Obix Data Model
Use Cases

General
- A single schema describes all Obix documents.
- An Obix file describes methods of an object including their input and output parameters.
- Vendors use Obix to describe C data structures and bit strings, object oriented components, and dynamic data objects such as Javascript or python.
- A server merges AnalogInputCore, AnalogReliability, and AnalogAlarming using multiple inheritance instead of creating one big AnalogInput with lots of optional properties.

Proprietary Data
- A vendor invents a new feature and clients use a standard Obix service to learn about it.
- A client not knowing about the vendor's types ahead of time, asks the system to describe these types.
- A vendor creates a structure and publishes a standard definition that enables data-driven clients to understand the new data structure without needing to be updated by skilled programmers.
- A manufacturer's properties and data types are explicitly defined using the exact same type model as the standard ones.
- A vendor creates a standard definition of a proprietary structure by extending a standard data type.
- A manufacturer extends an enumeration by adding values without redefining the base definition.
- A system provides a separate document to enhance, or overlay a published definition, providing localization, translation and optional property usage.
- A server provides an separate document to augment a standard definition for UI layout, security and data storage choices.

Device Models
- A single Obix XML file wholly describes a BACnet or Lonwork device.
- A Lonwork device description includes all SNVTs and SCPTs. It describes all the UNVTs and UCTPs including how to extract structured information from the bits. The SNVTs /SCPTs and the UNVTs/UCTPs are expressed using the exact same type system.
- A tool auto-generates the device description of a programmable device.
- A device description defines a device's objects including what standard BACnet object they subclass and the optional properties actually used.

Naming
- A server identifies objects with an opaque string.
- A server passes results by-reference using an opaque naming string.
- A server identifies objects using: file paths, component identifiers, UUIDs, SQL/OQL queries, URLs and device addresses.

Real-Time Points
- A client polls real-time control point data.
- A client reads the scalar value, status/quality information and units of a point.
- A client efficiently polls thousands of points.

Historical Trends
- A client queries trends as tabular information using a time range.
- A client requests the daily average energy consumption from an electric meter.
- A BACnet server includes sequence numbers and event records in it's trend data.

Alarming
- A client queries current unacknowledged alarms, using a variety of predicates, and then acknowledges them.
- A client receives an alarm with arbitrary tagged data. The client looks for an alarm definition for the specific device and upon finding it, displays the data in a meaningful way.

On The Wire
- A server only sends values that differ from their default.
- A client requests meta-data when learning a system and subsequently receives only values.

Configuration
- A client reads and writes information from a configuration database using Obix.
- A client configures a previously unknown device that may be natively represented by a C struct, object oriented component, or is natively stored in an RDBMS.

Clients and Tools
- A client discovers a nonstandard structure in the vendor's device and is able to display the data, know what is editable, and what the units and restrictions are.
- An XSLT compiler condenses an Obix file into compact format for use in resource-constrained field devices.
- An compiler transforms an Obix document into a format understood by a manufacturer's existing legacy product line.
- A client uses it's knowledge of the standard base object of a proprietary object for presenting a hard-coded user interface.
- A tool exports data using the instance format for the values. The export feature optionally enables all the definitions that are known to apply to the exported instance data to be included with the instance data to make a "stand-alone" package.
- A client queries enumerations for displayable strings and corresponding values. This enables generation of user interface elements for editable values.