

Introduction to igniteXML

igniteXML Overview

igniteXML is focused on solving two of the major challenges found in enterprise integration. The first challenge is management of enterprise integration models and industry standard frameworks. The second is generation of model/framework compliant XML schema. While primarily designed for use with Service Oriented Architecture (SOA), igniteXML supports other message based integration patterns such as Enterprise Application Integration (EAI).

igniteXML is a collaborative, multi user enterprise grade repository that manages enterprise integration models and generates model compliant XML schema for use in middleware messaging or other enterprise integration patterns that require model compliant subset schema.

Key features include:

- Canonical Model management and governance
- Industry Standard XML Framework management and governance
- Semantic Modeling
- Import of data models, XSD, and WSDL to create or enhance models
- Meta Data and Taxonomy overlays for managed objects
- Collaborative, multi user management with role based security
- Lifecycle management, version control, approval process
- Extensive search capabilities to discover usage of managed objects throughout the enterprise integration environment
- Impact Analysis of changes to managed objects
- Generation of model compliant XML schema through an easy to use browser based interface
- Discovery of model objects based on end user vocabularies (Business Analysts, Application Developers don't have to know the model to build compliant XML schemas, they use their own terminology)
- Integration with other SOA, EAI, modeling, etc. tools for end to end governance

The igniteXML Solution

Adoption of a comprehensive enterprise integration strategy often entails a move to a message based integration methodology. A typical migration from point to point interfacing to message based enterprise integration begins with a decision on the middleware to be used to transport messages between endpoints. Often this decision

results in the adoption of an ESB or EAI solution. Most organizations also adopt a set of standards for message structure and content (payload). The desire for consistent message payload results in the construction of an enterprise canonical model or acquisition of an industry standard framework used as the basis for message objects. This is the birth of an organization's enterprise integration model.

The goal of enterprise integration modeling is to provide a dictionary of common objects and definitions at an enterprise (or domain) level to enhance system interoperability. Enterprise integration models provide a foundation for a decoupled, consistent, reusable integration methodology which can be implemented using messaging supported by middleware products. Message payloads (business data content) in the form of XML schema are built from the common model objects thus providing the desired consistency and reusability.

Getting Stated With Enterprise Integration

Many organizations begin the management of their enterprise integration model using a desktop editing tool such as Altova's XML Spy. Initially this approach suffices even though some obvious management and governance issues are apparent from the outset. Difficulties immediately arise with collaboration and soon thereafter with change management in this file based management approach. Communication among model managers must be perfectly orchestrated or collisions and overlays occur on a regular basis. This is especially true in the earlier years as the model changes are frequent and more comprehensive.

As adoption and use of the model increases more and more messages are created causing change management to become a significant issue. Model managers need to research the effect of proposed model changes on previously deployed messages and perform what-if analysis for model restructures. In a file based management approach the relationships between the model and messages must be manually maintained (Excel spreadsheets are common) or are ignored completely due to the inherent complexity of managing such a large number of relationships.

Upgrading To an Enterprise Solution

igniteXML eliminates the key failings of a file based management approach by providing a multi user, role based collaborative model management environment with a comprehensive model-object-to-schema-object relationship paradigm. igniteXML solves these challenges and more to provide a comprehensive enterprise integration model management solution.

Challenge 1 – Collaborative Model Management

Model managers are assigned roles within groups which allows for a separation of concerns even within the enterprise integration group. Organizations are free to determine how they wish to group users (users can be part of more than one group) and assign model components to groups. Components can be restricted for use by certain groups or made publically available to all users. igniteXML also provides comprehensive collaboration and governance though the use of workspaces (private work in progress), check in, approval, and build (versioning) functionality.

Challenge 2 – Change Management

igniteXML maintains a n-level, bi-directional repository of references from model objects to model extensions to message objects. This provides a traceable, end to end relationship web which enables igniteXML's change management functionality. Object usage can be found within a workspace, within a certain group or across the entire enterprise in both directions - ancestors as well as descendants can be located. Through the use of API functionality, a SOA Registry/Repository can be integrated with igniteXML allowing model object discovery within deployable services. igniteXML also provides impact analysis, which displays effected objects when a change is made in a user's workspace. Since the change is made in the users workspace (uncommitted work in progress), if impact analysis indicates a negative impact the change can be removed and an alternative attempted.

Challenge 3 – Governance

As the demand for new message schemas increases, the capacity of the enterprise integration group to deliver message schemas in a timely manner decreases. One solution is to increase the enterprise integration staff. Is this a wise use of resources? Model managers are highly skilled, highly compensated resources whose function should be management of the enterprise integration model this is not a wise use of resources. All too often they are also tasked with building the message schemas due to governance concerns. igniteXML provides for an effective, efficient message schema construction method by shifting the responsibility of message schema construction to other resources such as Business Analysts, Systems Analysts and Application Developers. Governance is maintained since igniteXML allows message schema composition using only prebuilt components from the model.

Challenge 4 - Adoption

Another significant challenge facing organizations deploying message based enterprise integration architecture is adoption. Application development teams or service providers are reluctant to use the architecture due to the learning curve for understanding the model. In order to increase use of the architecture, enterprise integration groups often end up building the message schemas rather than forcing the application development teams to learn the model. igniteXML eliminates this challenge by allowing application

development teams to navigate the model components exposed by the model managers using the vocabulary familiar to them.

Challenge 5 – Message Content

A final challenge is controlling the content of the message schemas. As mentioned in Challenge 3, compliance to the model is automatically enforced by igniteXML since the user can only select modeler approved components to build a message. What about dependency resolution and schema optimization? Creating a valid XML schema from scratch can be a daunting task, especially for someone who does not work with XML every day. Creating an optimized schema is even more difficult. igniteXML automatically handles both these tasks. Using igniteXML a message developer is only concerned with finding the business objects needed for the message. igniteXML will resolve all the dependencies for the objects and build a valid schema (or set of schemas). The schema will be optimized to include only the objects necessary to make the schema valid. Without igniteXML, message schemas built from a model often suffer from significant bloat since manual optimization is difficult and prone to errors. Because igniteXML manages the complex relationship web inherent in most enterprise integration environments, creation of valid, optimized message schema can be accomplished with little effort or model knowledge.

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