DXtera's Integration and Component Architecture

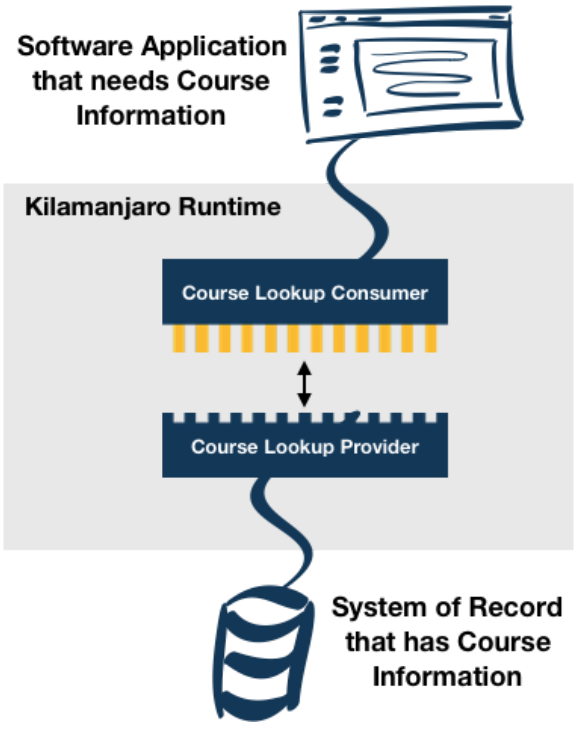
This technical overview serves to provide an initial introduction to select audiences. Please note the contents of this report are still in development and subject to change.

Every component of the DXtera's Integration Framework, from its runtime environment, business logic adapters, legacy system connectors, native service implementations and protocol bridges, can be updated, enhanced or even replaced as required. This is possible through ubiquitous use of the openly published service contracts at the core of its architectural design.

DXtera utilizes the leading educational business model and service contract specification available today, the Open Service Interface Definitions (OSIDs). These openly published and licensed models and service contracts help DXtera assure complete modularly and educational business alignment. The OSIDs are designed to reflect the business models of education, and they define all integration within the framework itself. DXtera builds all of its solutions utilizing the Java bindings of OSIDs and the Okapia Developer Toolkit. A key component of the toolkit is the Kilimanjaro Runtime environment, an open-source framework that manages plug-and-play integration between software components that implement the OSIDs.

Integration Example:

A simple example can be helpful in appreciating the power, scope and scale of the OSID specifications. The diagram at right illustrates integration within the Kilimanjaro Runtime environment to allow a user application that needs to access Course information with an enterprise system of record (SOR) that manages Course information. The OSID specifications define numerous operational functions that can be done with Course data. In this example, the consuming application needs to get basic Course information from the SOR, and all that is required to achieve this is a connector that implements the Course Lookup OSID specification. The defined API for Course Lookup includes basic functionality for getting one or more Courses by unique identifier, getting Courses of a particular type, or by course number.



While Course Lookup may be the only functionality required by this simple example application, there are many other functions defined in the OSID specification for operating on Course objects in an underlying system:

Course Lookup	Allows software to navigate and retrieve Course information
Course Query/Search	Allows software to search among Courses in catalogs
Course Administration	Allows software to create, update, and delete Courses, and to assign Id aliases
Course Notification	Allows software to register for and receive notifications when Course objects are added, changed or removed from a system
Course Catalog Navigation	Allows software to navigate Course-catalog mappings
Course Catalog Assignment	Allows software to assign Courses to catalogs
Course Smart Catalog	Allows software to manage queries and sequencing to create "smart" dynamic catalogs

The OSIDs also define software interfaces for the Course object itself. The attribute model for a Course, accessible through these interfaces for the purposes of examining a Course object, creating a Course Query for searching, defining a Course Form for authoring new Courses, etc, looks something like this:

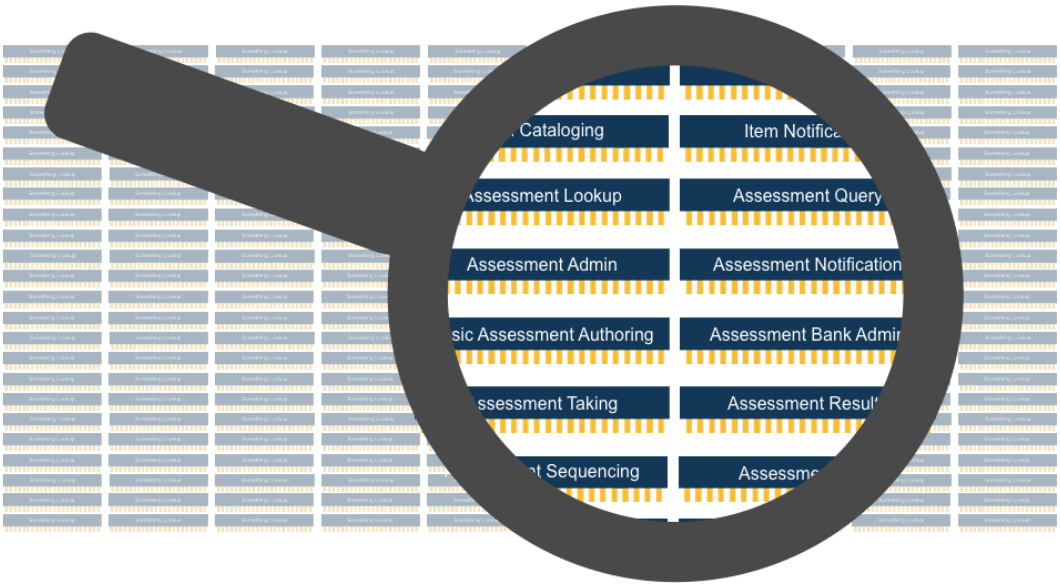
id	A unique and permanent identifier
displayName	The name of the Course
description	A description of the Course
genusType	The Type of the Course: standard, equivalency, thesis, seminar, walking tour, etc
isActive	Boolean indication of whether this Course is active.
title	The formal title of this Course. It may be the same as the display name or it may be used to more formally label the Course. A display name might be Physics 102 where the title is Introduction to Electromagnetism
number	The course number which is a label generally used to index the course in a catalog, such as T101 or 16.004
sponsors	The sponsors of the Course as a Resource, if available. These could be organizations, like departments

creditAmounts	The credits in which this course can be offered, The credits may be expressed as Grades
preRequisitesInfo	An informational string describing the Course prerequisites
preRequisites	The Requisites for the course prerequisites, if available. Each Requisite is an AND term such that all Requisites must be true for the prerequisites to be satisfied
levels	The list of Grade levels of the Course. Multiple levels may exist for different systems
gradingOptions	The various grading options available to register in this Course as a list of GradeSystems, if applicable
learningObjectives	The overall learning Objectives for this Course, if available
AdditionalRecords	Additional records defined through out-of-band interface agreements required to extend the Course model as needed

The table above references a number of other entities that have their own functional and object interfaces defined in the OSID specification. These include things like Types, Resources, Grades, GradeSystems, Objectives, etc.

A Vast Specification:

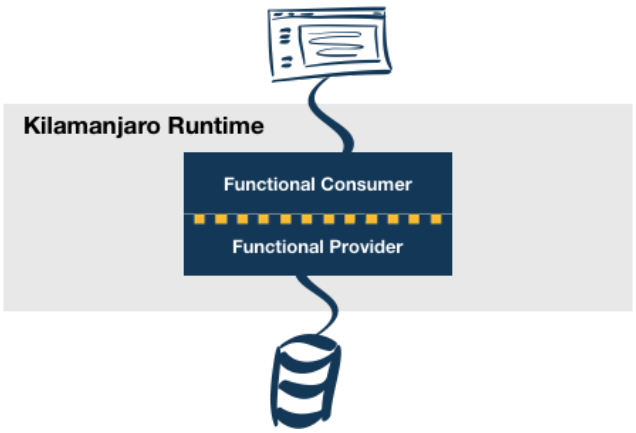
The example above only scratches the surface of the OSID specifications utilized by DXtera. The full specification is available at http://osid.org/specifications. It defines hundreds of business entities required for developing and integrating educational software, and over 10,000 APIs like the one outlined above.



The scope and scale of specification can be quite overwhelming, and for that reason implementers are encouraged to start small, focusing on one or two functional operations across one or two entities. Doing so will surface the patterns and differences that exist across the definitions. A real-world integration seldom requires just one or two entities and operations, and a number of functions across multiple entities will typically be needed. The DXtera architectural team is expert at mapping the OSID specification to existing systems and across integration challenges and can help DXtera members navigate the specification to achieve a particular integration or software design goal.

The Runtime Environment:

The Kilimanjaro Runtime Environment manages all aspects of DXtera's framework service stack, including configuration, adapter and connector management. The Runtime defines how consuming applications, business logic adapters, SOR and legacy connectors, and native implementations plug together. It provides the means for passing configuration information to underlying adapters, allowing business owners to define business logic, or to configure how local data models are mapped through the interfaces

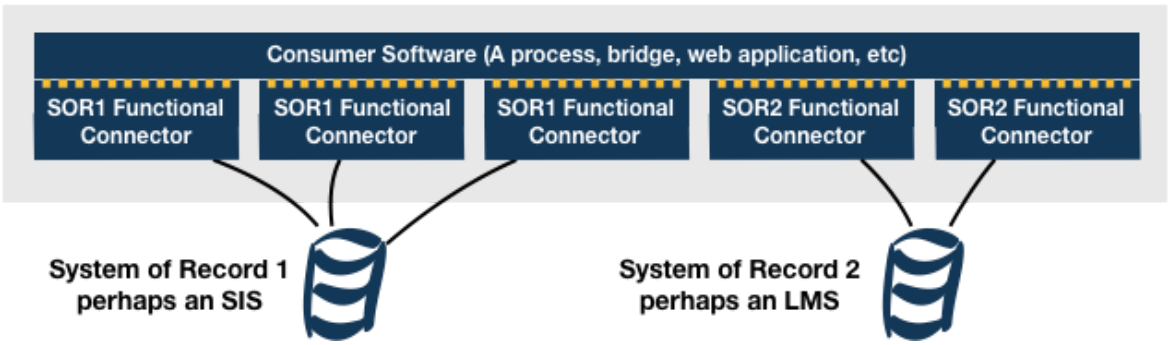


Kilimanjaro is part of the developer toolkit supporting the Java bindings of the OSIDs, and like everything in the specification, it is also implemented to a published OSID interface definition for an OsidRuntimeManager.

Kilimanjaro can be run in a campus' data center, alongside ERP systems, or hosted in the cloud. The DXtera Institute can provide hosting service or can partner with enterprise cloud providers.

SOR and Legacy Connectors:

Connectors implementing the service contract interfaces can be built to encapsulate the details of system integration with existing enterprise systems, and provide secure "plug-and-play" capability within the Runtime environment.



As suggested by the prior example, an enterprise system like a Student Information System is made up of many different entities and functional operations. Depending on the requirements of the consuming software, a number of these operations will need to be implemented and instantiated in the Runtime. For instance, to support the extract, transform and load process for DXtera's own Operational Data Store (ODS), Lookup connectors have to be built to over 20 different entities.

With the help of our member institutions, the DXtera Institute is developing connectors, adapters and consuming applications through various demonstration projects and other collaboration efforts. Once developed, tested and released, the Institute will maintain adapter code and/or binaries as part of its growing library of resources to be used by our members who require functional access to their legacy systems.

Native Service Implementations:

The DXtera Institute will also deliver purpose-built implementations of a number of service contracts as needed to support the operation of the framework itself. For instance, DXtera defines functionality for Authorization, ID management, Configuration, and activity Logging, to name a few.

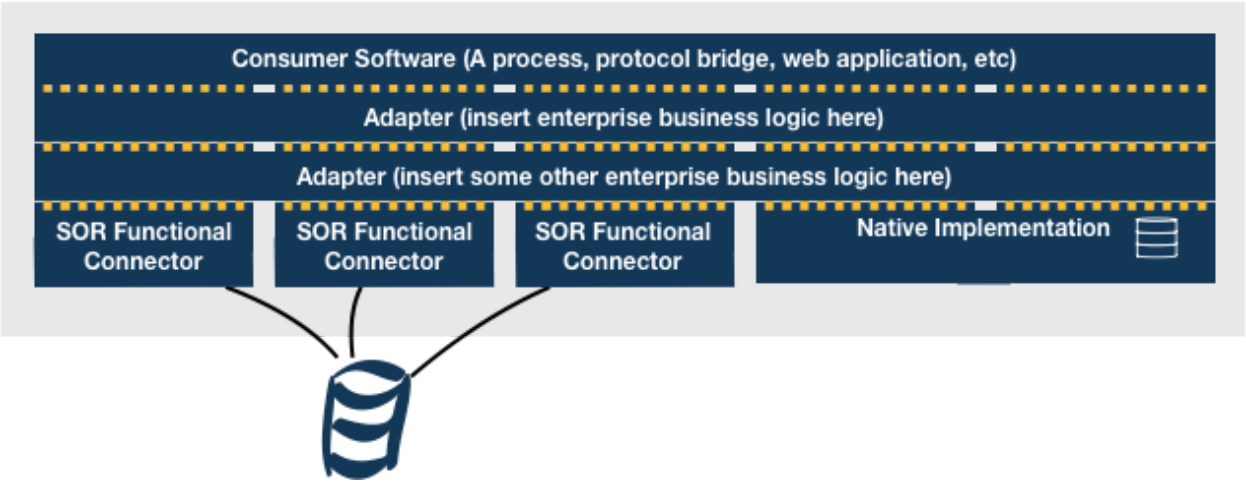
These kinds of services are foundational to the operation of the Runtime Environment and required for deployment of certain business logic adapters that the framework is designed to support, and therefore DXtera will deliver its own implementations to assure that institutions can get up and running.

Organizations wishing to utilize their own enterprise service technologies for these kinds of foundational functions can swap the DXtera provided implementations with service connectors that implement preferred technologies or leverage existing infrastructure.

In addition, DXtera is developing impomentations across many of the other OSID specifications for the purposes of supporting other products of the Institute, including the services being developed to implement DXera's own Operational Data Store.

Business Logic Adapters:

The behavior of the DXtera framework can be modified and enhanced through Business Logic Adapters developed for the purpose of defining and enforcing business rules or altering the functionality of underlying integrated systems. Business logic adapters both consume and implement the published service contracts, and can be layered into the DXtera Framework's service stack though Runtime configuration.



DXtera will deliver at least two business logic adapters in its first release of the Integration Framework:

> ***Federation*** - One of the key features of the DXtera integration framework is the ability to federate functionality from multiple underlying legacy systems. For instance, it may be

advantageous to consolidate information across multiple student information systems about all the courses available.

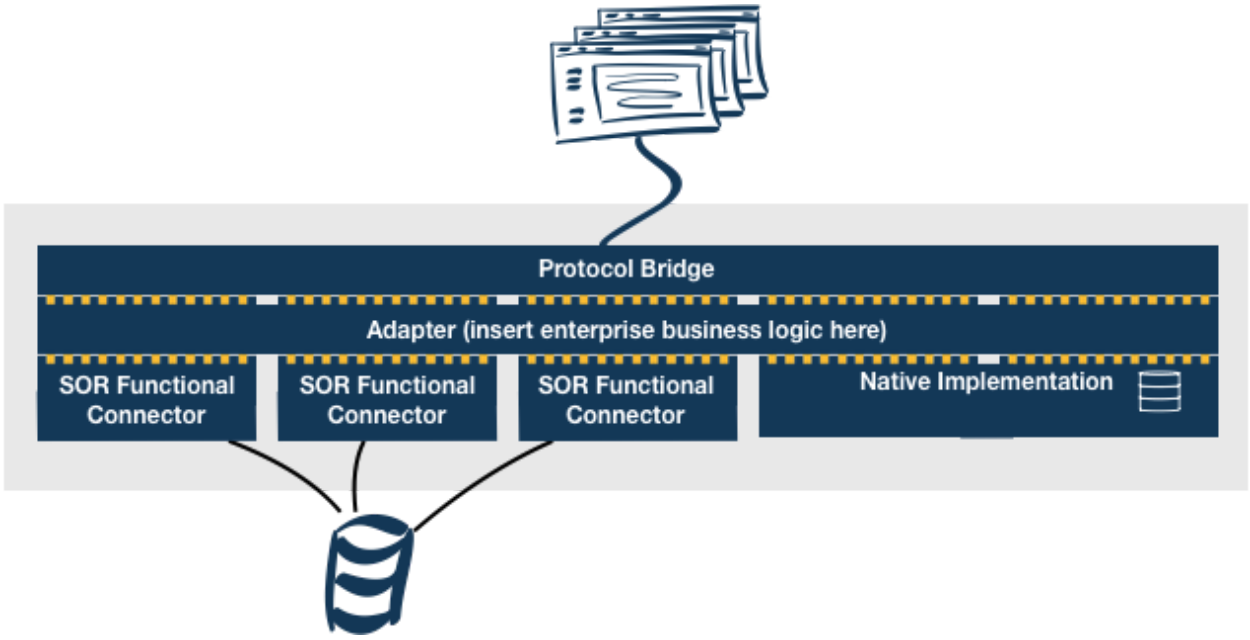
Aggregation - Sometimes information needs to be orchestrated among multiple service providers. For instance, while an insitution's legacy SIS may be the system of record for information about degree Programs, the descriptions of those Programs may be held in a course catalog system. Information from both systems needs to be brought together to fully describe a Course.

Orchestration - Sometimes multiple service implementations need to be bundled together for the purposes of coordinating information from a single provider. For example, a repository service may also provide a calendar view of assets. With Orchestration, a single service can be instantiated where the loading of the Repository and Calendaring services are handled by an orchestration provider.

Access Control - Access control adapters utilize an underlying Authorization service to intercept any requests to access or administer information and allow or deny such requests based on the privileges of the user. Many educational enterprises manage authorization rules in their own systems, which can be used instead of or in concert with DXtera's native Authorization service implementation

Protocol Bridges:

The functionality of underlying systems can be made available to applications and application developers via Protocol Bridges. There are a number of protocol technologies popular with application developers including SOAP based Web Services, RESTful APIs, Protocol Buffers, or bespoke application protocols to name a few. Protocol technologies are supported through Bridges designed to directly consume the underlying service of the framework via DXtera's defined service contracts.



Dxtera is delivering its own protocol bridges to expose its service contracts via REST protocols aligned with the OSID service contracts as closely as possible. Through the DXtera Institute's member feedback and funded projects we will develop new bridges as required.

One reason we are doing all of this is to support next generation student success applications or systems. The DXtera Institute is equipped to support our members or industry affiliates who are developing such applications to take advantage of the Framework, its service interfaces and supported protocols. The Institute is eager to help support its community in using and extending the Integration Framework to achieve the full potential of this groundbreaking system.

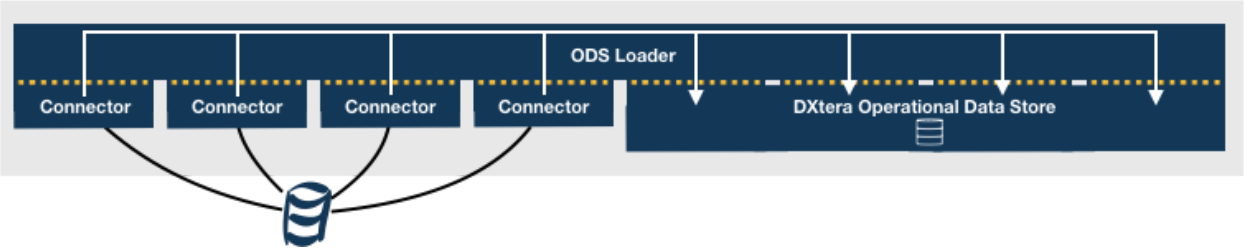
DXtera will deliver administration class applications designed to help our members configure, manage and analyse the framework itself. Such applications are required for configuring the service implementation stack, support job processing , manage authorizations, manage enterprise metadata, monitor system performance and other critical functions.

Not all applications need to be developed to a protocol. Applications can also be designed to run as HTML servlets directly consuming the OSID Java interfaces.

The DXtera Operational Data Store:

DXtera's Operational Data Store (ODS) is an enterprise operational data store that can track information across multiple business areas. DXtera's ODS is unique in that it is designed to consume digital information from multiple enterprise systems typically found in an educational enterprise (SIS, HR, LMS, etc). The ODS Loader application is an OSID consumer that utilizes

Lookup connectors across all entities required for reporting, including Program, Term, Course, CourseOffering and the entities that make up an academic record, to name a few.



DXtera's ODS itself is a native implementation of all the corresponding Admin functionality defined in the OSIDs. In other words, DXtera "eats its own dogfood" as the saying goes when developing its own software. Utilizing only the OSID connectors for the source system and the native OSID implementations for the ODS, the ODS Loader reads information from source systems and copies to the ODS, with incremental loads scheduled daily. The data is transformed in the SOR Connectors to the OSID domain model, assuring alignment with institutional business models across disparate enterprise systems.

Testing:

The Institute is continually developing and updating test harnesses available to institutional developers or vendors to meet particular institutional needs or to bring new framework components to the marketplace. This includes tests aimed at ensuring that legacy connectors, business adapters, native implementations, and protocol bridges can be safely and securely plugged together to modify the framework to meet new and evolving requirements and technologies.