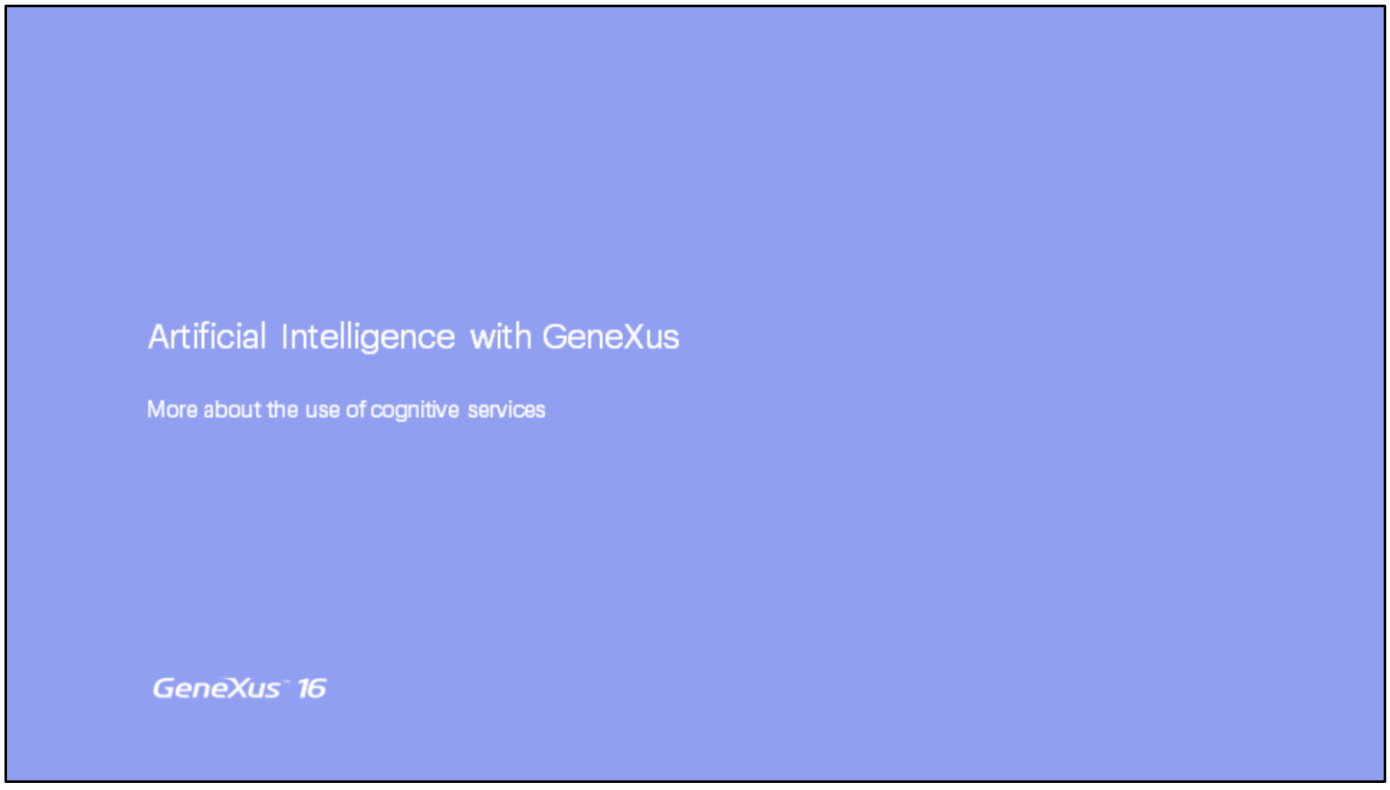
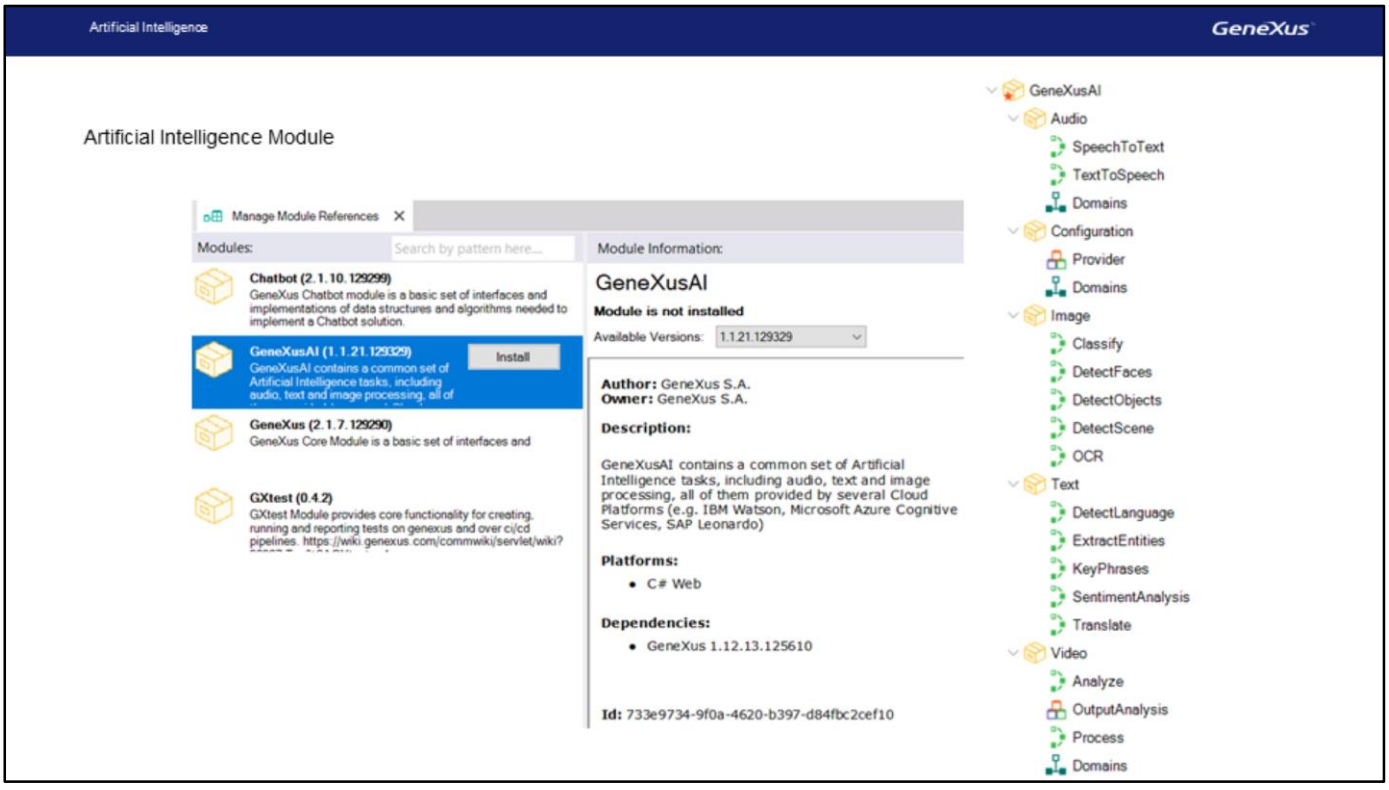
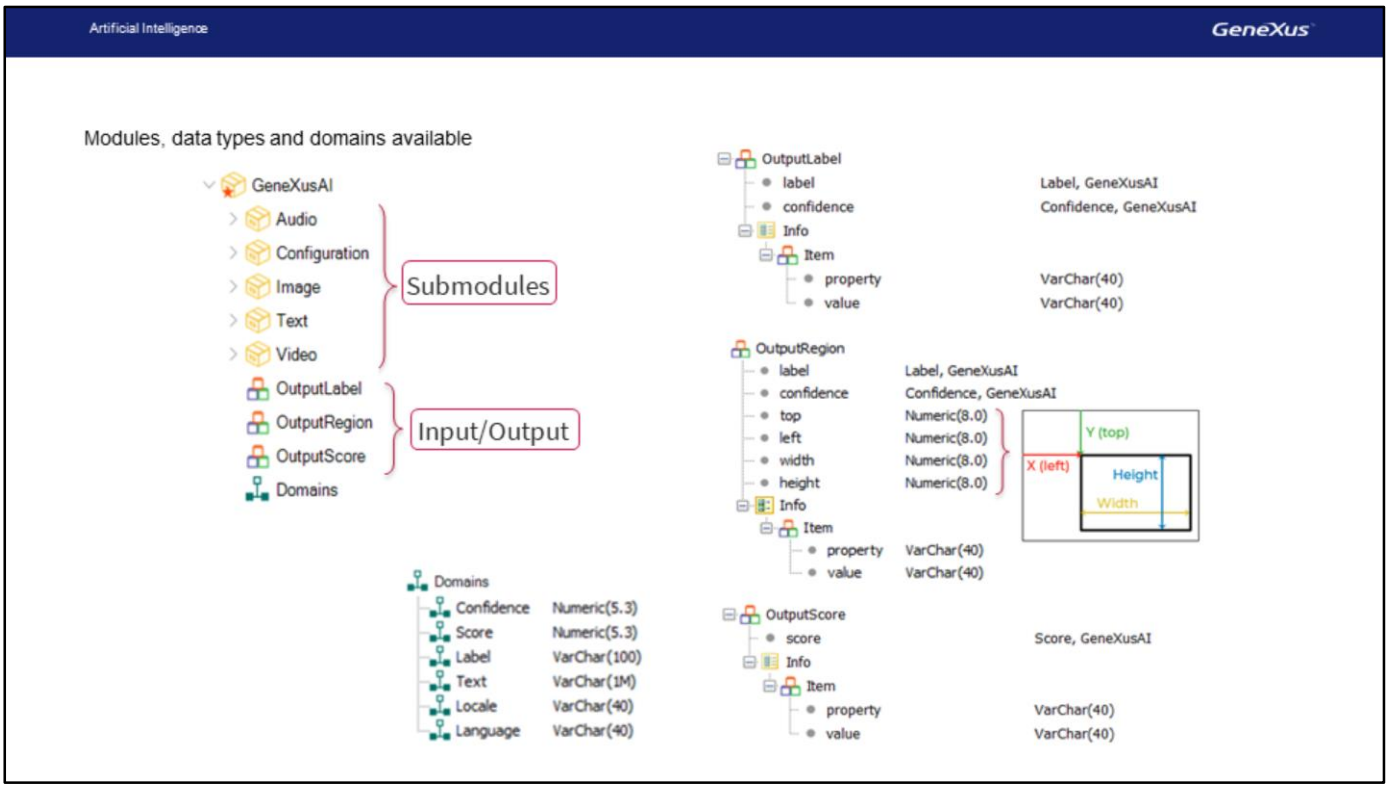


In the introduction to this topic we saw the basic information of what we can do with GeneXus in relation to Artificial Intelligence.
Here are the implementation details of how to invoke the methods and a brief example developed.



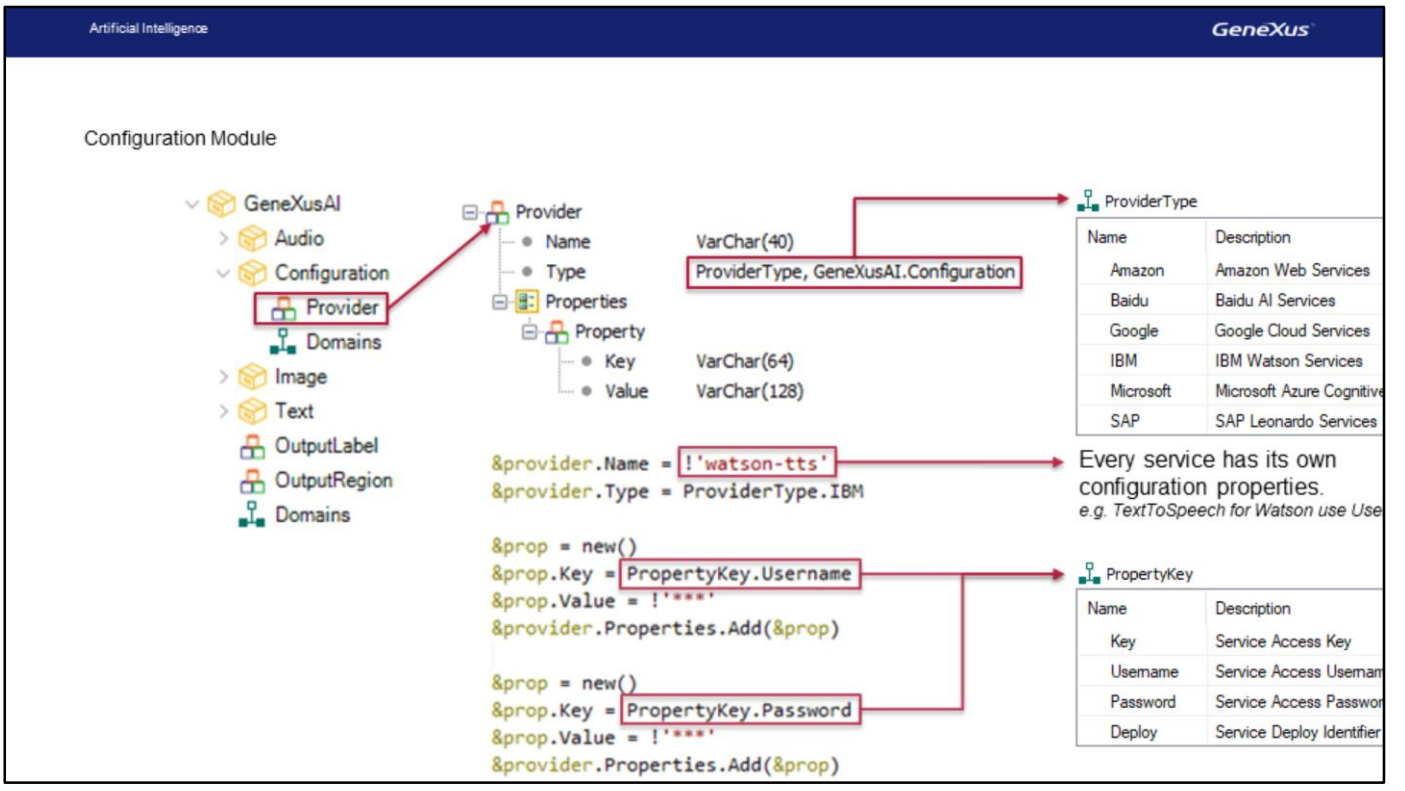
The way to use artificial intelligence algorithms is to install the GeneXusAI module, which includes 4 sub-modules with functions to work with Audio, Image, Text and Video and the Configuration module to define the service provider and access credentials.



In addition to the five sub-modules that we will see in detail below, we also have structures and domains to handle the input/output of AI tasks, each with a defined semantics.

- **Confidence**
- **Score**
- **Label**
- **Text**
- **Locale** (language of a specific region - for example, Spanish from Spain)
- **Language** (language in the generic sense - e.g. Spanish, English)

In addition:
- **OutputLabel** provides a label and its confidence.
- **OutputRegion**, in addition to a label and its confidence, also indicates a rectangular region.
  - To visualize what it would be like, let's see this image (<X,Y> indicates the upper left corner of the rectangle, and then simply shows its height and width).
  - It also provides additional information that could be returned by the supplier.
- **OutputScore** represents a score value with additional data.



First, we'll talk about the **Configuration module**.

- Every GeneXusAI Procedure takes as input the configuration of a specific vendor.
- To do this, we must indicate a name, a type of provider in the ProviderType domain, and a group of properties.
  * The **ProviderType** contains the possible suppliers that can be selected.

For example:

1. First, we complete the first SDT level, indicating a name and a provider (in this case, IBM)
   - Note that reference is made to the cognitive service in the name because each service has its own configuration properties.

2. Then we complete the second level of the SDT; that is, its properties.
   - First, we enter a Username
   - Next, we enter a Password.

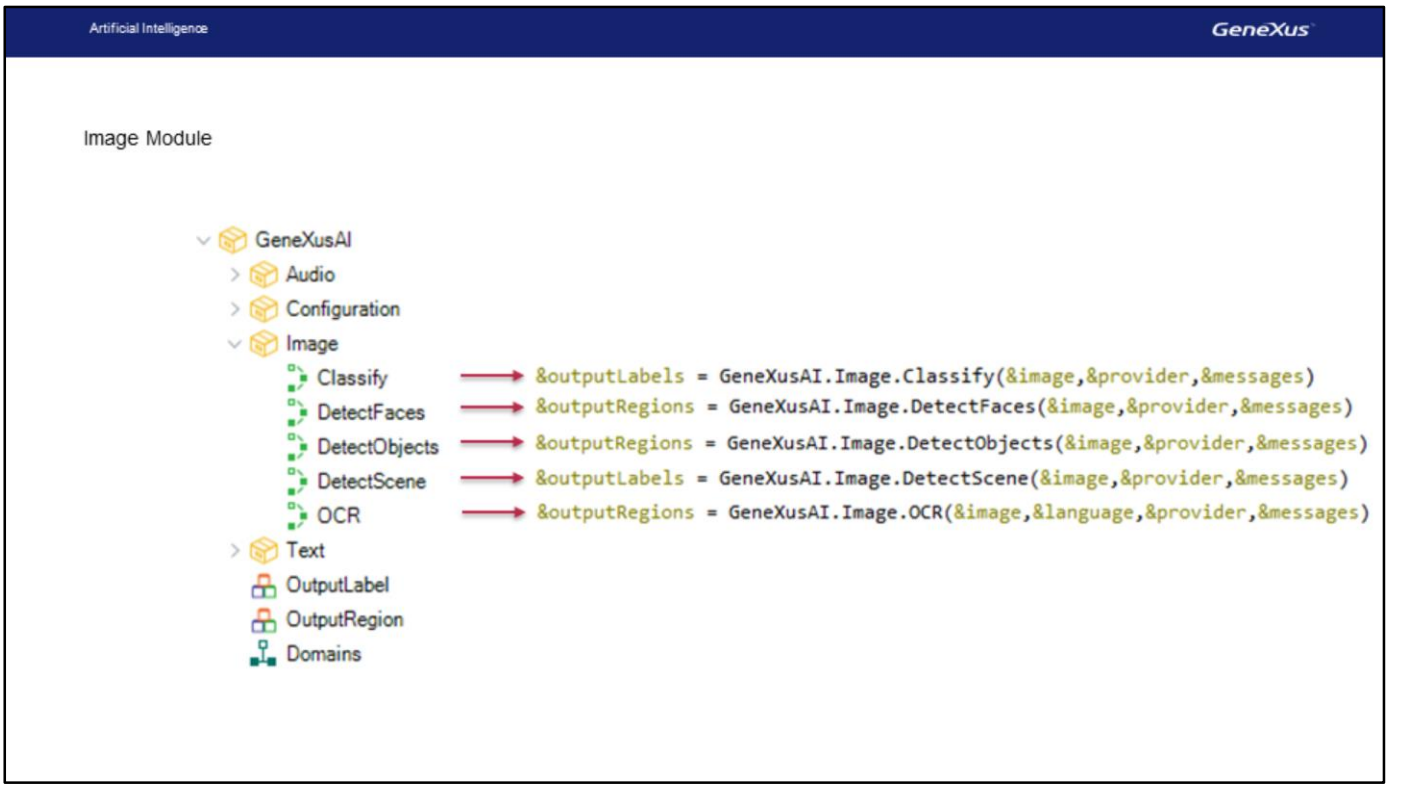
This is because TextToSpeech for IBM needs Username/Password to work, but other services may require an API key. To find out what properties can be configured we use the PropertyKey enumerated domain.

Audio Module

Now let's start looking at the rest of the modules.

First, the **Audio module** allows handling this type of multimedia content.

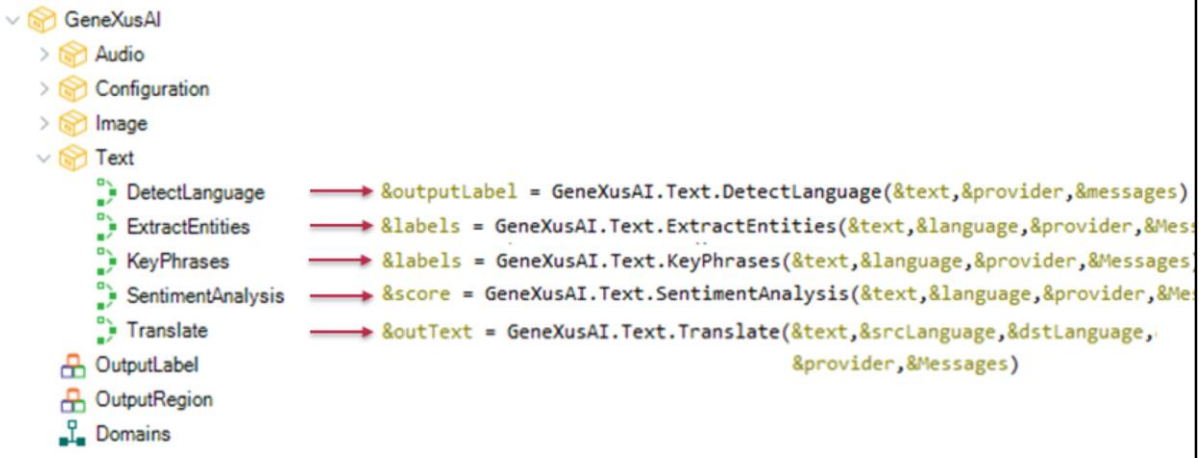
- **SpeechToText**: It allows us to transcribe audio to text.
  * Note how audio, regional language, provider settings, and messages are received. With this last parameter we will be able to handle errors that may occur during prototyping or execution.
  – The specification can be found in the Wiki.
- **TextToSpeech**: It allows us to synthesize text to audio.
  * Similarly, it receives a text, a type of voice, a regional language, etc. and returns an audio with the spoken text.



In the **Image module** we find:

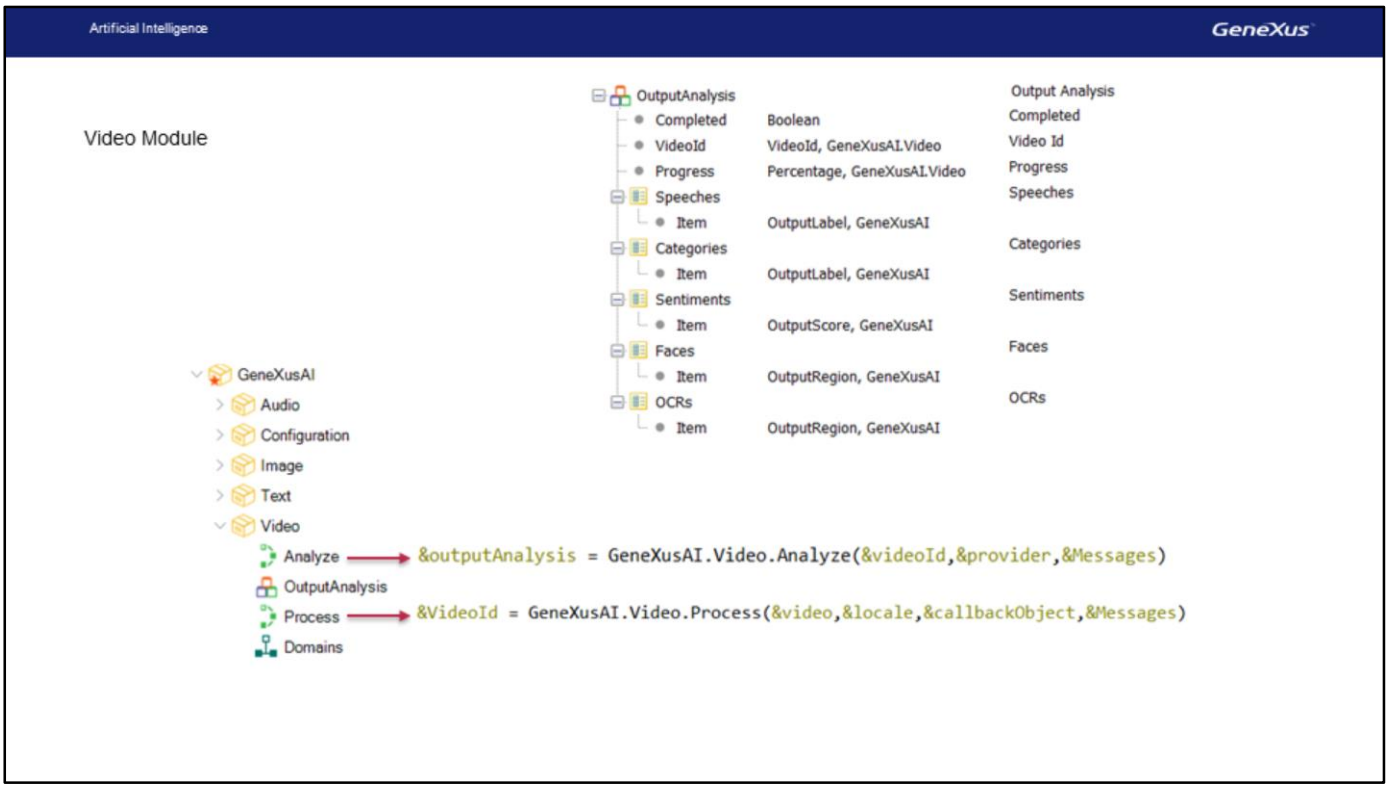
- **Classify**: Given an image, it returns a series of categories that describe the image (that's why its output is OutputLabels – in plural).
- **DetectFaces**: Given an image, it recognizes faces and can give us generic information about each one (e.g. gender, estimated age, etc.). Its output is OutputRegions (also in plural) because it indicates the position of each face in a rectangle.
- **DetectObjects**: Similar to the previous case but for objects.
- **DetectScene**: It describes the scenarios of an image (for example, if it is a city, beach, desert, etc.).
- **OCR**: It's an acronym for Optical Character Recognition, and allows us to extract text from an image.

Text Module



The **Text module** allows us to handle aspects of natural language, and is composed of:

- **DetectLanguage:** It allows us to detect a language from written text. Its output is OutputLabel as it not only indicates the language, but also a confidence level of detection.
- **ExtractEntities:** It extracts entities from text, e.g. organizations, cities, names of people, etc.
- **KeyPhrases:** It allows us to extract the most relevant phrases (or words) from the text.
- **SentimentAnlysis:** It allows us to infer the level of negativity or positivity of a text in 0 and 1, being 0.5 the neutral feeling.
- **Translate:** It allows us to translate text from a source language to a target language.



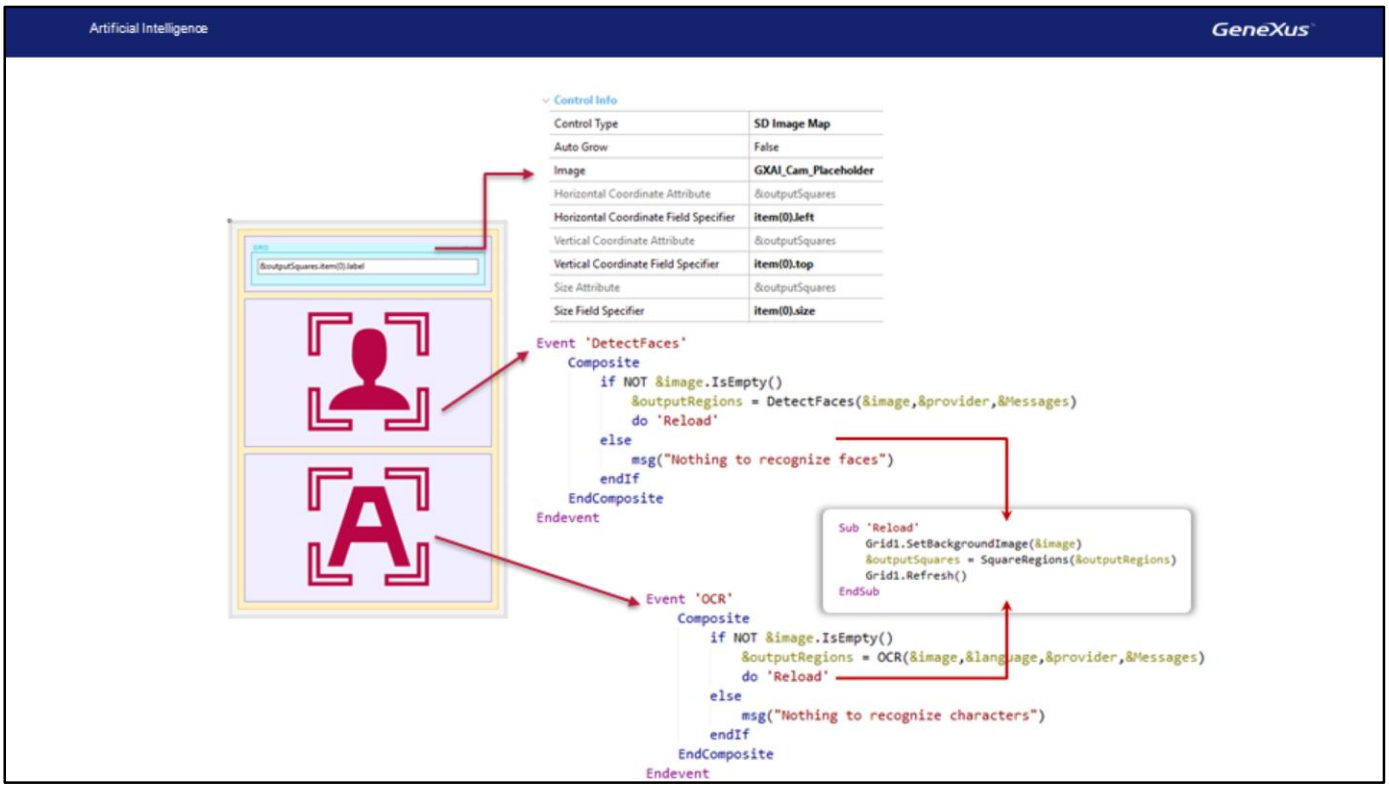
The **Video module** allows us to process a video and analyze it to detect speech, categories of objects, feelings, faces and perform optical character recognition. It's made up of:

- **Analyze:** It analyzes a video after it has been processed by the Process procedure.
- **Process:** It processes a video in synchronous form (in a process that periodically verifies the state (polling) or asynchronous form, defining a callback object, i.e. an object that is invoked every time something changes in the video process).

Detect faces and OCR example

DEMO

Let's see a simple example for Smart Devices.



Here we see how the previous example was implemented.

1. This is the layout associated with that screen. We have a grid, and two buttons with the images we saw in the animation.
2. The Grid is of SD Image Map type to be able to “draw the rectangles” (actually, for the moment, we can draw squares).
3. The first button has an associated event that verifies that the image is not empty and invokes the GeneXusAI DetectFaces procedure.
4. Similarly, the second button invokes the OCR procedure.

The ‘Reload’ subroutine that we see there simply sets the background image, gets the squares from the rectangles of the outputRegion, and finally performs a grid refresh.

GeneXus™

The power of doing.

More videos	training.genexus.com
Documentation	wiki.genexus.com
Certifications	training.genexus.com/certifications