

# Enterprise Integration with a Canonical Model



# Enterprise Integration with a Canonical Model

---

## Introduction

Most organizations are moving away from point to point integration in a bid to be more agile and to reduce integration costs (via greater reuse). Whether at the stage of using simple messages to talk to web services or a fully governed SOA approach, the key to achieving these goals of agility and reuse is to solve the challenge of consistent data. If a Canonical Model can be established and adopted, (ideally at an enterprise level or at the very least at a domain level) the move away from point to point integration can be successfully achieved. As one Chief Architect recently put it “ if you don’t standardize data you cannot re-use services”

This paper presents an overview of igniteXML’s support for enterprise integration through the management of canonical models as an enterprise integration model. A high level view of the igniteXML architecture is presented as are examples of how igniteXML may be integrated into an existing or proposed enterprise integration environment.

Companion papers provide an introduction to igniteXML, detail how specific enterprise integration modeling patterns are implemented using igniteXML and provide technical details of igniteXML.

---

## The igniteXML Solution

### What does it do?

- Manage models and referenced extensions for object reuse
- Organize model objects into logical components
- Provides an enterprise semantic model
- Import XML (including Schema and WSDL)
- Generate valid, optimized, model compliant XML schema for use as message payloads (body)
- Generate vocabulary relationship documents which can be used as the basis for mapping documents
- Generate statistical reports of managed objects

Following are examples of how igniteXML may be integrated with other enterprise integration products.

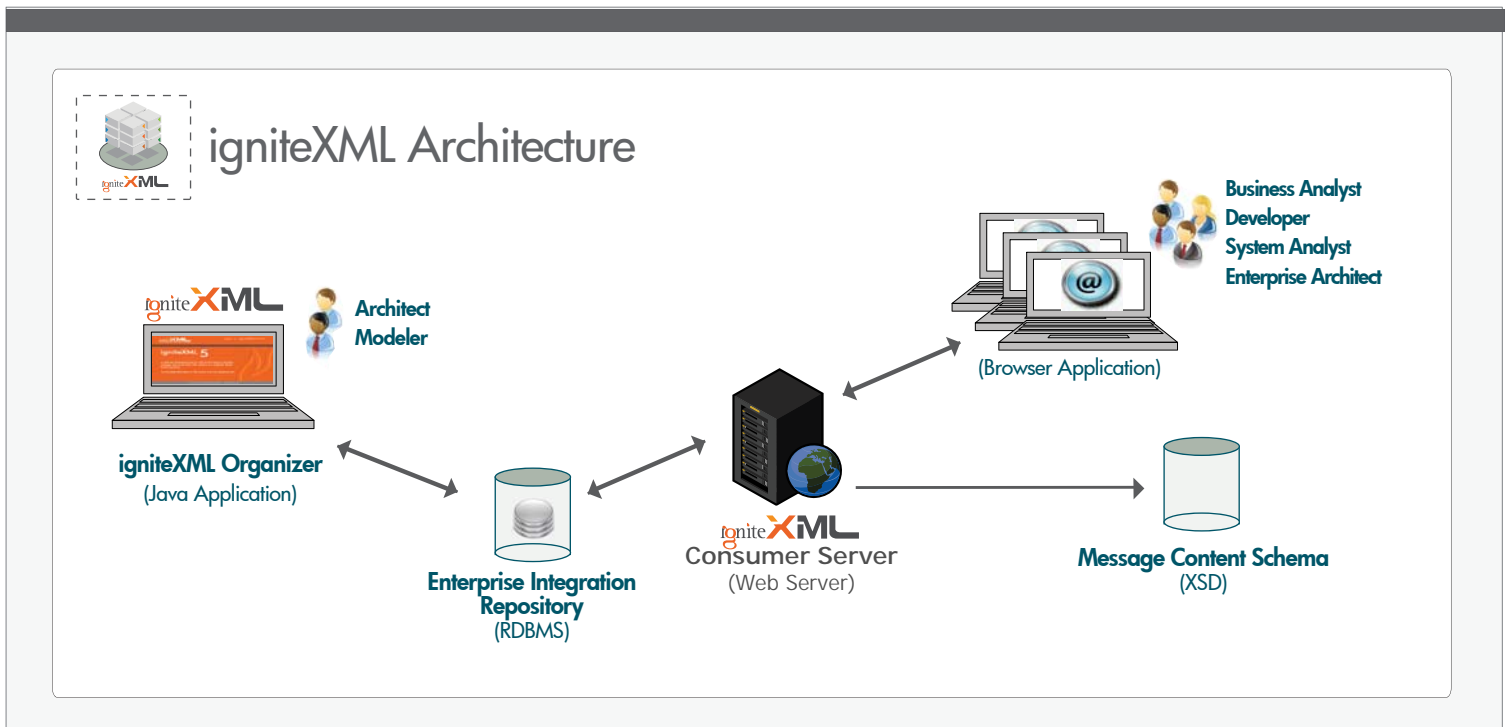
# Enterprise Integration with igniteXML

## igniteXML Architecture

The igniteXML architecture is comprised of two distinct components – the Organizer Module and the Consumer Module. Models are managed in the Organizer Module and XML schema are generated from the Consumer Module.

Model managers use the Organizer Module to structure model objects into XML schema building blocks that are exposed to model consumers via collections termed Core Components. For example, a Core Component named “Customer” might have building blocks such as “Name”, “Address”, “ContactInformation”, etc. The “Address” building block might reference another building block in a different Core Component named “Address” which has building blocks such as “Street”, “City”, “State”, “ZipCode”, etc. The depth and complexity of building blocks is completely under the control of the model manager as is the definition of the Core Component containers the building blocks exist in. The model manager decides which Core Components (and thus which building blocks) to expose to XML message builders using the Consumer Module

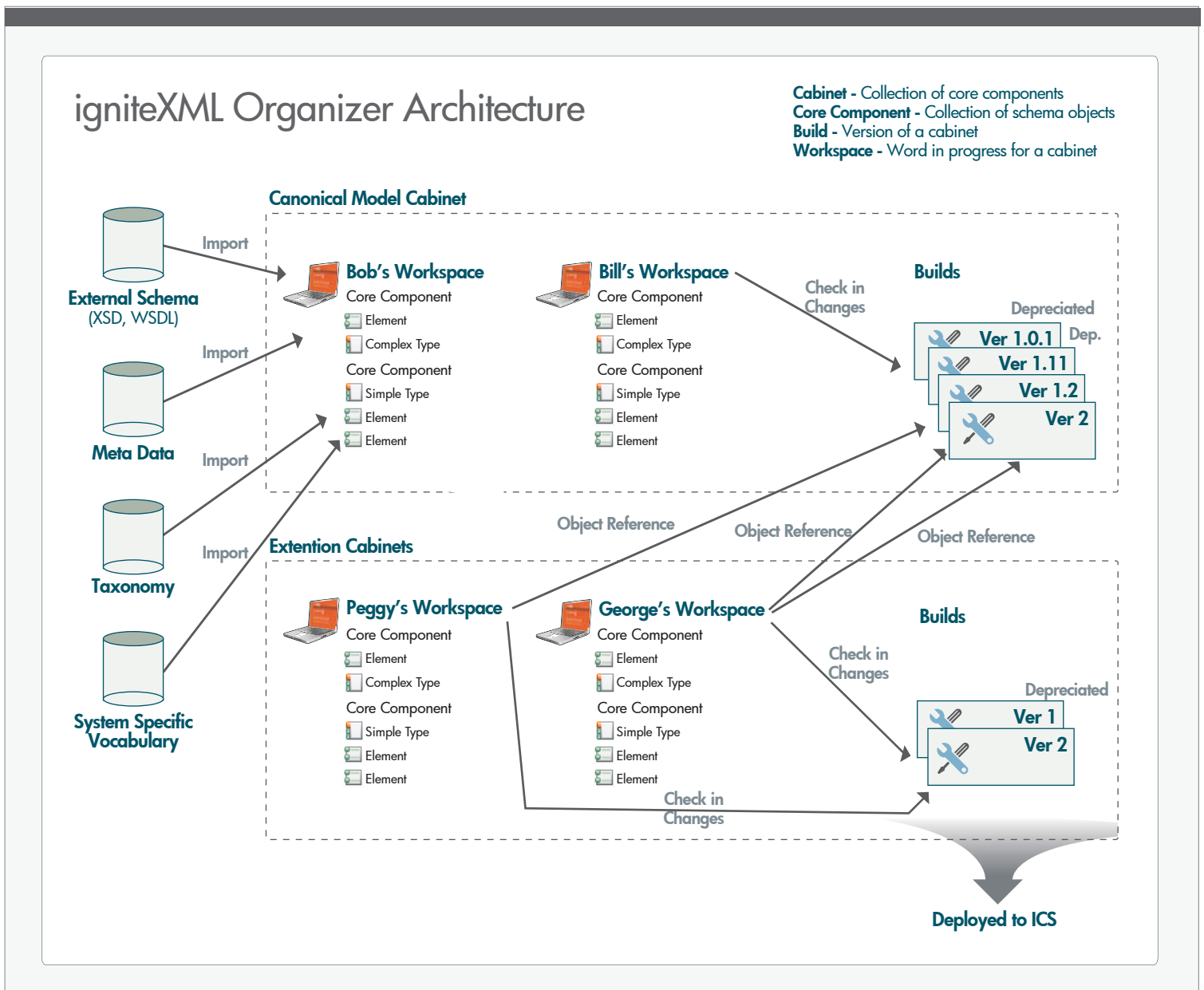
Using the Consumer Module, message builders navigate a tree structure or use a comprehensive search capability to find the desired building blocks exposed by the model managers and compose a subset XML schema. The message builder works at a logical level and does not need to know the details of XML to generate a subset schema. In the Consumer Module the message builder selects only the building blocks they are interested in for their message. When the message builder requests a schema to be generated, igniteXML will use the structure established by the model managers in the Organizer Module to construct a valid, optimized XML schema based on the building block selections made by the message builder.



# Enterprise Integration with igniteXML

## Organizer Module

The Organizer Module is used by model managers to organize the components of a model and deploy them to the Consumer Module. It uses a Client/Server architecture with a Java application installed on a user's desktop accessing a RDBMS (SQL Server or Oracle) installed on a server within the organizations LAN. Access is integrated with a users Windows authentication.

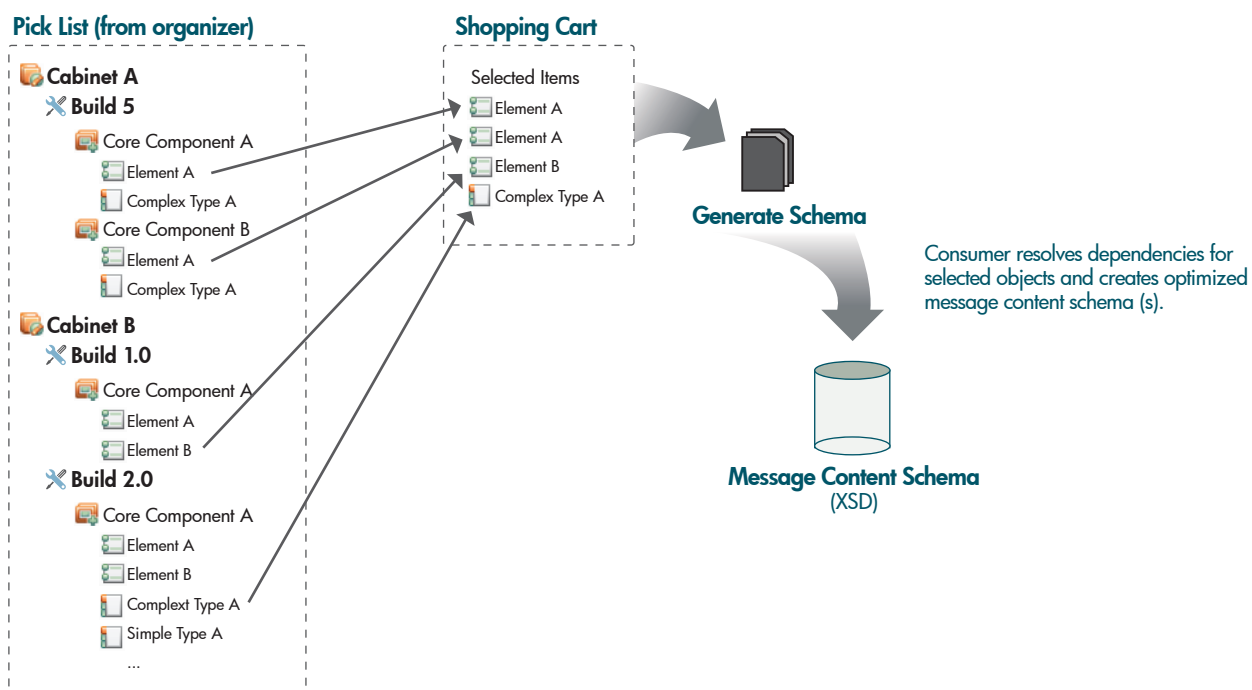


# Enterprise Integration with igniteXML

## Consumer Module

The Consumer Module is used by Business Analysts, System Analysts and Application Developers to construct message schema from the components exposed by the model managers. It uses a web service architecture with the end user accessing the application via the Internet Explorer web browser. Supported J2EE Web Application Servers are Oracle WebLogic, IBM WebSphere and Apache Tomcat. Access is controlled by the application server.

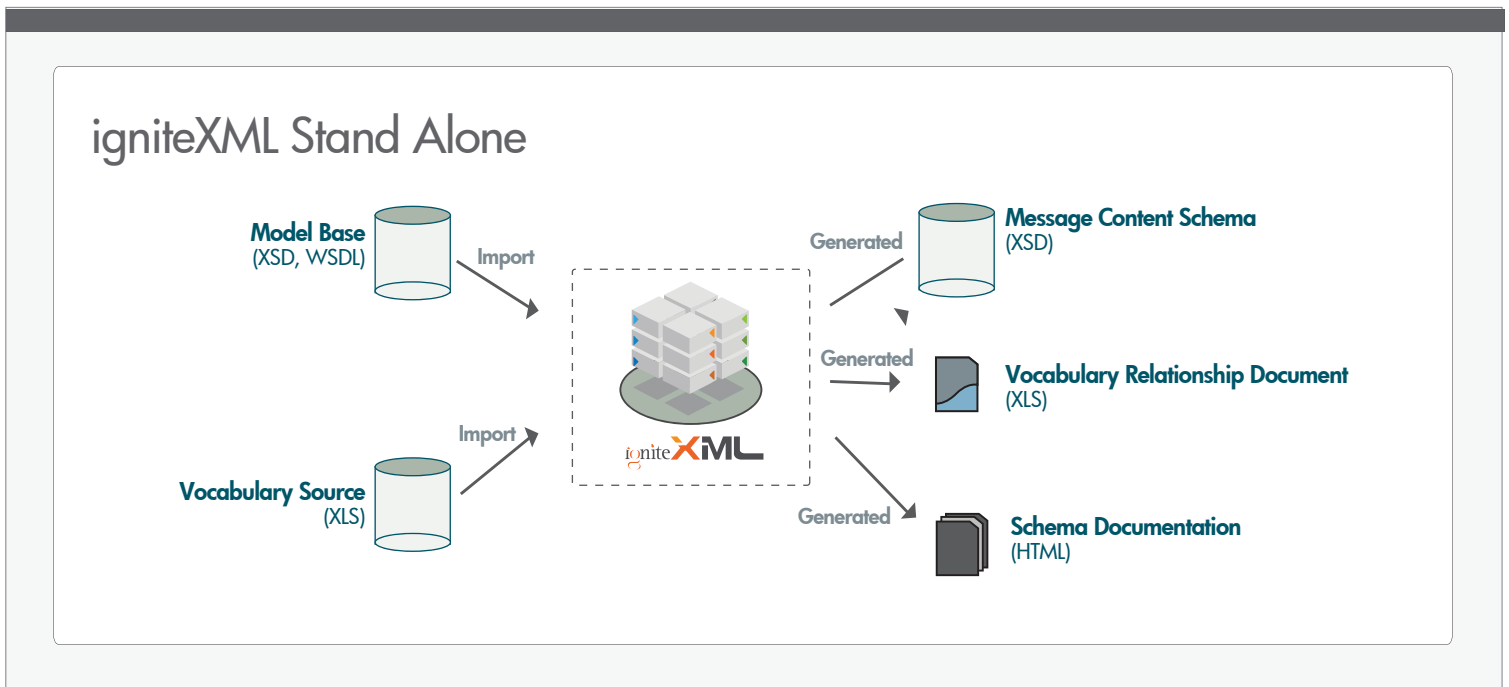
### igniteXML Consumer Server Architecture



# The igniteXML Solution

## How does igniteXML add value to existing tools?

igniteXML may be used as a standalone solution for the creation of Canonical Models and management of governed messages. Organizations implementing igniteXML as part of an enterprise integration architecture realize an even greater benefit when igniteXML is paired with other best of breed enterprise integration products for a complete integration solution.



Following are examples of how igniteXML may be integrated with other enterprise integration products.

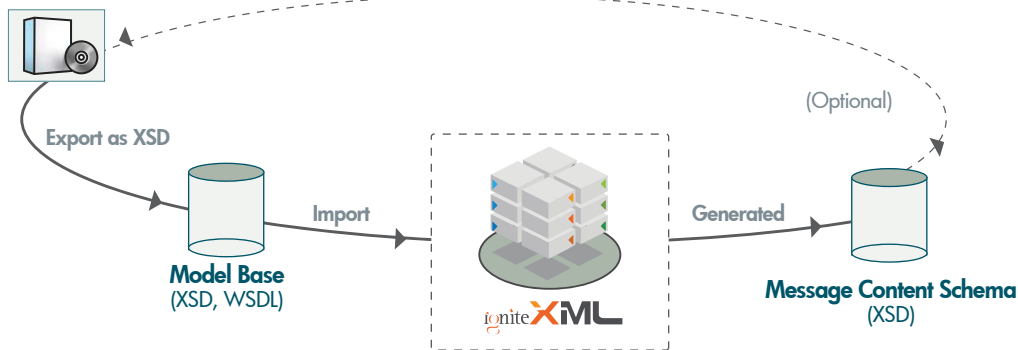
# The igniteXML Solution

## Existing Data Modeling tools

Some organizations use graphic modeling software such as CA Erwin or IBM Rational to manage data models and occasionally canonical models. There is a gap between these models and what is needed in the XML messages and Schemas used in the run time. igniteXML uniquely bridges this gap. Organizations may continue to use this software to graphically depict their models and can output instances of the model to igniteXML.

## igniteXML with Modelling Software

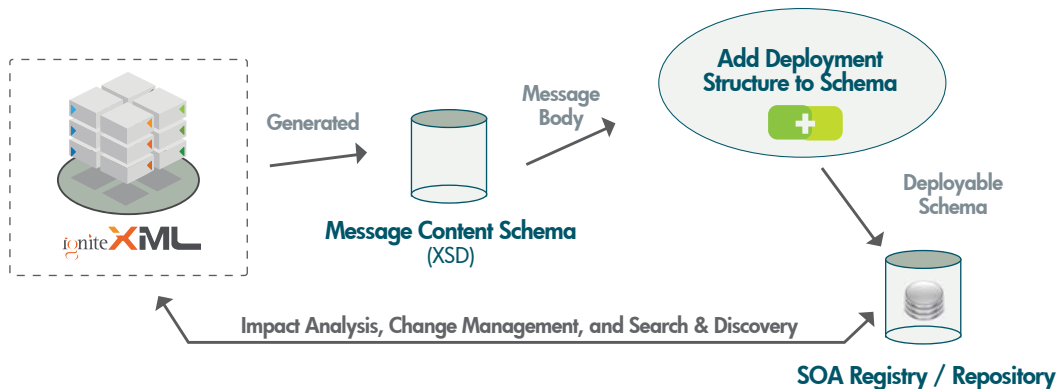
### Modelling Software



## Existing SOA Registry/Repository tools

Many organizations maintain a SOA Registry/Repository such as Software AG's CentraSite, HP's Systinet or IBM's WSRR to manage their SOA Governance. A SOA Registry/Repository can be integrated with igniteXML via a Java API. This can be used to migrate Registry/Repository users away from the non governed approach of re-using XML schemas and messages (which usually involves manually extending them outside the governed environment) to re-using the Canonical Model to easily and quickly build XML messages and Schemas which are current, complete, consistent and runtime optimized.

## igniteXML with SOA Registry/Repository



# The igniteXML Solution

## Existing SOA Policy Engine tools

A policy engine, often found as part of a SOA Registry/Repository suite can be integrated with igniteXML. This allow policy control to be extended to both the creation of Messages and Schema as well as to the organization and collaborative management of the Canonical Model.

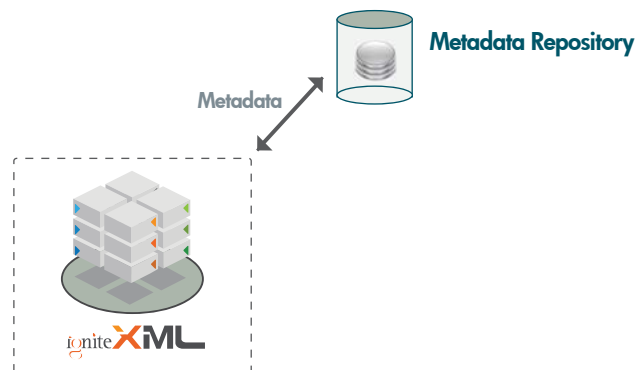
### igniteXML with Policy Engine



## Existing Metadata tools

Even though igniteXML has substantial meta data support some organizations have a significant investment in a preexisting metadata repository like ASG – Rochade which they wish to maintain. These repositories can be integrated with igniteXML so metadata may be attached to managed objects and output in igniteXML generated schemas.

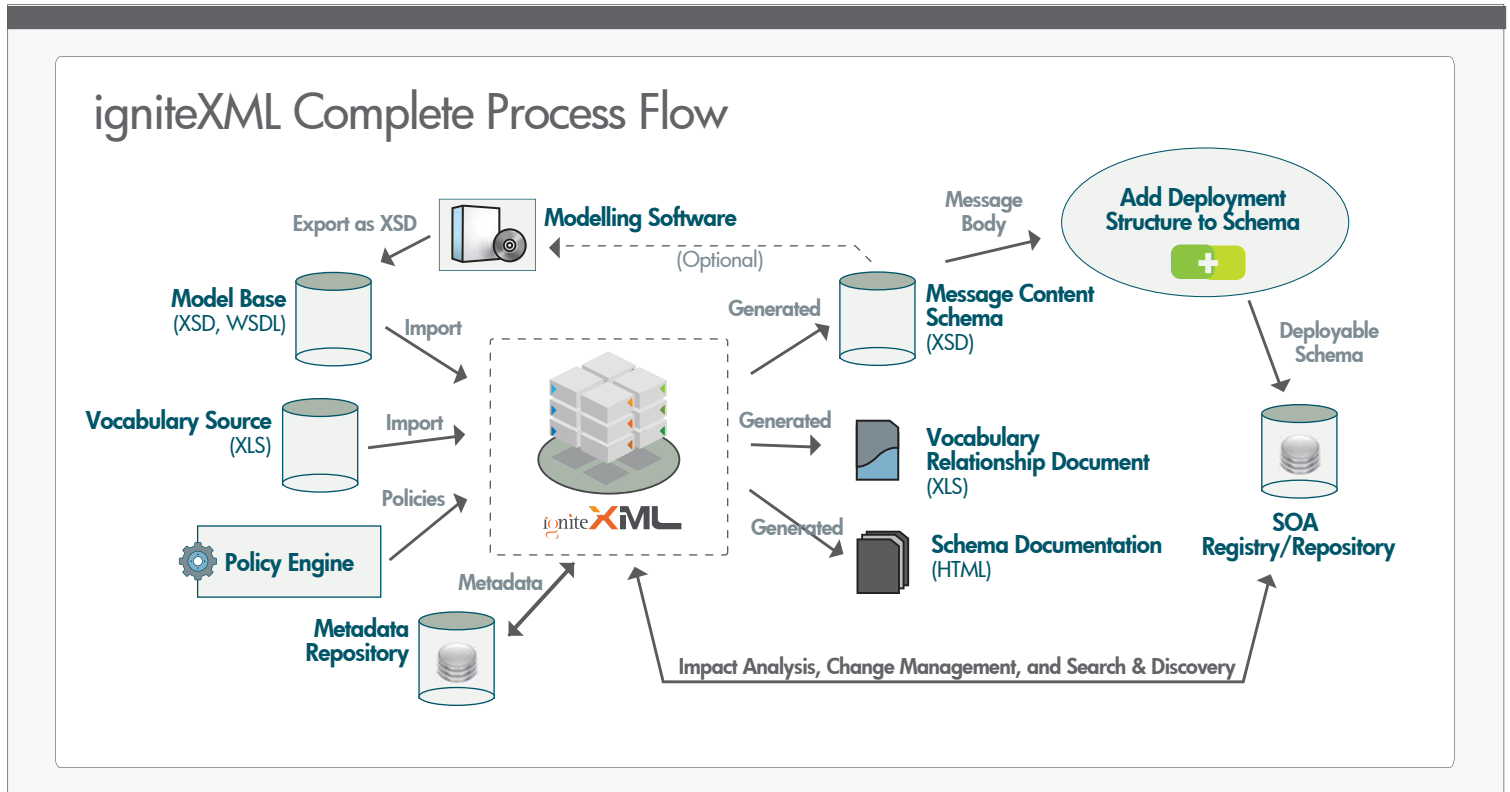
### igniteXML with Metadata Repository



# The igniteXML Solution

## igniteXML Complete Process Flow

A complete, end to end enterprise integration flow incorporating igniteXML as the primary solution is depicted in following diagram.



### US Corporate Headquarters:

digitalML-USA Inc.  
580 California Street,  
Suite 1600,  
San Francisco, CA 94104  
Tel: 1.415.373.0300 Fax: 1.415.373.0307  
Email: sales@digitalml.com

### UK Corporate Headquarters:

digitalML Ltd.  
Atrium Court,  
The Ring,  
Bracknell,  
Berkshire, RG12 1BW,  
United Kingdom  
Tel: 44 1344 390542 Fax: 44 20 7990 7609  
Email: support@digitalml.com