human-readable form.

Support for MGRS grids

The MGRS is a NATO geocoordinate standard for locating points on Earth. The MGRS is derived from the Universal Transverse Mercator (UTM) grid system and the Universal Polar Stereographic (UPS) grid system but uses a different labeling convention.

The new feature has two aspects. First, it offers the capability to parse and format coordinates according to the MGRS standard, and second, you can display an MGRS grid layer on a LuciadCPillar map.

Sample code to get you started



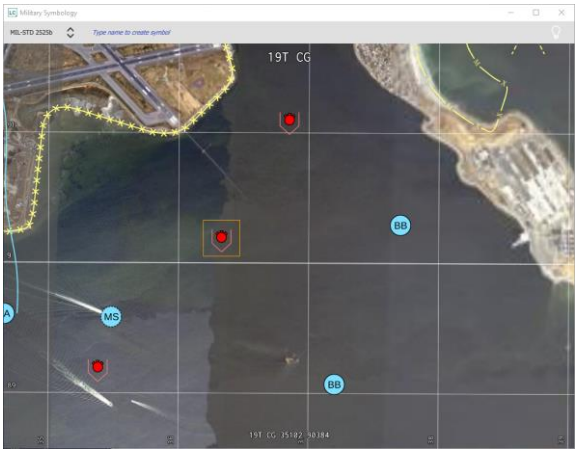


Figure 9 and Figure 10: An illustration of an MGRS grid overlaid on the map

The Military Symbology sample has been extended with an illustration of MGRS grid and MGRS coordinate formatting. The sample shows a coordinate readout where the mouse location is formatted as an MGRS coordinate instead of a Latitude-Longitude coordinate. The grid lines are labeled, and an overview level is shown at the top of the page when you zoom in to a large detail level.

Documentation

A new article “Working with MGRS” has been added to the documentation. It contains information on how to add a grid to the map, visualize an overview label as well as how to format coordinates to an MGRS formatted string.

HEXAGON

GETTING STARTED | DOCUMENTATION | SAMPLES | API REFERENCE | RELEASE NOTES | PREREQUISITES | CONTACT | C++

LuciadCPillar / Symbology / Military Grids / Working with MGRS

Working with MGRS

The Military Grid Reference System (MGRS) is the geo-coordinate standard used by NATO militaries for locating points on Earth. The MGRS is derived from the Universal Transverse Mercator (UTM) grid system and the Universal Polar Stereographic (UPS) grid system, but uses a different labeling convention.

Figure 1. The MGRS grid is included in the Symbology sample

Formatting and parsing points to/from MGRS

To format and parse a `Point` to/from MGRS strings, use an `MgrsFormat`.

Program: Use an `MgrsFormat` to format points to MGRS strings

```
MgrsFormat mgrsFormat = MgrsFormat::newBuilder()
    .precision(MgrsFormatPrecision::Precision1M)
    .formatType(MgrsFormatType::Mgrs)
    .zoneSeparator(" ")
    .coordinateSeparator(" ")
    .build();
auto wgs84 = CoordinateReferenceProvider::create("EPSG:4326");
auto point = Point(wgs84.value(), {5, 45, 0});
auto stringResult = mgrsFormat.format(point);

auto parsed = mgrsFormat.parse(stringResult.value());
```

For example, you can use this to show MGRS coordinates at the mouse location. See the Symbology sample for a demonstration.

Visualizing an MGRS grid

Figure 11: The new article "Working with MGRS" is your starting point in the documentation.

GeoPackage editing support and persistence API

LuciadCPillar now offers support to edit an existing GeoPackage file. It is possible to add, remove and change features.

The new LuciadCPillar persistence API gives you full control over when you persist these changes. This allows you to take the performance impact on your application into consideration. For example, when you are editing a polyline on the map, you most likely want to save the change to the GeoPackage file when you're done editing. Saving every change, even while you are dragging your finger over your screen to move the polyline around, may be detrimental to the application's performance, but it is still possible if that is really what you need.

Documentation

An article on "GeoPackage Feature Editing" was added to the documentation, as well as an article on "Working with feature models with save support."

Support for .NET standard 2.0

LuciadCPillar has been upgraded to support .NET standard 2.0, which allows you to integrate LuciadCPillar in modern .NET projects. We also increased the minimal Visual Studio version to 2019. If you have any questions or concerns related to this change, please contact the Luciad Product Management team at product.management.luciad.gsp@hexagon.com.

Product license versioning

Starting from the 2023.1 release, you only need a new product license for a major LuciadCPillar product version.

More specifically, for version 2022.0 and 2022.1, you still need separate licenses. If you use your license file for LuciadCPillar version 2023.0 with LuciadCPillar version 2023.1, it will work. Of course, both product versions must have matching configurations, with an equivalent product name, product tier and options list.

Other improvements

- **Support .ovr files for GeoTIFF data:** .ovr files contain an external pyramid with lower-resolution levels for the accompanying GeoTIFF file. If present, LuciadCPillar will use the information in the .ovr files, offering a smoother rendering. Note that the pyramid information can also be embedded within a GeoTIFF file.
- **LuciadCPillar now supports elevation data in the GeoTIFF format** with the values encoded as 16-bit unsigned integers.

About Hexagon

Hexagon is the global leader in digital reality solutions, combining sensor, software and autonomous technologies. We are putting data to work to boost efficiency, productivity, quality and safety across industrial, manufacturing, infrastructure, public sector and mobility applications.

Our technologies are shaping production and people-related ecosystems to become increasingly connected and autonomous — ensuring a scalable, sustainable future.

Hexagon's Safety, Infrastructure & Geospatial division improves the resilience and sustainability of the world's critical services and infrastructure. Our solutions turn complex data about people, places and assets into meaningful information and capabilities for better, faster decision-making in public safety, utilities, defense, transportation and government.

Hexagon (Nasdaq Stockholm: HEXA B) has approximately 24,000 employees in 50 countries and net sales of approximately 5.2bn EUR. Learn more at hexagon.com and follow us @HexagonAB.

official English version. Additionally, some text, graphics, PDF documents and other accompanying material may not have been translated.

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 does not endorse companies or products to which it links.

Third-party websites are owned and operated by independent parties over which Hexagon has no control. Hexagon 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.

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

Revisions

Hexagon 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.

Questions

Contact us with any questions regarding these terms.