GXF Core concepts

Here is a list of the key GXF terms used in this section:

- **Applications** are built as compute graphs.
- **Entities** are nodes of the graph. They are nothing more than a unique identifier.
- **Components** are parts of an entity and provide their functionality.
- **Codelets** are special components which allow the execution of custom code. They can be derived by overriding the C++ functions `initialize`, `start`, `tick`, `stop`, `deinitialize`, and `registerInterface` (for defining configuration parameters).
- **Connections** are edges of the graph, which connect components.
- **Scheduler and Scheduling Terms**: components that determine how and when the `tick()` of a Codelet executes. This can be single or multithreaded, support conditional execution, asynchronous scheduling, and other custom behavior.
- **Memory Allocator**: provides a system for allocating a large contiguous memory pool up-front and then reusing regions as needed. Memory can be pinned to the device (enabling zero-copy between Codelets when messages are not modified) or host, or customized for other potential behavior.
- **Receivers, Transmitters, and Message Router**: a message passing system between Codelets that supports zero-copy.
- **Tensor**: the common message type is a tensor. It provides a simple abstraction for numeric data that can be allocated, serialized, sent between Codelets, etc. Tensors can be rank 1 to 7 supporting a variety of common data types like arrays, vectors, matrices, multi-channel images, video, regularly sampled time-series data, and higher dimensional constructs popular with deep learning flows.
- **Parameters**: configuration variables used by the Codelet. In GXF applications, they are loaded from the application YAML file and are modifiable without recompiling.

In comparison, the core concepts of the Holoscan SDK can be found here.