CLAIMS

1. A system for managing entity life information; comprising:

an individual life entity platform comprising one or more processors or microprocessors in electronic communication with at least one database on a memory storage device, wherein the one or more processors or microprocessors are programmed to:

receive at a payload feeder, over a telecommunications network, data input messages containing entity care information from one or more data sources, wherein at least a portion of the data input messages are in different types and/or formats;

convert a data input message into a payload manifest schema comprising one or more observables, said observables comprising discrete data key-value pairs matching parsed data from the data input message to semantic ontology definitions;

transform the payload manifest schema to a curated manifest, said curated manifest comprising a persisted multi-JSON semantic normalized representation of select entity care data; and

send the curated manifest for storage in said at least one database;

wherein the platform has a system architecture comprising a plurality of services components, said plurality of service components comprising a payload service component, a manifest curator service component, a data refinement service component, an identity service component, a persisted contextual best record view service component, a knowledge engine service component, and an ontology service component;

wherein the system architecture is organized by one or more cooperating monoliths, wherein each of said one or more cooperating monoliths comprises two or more of said plurality of service components and a monolith-specific internal bus providing communications between the service components within the corresponding cooperating monolith; and

wherein the system architecture further comprises one or more external bus, said one or more external bus providing communications between the payload feeder and said one or more cooperating monoliths.

2. The system of claim 1, wherein the curated manifest is immutable.

3. The system of claim 1, wherein there is a single curated manifest created from a single data input message.

4. The system of claim 1, wherein the curated manifest has a structure comprising a header section, an actor section, an item section, and an observation section,

5. The system of claim 4, wherein the one or more processors or microprocessors are programmed to create an observation component for each observable data key-value pair, create an observation by grouping one or more observation components, and add the observation to the observation section of the manifest.

6. The system of claim 5, wherein the one or more processors or microprocessors are programmed to identify one or more actors in actor-related observations, perform an identity matching process to identity an entity corresponding to said one or more actors, and enter actor and entity data in the actor section of the curated manifest.

7. The system of claim 5, wherein the one or more processors or microprocessors are programmed to identify one or more items in item-related observations, and enter item data in the item section of the curated manifest.

8. The system of claim 1, wherein the one or more processors or microprocessors are programmed to receive, over a telecommunications network, a data request, and to provide a contextually best record view from the best data in one or more curated manifests responsive to the data request, wherein the best data is determined from contextual criteria based on the requested context.

9. The system of claim 8, wherein contextual criteria comprises data source.

10. The system of claim 8, wherein contextual criteria comprises data status.

11. The system of claim 8, wherein the contextually best record view is created on demand.

12. The system of claim 8, wherein the contextually best record view is a previously-created persisted contextually best record view.

13. The system of claim 12, wherein the persisted contextually best record view is updated in real-time as curated manifests are created.

14. The system of claim 1, wherein the receive function, convert function, and transform function are carried out with multiple process threads, wherein each thread corresponds to a specific function, and multiple functions can run simultaneously.

15. A method for managing entity life information; comprising:

one or more processors or microprocessors in electronic communication with at least one database on a memory storage device in an individual life entity platform, wherein the one or more processors or microprocessors are programmed to:

receiving, by one or more processors or microprocessors in electronic communication with at least one database on a memory storage device, over a telecommunications network, one or more data input messages containing entity care information from one or more data sources, wherein at least a portion of the data input messages are in different types and/or formats;

converting a selected data input message into a payload manifest schema comprising one or more observables, said observables comprising discrete data key-value pairs matching parsed data from the data input message to semantic ontology definitions;

transforming the payload manifest schema to a curated manifest, said curated manifest comprising a multi-JSON semantic normalized representation of select entity care data; and

sending the curated manifest for storage in said at least one database;

wherein the platform has a system architecture comprising a plurality of services components, said plurality of service components comprising a payload service component, a manifest curator service component, a data refinement service component, an identity service component, a persisted contextual best record view service component, a knowledge engine service component, and an ontology service component;

wherein the system architecture is organized by one or more cooperating monoliths, wherein each of said one or more cooperating monoliths comprises two or more of said plurality of service components and a monolith-specific internal bus providing communications between the service components within the corresponding cooperating monolith; and

wherein the system architecture further comprises one or more external bus, said one or more external bus providing communications between the payload feeder and said one or more cooperating monoliths.

16. The method of claim 15, wherein the curated manifest is immutable, and there is a single curated manifest created from a single data input message.

17. The method of claim 15, wherein the curated manifest has a structure comprising a header section, an actor section, an item section, and an observation section, and further comprising the steps of:

creating an observation component for each observable data key-value pair, create an observation by grouping one or more observation components;

adding the observation to the observation section of the manifest;

identifying one or more actors in actor-related observations;

performing an identity matching process to identity an entity corresponding to said one or more actors;

entering actor and entity data in the actor section of the curated manifest;

identifying one or more items in item-related observations, and

entering item data in the item section of the curated manifest.

18. The method of claim 15, further comprising the steps of:

receiving, over a telecommunications network, a data request; and

providing a contextually best record view from the best data in one or more curated manifests responsive to the data request;

wherein the best data is determined from contextual criteria based on the requested context, said contextual criteria comprising data source and/or data status.

19. The method of claim 15, wherein the contextually best record view is a previously-created persisted contextually best record view updated in real-time as curated manifests are created.

20. The method of claim 15, wherein the process steps of receiving, converting, and transforming are carried out with multiple process threads, wherein each thread corresponds to a specific process, and multiple processes can run simultaneously.