Data Abstraction Best Practices Technical Guide

Project Name	AS Assets Data Abstraction Best Practices
Document Location	This document is only valid on the day it was printed. The source of the document will be found in the ASAssets_DataAbstractionBestPractices folder (https://github.com/TIBCOSoftware)
Purpose	Self-paced instructional

This document presents a detailed explanation of the open source Data Virtualization Data Abstraction Best Practices capability. It provides detailed overview of the architecture and usage.

www.tibco.com

Global Headquarters
3303 Hillview Avenue
Palo Alto, CA 94304

Tel: +1 650-846-1000
+1 800-420-8450
Fax: +1 650-846-1005

Revision History

Version	Date	Author	Comments
V1	04/28/2010	Mike Tinius	Initial revision
V3	08/23/2010	Mike Tinius	Added SOA Read/Write white paper
V5	11/05/2011	Mike Tinius	Added Business Views in Business Layer
V7	04/08/2013	Mike Tinius	Modified major layers to 3. Enhanced technical guidance.
V7.1	08/20/2013	Mike Tinius	Updated documentation to Tibco format.
V8.0	11/07/2013	Mike Tinius	Updated for release 8.0
V8.1.2	04/09/2014	Mike Tinius	Removed version number from title page
V8.1.6	05/20/2015	Mike Tinius	Updated White papers and Powerpoints to Cisco format.
V8.1.9	12/06/2017	Mike Tinius	Transitioned to Tibco for release 8.1.9

Related Documents

Name	Version
Tibco Data Abstraction Best Practices White Paper.pdf	8.1.9
Tibco Data Abstraction Best Practices.pptx	8.1.9
Tibco Data Abstraction Best Practices Technical.ppt	8.1.9
Tibco SOA-Centric Read/Write Methodology Technical Note.pdf	8.1.9

Supported Versions

Name	Version
TIBCO® Data Virtualization	7.0 or later
AS Assets Utilities open source	2017Q4 or later

Table of Contents

1 Complexity and Agility Require Data Abstraction

Business and IT Challenges

All large enterprises and government agencies today are looking to improve on their bottom-line, cut costs or reduce their risk by providing better access to information assets. Towards these objectives, all must overcome the challenges presented by significant volumes of complex, diverse data spread across various technology and application silos. Further complicating matters are a range of problems such as each source having its own access mechanisms, syntax, and security. Other problems involve sources that are not structured properly for consumption or reuse, have incomplete data or duplicate data, and have a mix of latency issues.

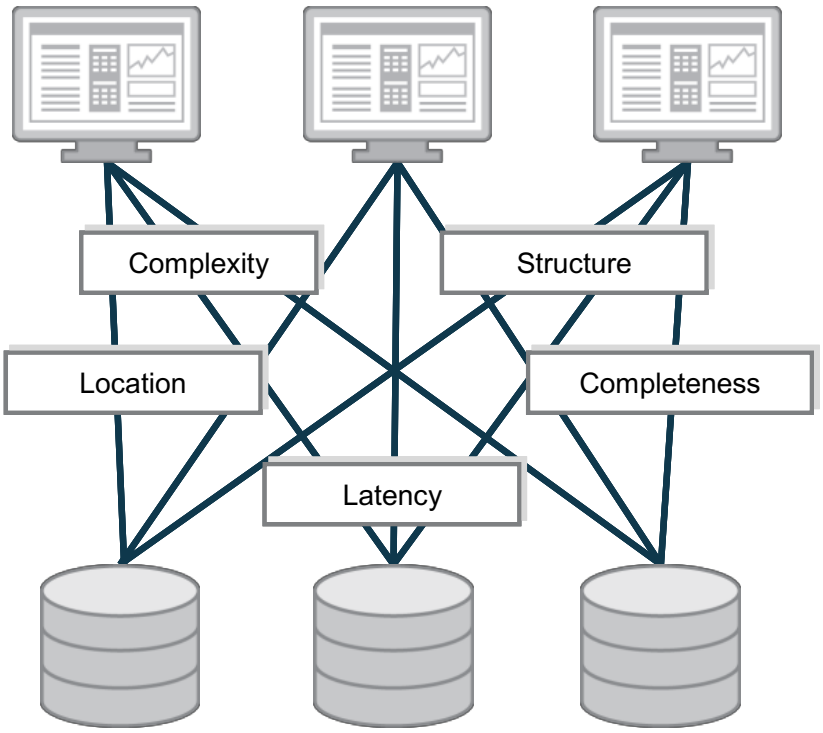

Figure One: Data Abstraction Challenges

Data abstraction overcomes data structure incompatibility by transforming data from its native structure and syntax into reusable views and data services that are easy for application developers to understand and solutions to consume. But some data abstraction approaches work better than others. For example, organizations that build a data abstraction layer by hand in Java or use business process management (BPM) tools are often constrained by brittleness and inefficiencies. Further, such approaches are not effective for large data sets since they lack the robust federation and query optimization functions required to meet data consumers' rigorous performance demands.

Solution – Use Data Virtualization to Architect a Data Abstraction Layer

Tibco Data Virtualization is an optimal way to implement an abstraction layer at enterprise scale. From an enterprise architecture point of view, Tibco's Data Virtualization solution forms a semantic abstraction or data services layer in support of multiple consuming applications. Sometimes called Information-As-A-Service by Forrester Research or SOA Data Services by Gartner, this middle layer of reusable services as shown in Figure Two decouples the underlying source data and consuming solution layers. This provides the flexibility required to deal with each layer in the most effective manner, as well as the agility to work quickly across layers as applications, schemas or underlying data sources change.

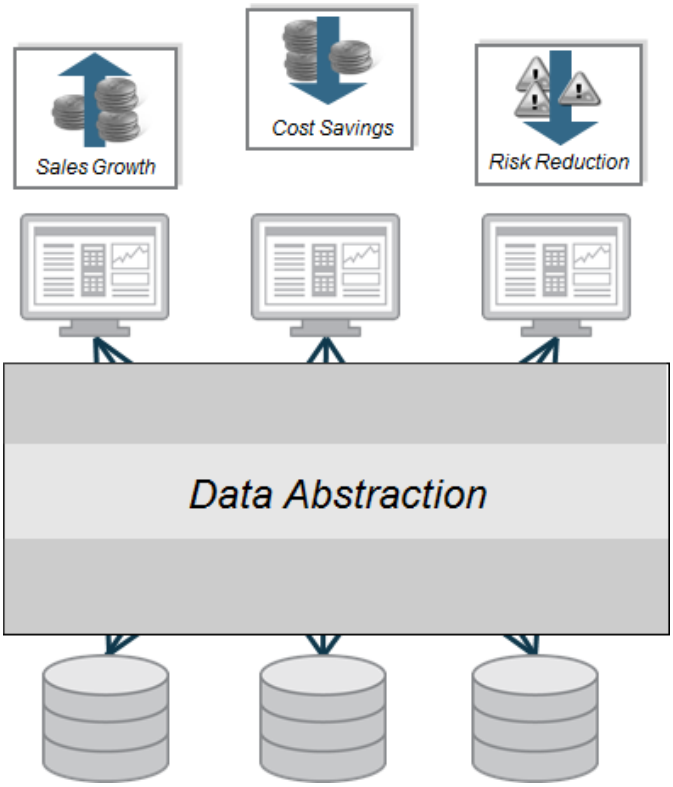

Figure Two: Data Abstraction

Data abstraction helps enterprises and government agencies achieve a number of key objectives including:

- **Right business information at the right time** – Fulfill complete information needs on demand by linking multiple, diverse data sources together for delivery in real-time.

- **Business and IT model alignment** – Gain agility, efficiency, and reuse across applications via an enterprise information model or logical business model. Often times, known as the "Canonical" model, this abstracted approach overcomes data complexity, structure, and location issues.

- **Business and IT change insulation** – Insulate consuming applications from changes in the source and vice versa. Developers create their applications based on a more stable view of the data. Allow ongoing changes and relocation of physical data sources without impacting consumers.

- **End-to-end control** – Use a single platform to design, develop, manage and monitor data access and delivery processes across multiple sources and consumers.

- **More secure data** – Consistently apply data security rules across all data sources and consumers via a unified security methods and controls.

Selected Examples

Meeting Multiple Reporting Needs via a Data Abstraction Layer – A major money-center bank used Tibco Data Virtualization in this way to support multiple reporting requirements including prime brokerage, reconciliation, risk management, and more (over 25 applications in all), from across over 200 disparate sources and thereby accelerated new reporting development and reduce IT costs.

Accelerating Time to Market via Data Abstraction – Pharmaceutical giant, Pfizer, Inc., has built a data abstraction layer to simplify and accelerate access to a wide range of research and clinical trials data across its R&D, marketing and manufacturing teams. Using Data Virtualization in this way provides decision makers with the information required to bring new drugs to market faster, while adhering strictly to FDA regulations.

2 Tibco's Data Abstraction Reference Architecture

The diagram below outlines the layers that form Tibco's Data Abstraction Reference Architecture. Architects and analysts can use this as a guide when abstracting data using Tibco's Data Virtualization platform. The various layers included in this reference architecture are described below:

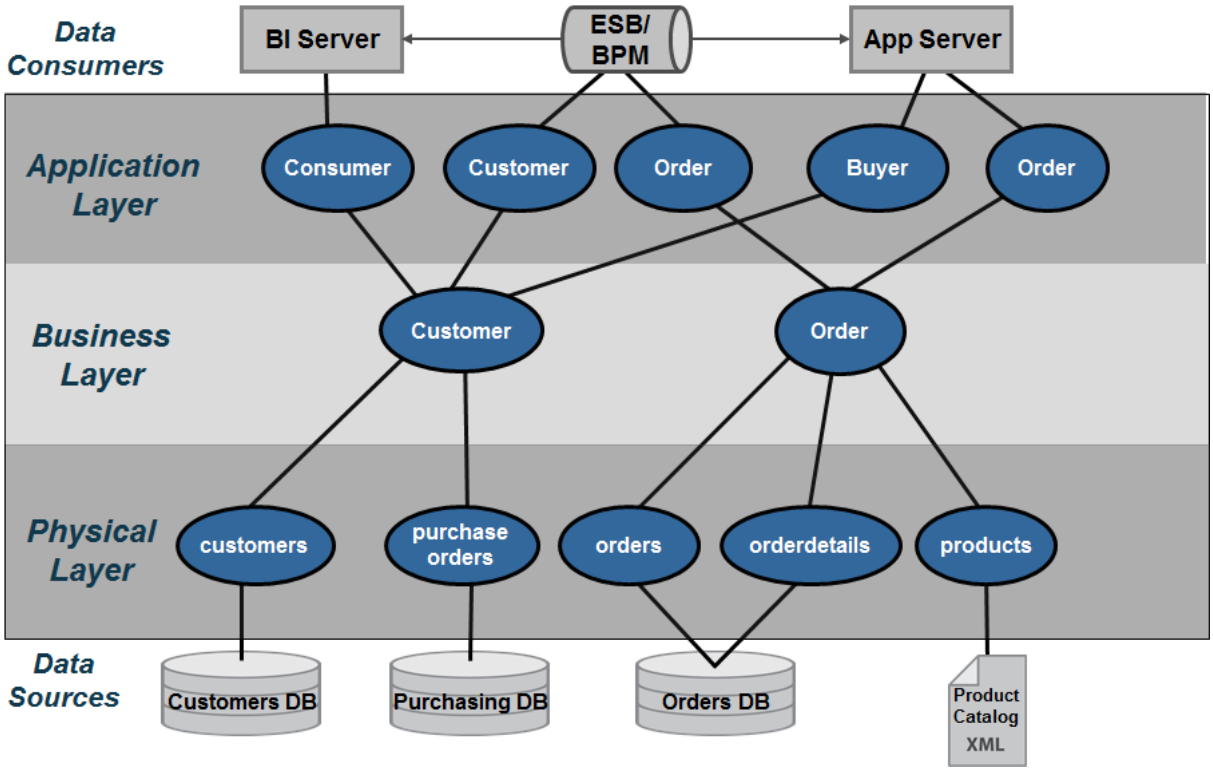

Figure Three: Tibco's Data Abstraction Reference Architecture

- **Data Consumers** – Client applications need to retrieve data in various formats and protocols that they understand. Data Virtualization delivers the data to consumers using the most popular standards including SOAP, REST, JDBC, etc.

- **Application Layer** – The "Application Layer" serves to map the Business Layer into the format which each application data consumer wants to see. It might mean formatting into XML for Web services or creating views with different alias names that match the way the consumers are used to seeing their data.

- **Business Layer** – The "Business Layer" is predicated on the idea that the business has a standard or canonical way to describing key business entities such as customers and products. In the financial industry, one often accesses information according to financial instruments and issuers amongst many other entities. Typically, a data modeler would work with business experts and data providers to define a set of "logical" or "canonical" views that represent these business entities.

These views are reusable components that can and should be used across business lines by multiple consumers.

- **Physical Layer** – The “Physical Layer” provides access to underlying data sources and performs a physical to logical mapping by integrating physical metadata and formatting views.
    - The physical “Metadata” is essentially imported from the physical data sources and used as way to onboard the metadata required by the data abstraction layer to perform its mapping functions. As an “as-is” layer, entity names and attributes are never changed in this layer.
    - The “Formatting” resources provide a way to map the physical metadata into the Data Virtualization layer by aliasing the physical names to logical names. Additionally, the formatting views can facilitate simple tasks such as value formatting, data type casting, derived columns and light data quality mapping. This layer is derived from the physical sources and performs a one-to-one mapping between the physical source attributes and their corresponding “logical/canonical” attribute name. This layer serves as a buffer between the physical source and the logical business layer views. As such, caching may be introduced at this level if and when it makes sense. Rebinding to different physical views during deployment is another role these views take on. Naming conventions are very important and introduced in this layer.
- **Data Sources** –The data sources are the physical information assets that exist within and without an organization. These assets may be databases, packaged applications such as SAP, Web services, Excel spreadsheets and more.

3 Enabling Forrester's Data Virtualization Vision

Forrester Research provides the following guidance for data abstraction in their "Data Virtualization Reaches Critical Mass" report.

According to Forrester "Leading firms have implemented a layered architecture combining both physical and virtual data stores, choosing the appropriate mix based on different areas' performance requirements. In the most-successful cases, the firms create an hourglass-shaped architecture that funnels mappings of disparate source data through canonical business information models. As a result, one large drug manufacturer is able to successfully operate with both IBM Cognos and SAP Business Objects BI tools for different customer groups, using data that has been reconciled against common metadata using virtualization.

In addition to the use of canonical models in the middle, we have identified two other noteworthy characteristics of this architecture: 1) virtual/physical modality choices tend to be more physical in the staging layers close to the actual data sources and more virtual as data moves closer to the users, and 2) a final virtual mapping layer ensures that the solution provides data to consumers in just the required format."

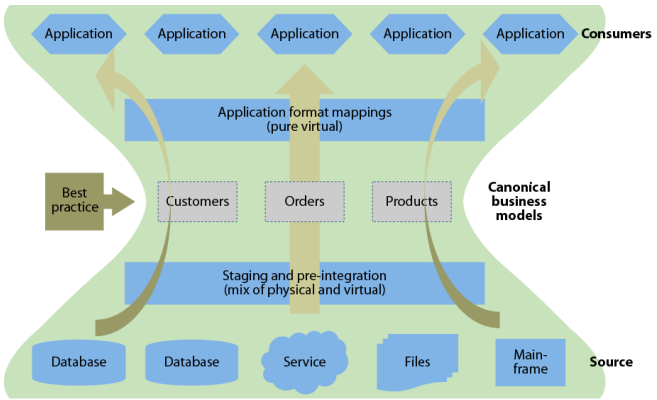

Figure Four: Source: Forrester Research Data Virtualization Reaches Critical Mass

There is a striking resemblance between Forrester and Tibco's best practice architectures:

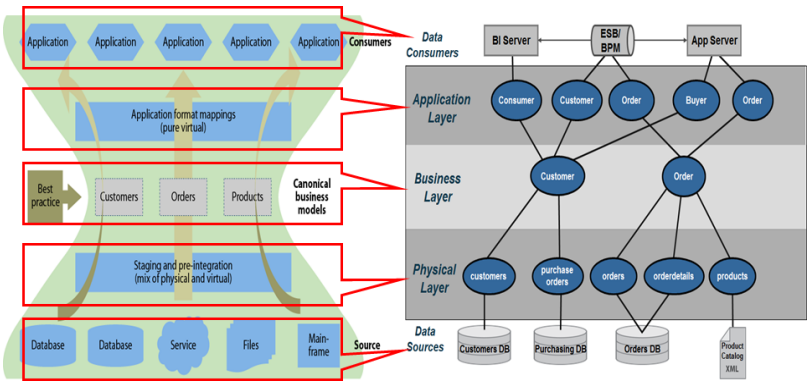

Figure Five: Comparing Forrester and Tibco's best practices architectures

4 Enabling Gartner's Discipline of Data Integration

Since 2005, Gartner has been researching the concept of data services in relation the broader business and IT evolution. More recently, Gartner discussed the "Discipline of Data Integration" at the Business Intelligence Summit as shown in Figure Six.

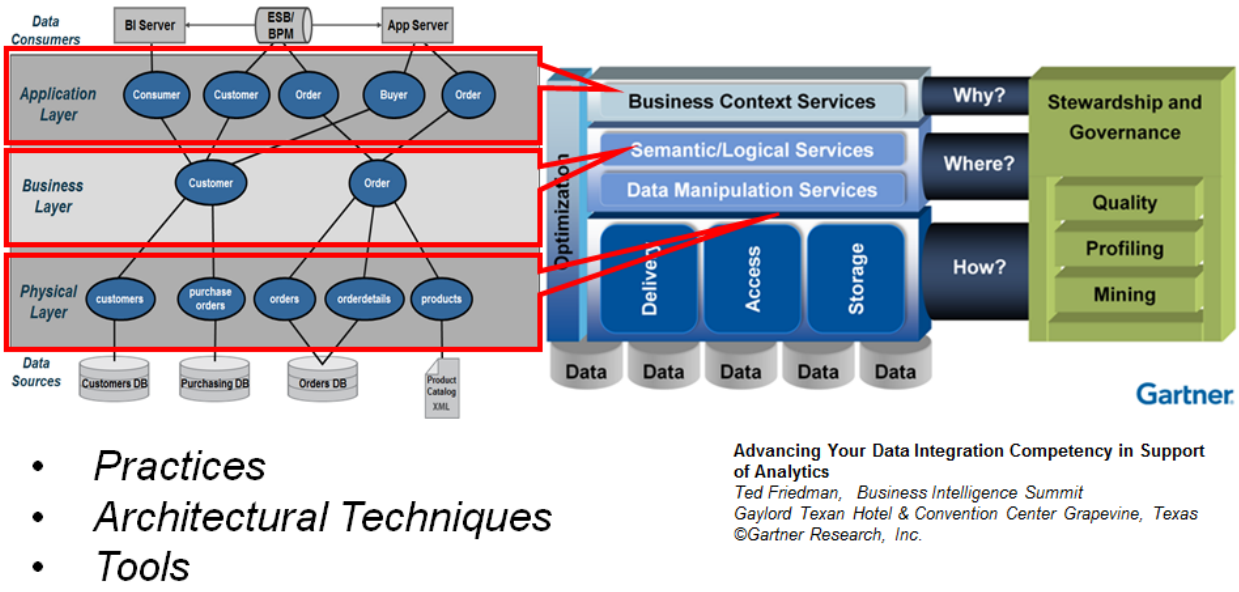

Figure Six: Tibco's Data Abstraction Architecture Implements Gartner's Discipline of Data Integration

According to Gartner, **"**The discipline of data integration comprises the practices, architectural techniques and tools for achieving consistent access to, and delivery of, data across the spectrum of data subject areas and data structure types in the enterprise, to meet the data consumption requirements of all applications and business processes. As such, data integration capabilities are at the heart of the information-centric infrastructure and will power the alignment and delivery of data in support of BI and performance management. Contemporary pressures are leading to an increased investment in data integration in all industries and geographic regions. Business drivers, such as "the imperative for speed to market" and "agility to change business processes and models", are forcing organizations to manage their data assets differently. Simplification of processes and the IT infrastructure are necessary to achieve transparency, and transparency requires a consistent and complete view of the data, which represents the performance and operation of the business."

Tibco's data abstraction reference architecture can be used to implement Gartner's "Discipline of Data Integration" as follows:

- **Practices** – Tibco has shaped the best practices that customers use today to implement data virtualization in their organizations. Tibco also has influenced the practices of recommended by leading IT analysts and system integrators. This thought-leadership and real-world experience helps users gain confidence when deploying data virtualization in their organization.

- **Architectural Techniques** – Tibco Professional Services brings a wealth of knowledge and skills to help users architect their data virtualization solutions. Tibco's architectural techniques are including in Tibco Professional Services' Quick Start program. This program is designed to help customers get a project up and running quickly and maximize their return. During this program, customers are introduced to the "Data Abstraction Best Practices Technical Guide" that Tibco Professional Services Consultants use an architectural techniques blueprint.

- **Tools** – The Tibco Data Virtualization Platform provides a complete and proven tool to implement Gartner's "Discipline of Data Integration".

- **Business Context Services** – In Tibco's reference architecture the Application Layer provides the mechanisms for mapping and publishing views or web services in the context of the applications. Tibco's Application Layer maps into Gartner's Business Context Services. Application consumers require delivery of data via different protocols. Within Tibco's reference model, data consumers use a variety of standard protocols including JDBC, ODBC, SOAP/HTTP, REST and ADO/.Net to access needed data. These standard protocols support the BI, MDM, Web service API's and Enterprise Objects consumers included by Gartner.

- **Semantic/Logical Services** – Gartner's "Semantic/Logical" services provide for the transformation of the physical model into the business context view of the information. The term logical and semantic are often referred to as canonical. It is a way of defining a common data dictionary across the business. The terms or attributes from this data dictionary are grouped together into semantically similar entities. Tibco supports these needs with its formatting views.

- **Data Manipulation Services** – Gartner's "Manipulation" functions include Access, Storage, and Delivery which align with Tibco's physical layer. This is where Tibco's introspection, discovery, and source data access tools expose the physical layer. Increasingly, Tibco is providing access to a wide array of data sources including relational, service oriented, file, packaged applications and big data.

- **Optimization** – Both Gartner and Tibco view optimization as spanning the entire architecture from source to consumer, both during design and runtime, perfectly matching how Tibco's optimizers work.

Gartner's recent research on the Logical Data Warehouse extends and enhances this guidance.

5 User Roles and Responsibilities

Implementing an enterprise scale data abstraction layer involves a variety of IT staff as can be seen in below. In this section we identify the roles and responsibilities needed for successful data abstraction layer development and operation.

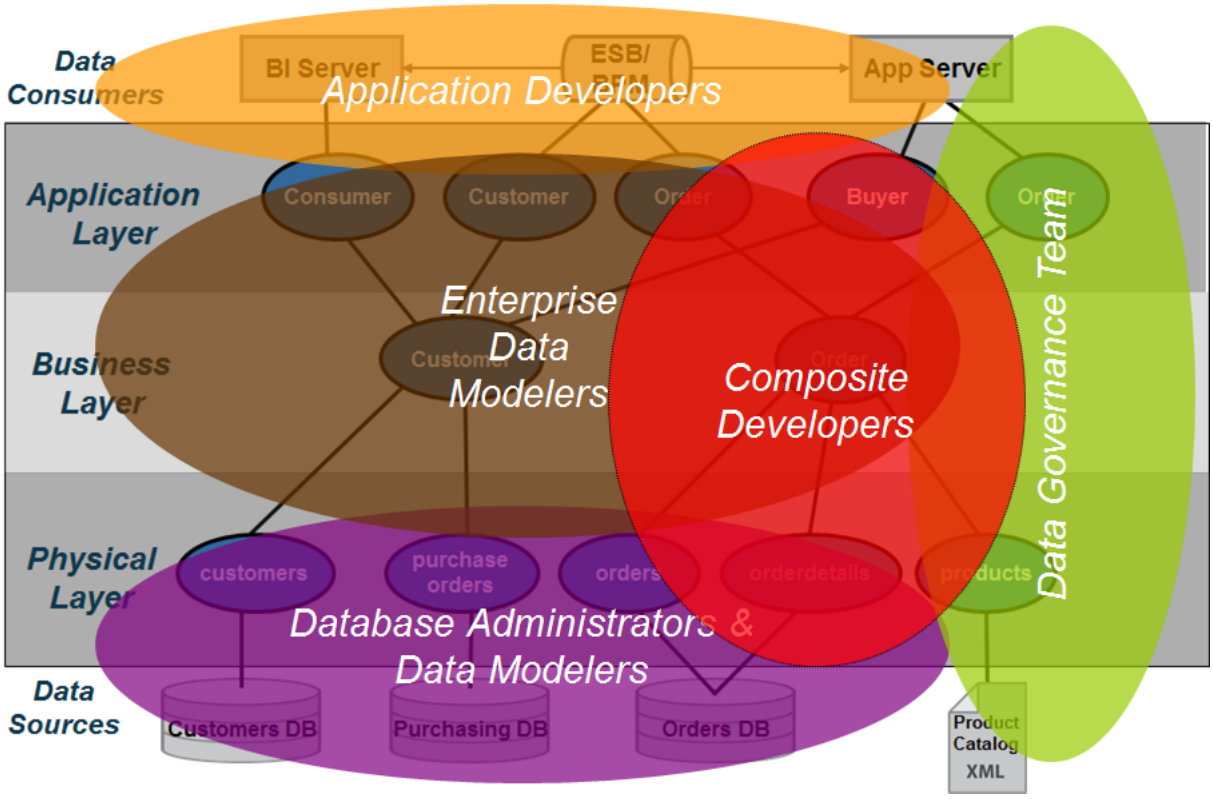

Figure Seven: Data Abstraction Roles and Responsibilities

The following roles and responsibilities should be considered when implementing a data abstraction layer:

- **Application Developers** – Application developers are concerned with the API's and the various mechanisms by which they can access the data from the data abstraction layer. Often times, data lineage is important to the client-side developers because they need to know the originating source. Tibco can provide data lineage from the client services views down to the physical metadata.

- **Enterprise Data Modelers and Data Architects** – Enterprise data modelers and data architects work with subject matter experts who know the business and use their expertise to craft logical data models. These logical data models can be further refined into something that is closer to a logical database design which could be used as the basis for building views in the business layer. Enterprise data modelers work with Tibco developers to design effective views and services.

- **Tibco Developers** – Tibco developers are responsible for the implementation of the layers using the Tibco data virtualization middleware. This implementation is based on the overall architecture and concepts laid forth in this document and performed in conjunction with the other teams involved. They need to understand the application developer requirements so they can construct the API layers and views above the business layer as well as the underlying sources. Beyond lineage tools and folders available within Tibco, these developers often use additional control systems to understand and manage key mappings across the various layers.

- **Database Administrators (DBAs) and Database Modelers** – DBAs and database modelers provide access to the physical data sources. Further they help define logical database designs from their corresponding physical database designs and implementations. They work with Tibco developers to ensure are properly tuned and assist with creating indexes if necessary.

- **Data Governance Teams** – Data governance staff are involved throughout the process making sure that enterprise data access and data mapping rules are followed. Further, they provide guidance on how data should be modeled and often help to resolve data quality issues.

6 Data Service Design Guidance

In this section, the data service design and development efforts required to implement Tibco's data abstraction reference architecture are described starting with the physical layer and moving up to the application layer. This is representative of how a typical Tibco developer approaches this activity. The next section will go into more detail about the sub-layers that may be utilized during implementation.

It is important to keep in mind that any reference architecture is just that. It is a guide. Each customer must assess what makes sense for them and use the layers accordingly. These guidelines have matured over the last several years as Tibco Professional Services have worked with customers and refined these best practices. At Tibco we believe, that if you do follow the data abstraction best practices that developers will have an easier time managing, finding and deploying resources.

Sub-Layer Architecture View

The sub-layer architecture view outlines three primary layers shown on the left of the diagram that contain two or three sub-layers shown on the right of the diagram. At the bottom of the diagram are the data sources or information assets within an organization. At the top are various data consumers that need access to these information assets. This section will describe the purpose of each layer.

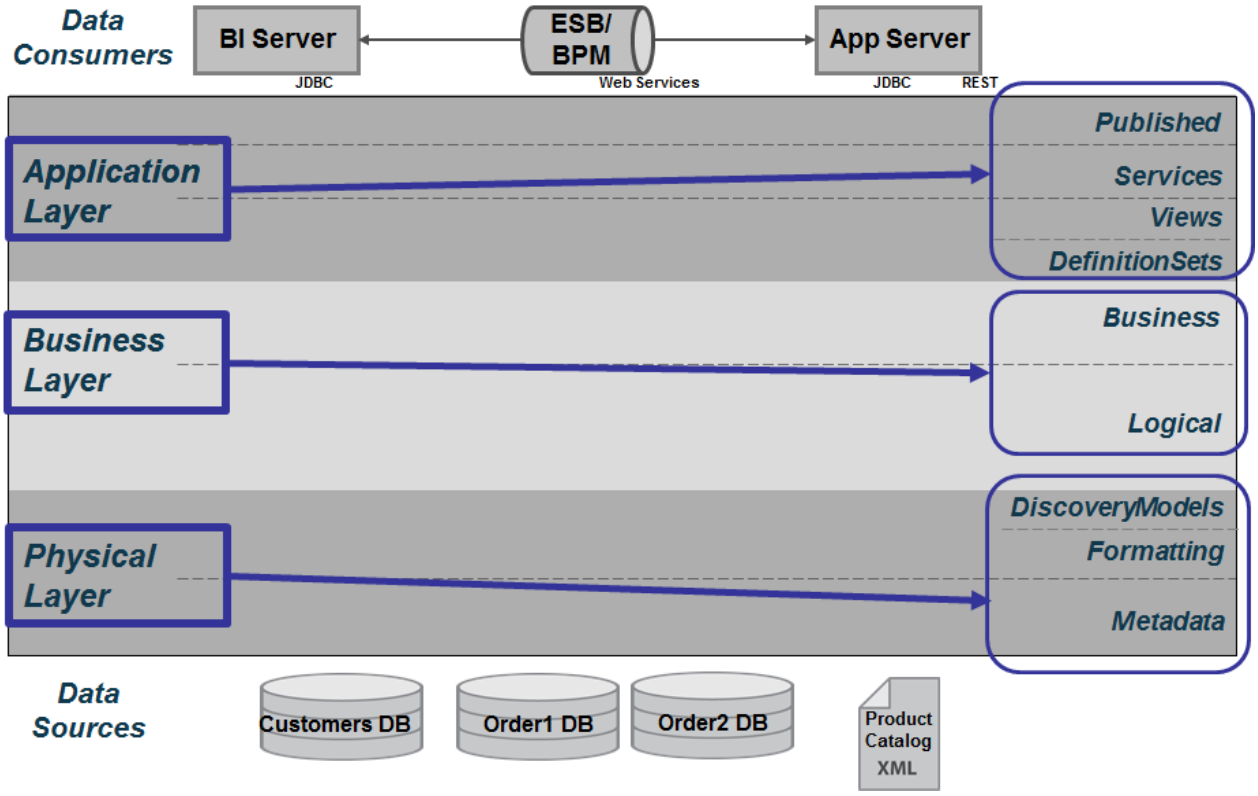

Figure Eight: Sub-Layer Architecture View

Sub-Layer Definitions

The sub-definitions are provided as follows:

- ***Application Layer*** – this layer serves to map the Business layer into the format which the Data Consumer wants to see their data. It might mean formatting into XML for web services or creating views with different alias names that match the way the consumers are used to seeing their data.
    - ***DefinitionSets*** – Provides a central place to manage various WSDL, schema and SQL definition sets. When working with web services and shaping of XML documents, the WSDL and schemas will be used by Client Published and Client Services resources.
    - **Published** – Provides the contract with the consuming application. These resources can be views or procedures that establish a contract with the client application.
    - ***Services*** – Procedural logic for shaping results, applying parameters, custom logic and XML shaping.
    - ***Views*** – Application/Client Views map the Business Layer's business and logical views to the names used by the client API. Additionally, it may be necessary to pivot data for use in the UI application.
- ***Business Layers*** – this layer is predicated on the idea that the business has a standard or canonical way to describing their business. In the financial industry, one often accesses information according to financial products, instruments and issuers amongst many other entities. Typically, a Data Modeler would work with business experts and data providers to define a set of "logical" or "canonical" views that represent the business. These views are reusable components that can and should be used across business lines by multiple consumers.
    - **Business** – Business Layer Business resources include views and other resources that implement business rules by narrowing sets of data via where clauses or aggregating data.
    - ***Logical*** – Business Layer Logical resources include views and other resources predicated on the canonical business entities or concepts. These views perform regular single-source joins, federated joins and federated union of physical layer sources. It may also be necessary for the logical views to pivot columns to rows.
- ***Physical Layer*** – the physical layer provides access to physical sources and implements the first level of abstraction by mapping physical to logical names.
    - **DiscoveryModels** – The Tibco Discovery Models provides a place to store the introspected discovery model information. Because Tibco Discovery is targeted at finding relationships in the physical data, it makes sense to store the models in the Physical Layer close to where the actual physical source metadata is located.
    - ***Formatting*** – Formatting resources include views and other resources that perform the mapping of physical to business logical. Ultimately, physical data sources have to be mapped into this virtualization layer and it is the job of the formatting views to provide simple tasks such as name aliasing, value formatting, data type casting, derived columns and light data quality mapping. In general, these views are derived from the physical

sources and perform a one-to-one mapping between the physical source attributes and their corresponding “logical/canonical” attribute name. Naming conventions are very important and introduced with these views.

- ***Transformations*** – The transformation sub-folder provides a location in which XML (hierarchical) to Relational mapping can be performed. Typically, an XSLT will be used to transform hierarchical XML documents into table/row set type structures. This allows for easier mapping and joining with Logical resources.

- ***Metadata*** – Provides the metadata imported from the physical data sources. Consider the physical metadata as the way to onboard data sources into Tibco giving developers a tangible entity create views from. Entity names and attributes are never changed in this layer. It’s an as-is layer.

Sub-Layer Naming Convention Overview

The **sub-layer folder naming conventions** overview provides a quick snapshot of the guidelines used for naming conventions of the layers. The top layer is the project name, for example: “DataAbstractionSample”. There are some standard folders beginning with underscores. The remaining layers alphabetically coincide with the layers presented in the previous diagrams starting with “Application” layer, “Business” layer, and “Physical” layer as shown in the figure below:

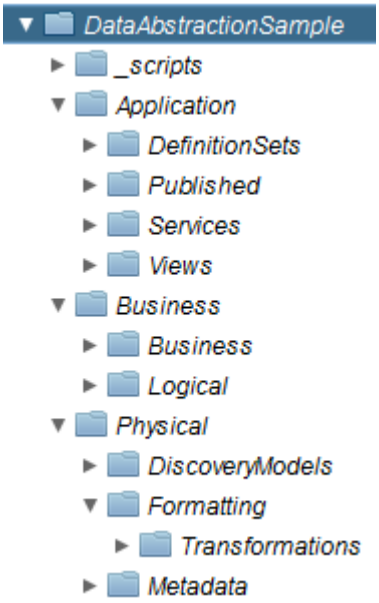

Figure Nine: Sub-Layers Naming Conventions

The **subject area folder convention** is a way of further categorization folders in each of the sub-layer folders. Over time as the number of resources increase, it is necessary to create sub-folders simply to be able to improve maintenance. It helps developers who are not familiar with a set of views to be able to put context around their purpose. Let’s explore the different folder structures and what makes sense.

- *Physical Layer*

        - *Metadata – Create sub-folders that define what the data source is. This will help when using the generation scripts. The generation scripts provide an option to pass in the "derivedFilter" name or partial path. The gives the developer a chance to narrow the scope of what they are generating. By placing data sources into sub-folders it aids the developer by being able to focus the generation on a specific folder and thus will take less time to generate especially if a regenerate is required.*

        - *Formatting – Use the same sub-folders that are defined in the Physical Metadata as the formatting views are one-to-one with the data source and the association still remains strong. Since customers know their data very well, there is a compelling reason to continue the association into this layer.*

            - *Transformation – Use this folder to transform XML to relational. Maintain the concept of data source folders for easier maintenance.*

    - *Business Layer*

        - *Logical and Business – Both of these layers break the barrier and association with the physical data. The folders that are created in these two layers should be focused on business cases and grouping of views into like concepts. Take for example the super class of "PRODUCT". It would make sense to have a Product sub-folder with various sub-class views created in that folder. The same would hold true for "CUSTOMER" and "ORDERS". Consider the use of broad-topic sub-folders when creating views in the Logical and Business sub-layers.*

    - *Application Layer*

        - *Application Definition Sets, Views, Services and Published – These layers represent what the consumer wants to see. They may simply be the same concepts that exist in the Logical and Business View or sub-folders may be organized according to client usage. Customer should consider their application and determine what makes the most sense.*

The **location of non-view resources convention** is a discussion about where to put non-view resources such as procedures, packaged queries, triggers and etc. There are different schools of thought and there is no hard and fast rule on this. Ultimately, it is up to the development team. Let's explore a couple of schools of thought based on the experience of Tibco Professional Services:

- *Proximity School of Thought – Create non-view resources in the proximity of the folder (subject-area) that they are being created for.*

- *Application Service School of Thought – Place all procedures in the Application Services sub-layer no matter if they are common or subject-area specific.*

- *Combination School of Thought – A combination of the two schools of thought would place common or generic procedures which could be used by all resources across the layers in the Application Services sub-layer. Place specific sub-area resources in*

proximity to the folder they represent. Promote resources to Application Services if they become reusable across different layers.

View and Column Naming Convention Overview

Naming conventions for views and columns is important to establish at the beginning of the project. You may choose different case styles at different layers. It is important to establish consistency and stick with a convention once it is established. Another aspect of naming conventions applies to aliases and abbreviations used in words.

Case Style rules are as follows:

- ***Original Case –*** The original case designation simply allows for generating a view with whatever case the original view already contains.
- ***CamelCase*** – First letter of each word part is upper case and remaining word part is lower with no separators.
- ***javaCase*** – First word part is lower case. Following word parts are Camel Case.
- ***Title_Case*** – First letter of each word part is upper case and remaining word part is lower with separators retained.
- ***UPPER_CASE*** – All word parts are UPPER case with separators retained.
- ***lower_case*** – All word parts are lower case with separators retained.

Abbreviations and Aliases – Use of abbreviations is acceptable (e.g. ID = Identifier). Establishing a data dictionary of aliases and abbreviations is important. The list should be driven by the data modeling and governance teams. Whatever the list is, consistency of use is the key. Don't overload terms. Here are some common examples to give you an idea:

Abbreviation		***Alias***
CD	:	Code
DT	:	Date
TS	:	Timestamp
DS	:	Datasource
TYP	:	Type
DESC	:	Description
ID	:	Identifier

View/Entity Names – Some customers may choose to keep the entity names the same as the Physical Metadata. Others may choose to make the entity names more user-friendly and transform the names to their logical counter-part. Some examples include:

- Example 1 style: Entity names in UPPER Case with no spaces and uses underscores to separate word parts.
- Example 2 style: Title_Case with underscores separating word parts.

Column/Attribute Names – Based on customer requirements. Some examples include:

- *Example 1*: Attribute (Alias) names in CamelCase with no spaces and uses underscores to separate words parts.
- *Example 2*: Attribute (Alias) names in all UPPER CASE with the use of underscores much like the original database source would provide.

Name Composition – It is recommended to not start names with a number, special character, or underscore as it causes those names to be double quoted within Tibco paths.

Name Length – It is recommended to keep resource names reasonably short. Reasonably short is not defined here on purpose as it is customer subjective. However there are two considerations to keep in mind when naming views and folders.

- *Version Control System (VCS) – when checking and checking out resources using one of the Promotion and Deployment Tool (PDTool) supported VCS systems on windows specifically; there is a 260 character windows path limitation that you may run into. Keeping folders names, depth of folders and resource names under that limit is important. Additionally, PDTool and PDToolStudio attach the resource type to the end of the resource name in order to distinguish two different types of resources with the same name. For example the view "getCustomer" is actually "getCustomer_table.cmf" and the procedure "getCustomer" is really "getCustomer_procedure.cmf" in the file system.*
- *Another use case has to do with caching views. If you are using Oracle as the target cache database then there is a column limitation of 30 characters. When Tibco creates the cache table it puts double quotes around the view and column names. Therefore, when caching views the view and column names should not be longer than 28 characters. The Common_Model_v2_file[1-3].xls spreadsheet contains a column that provides a length check so that developers can easily see if they will be violating the rule. This is not a mandatory rule. It is guidance to help the developer determine if they will exceed the limit before the view is created. The developer must consider if that view will ever be cached in Oracle or not. If it will, then the view columns must be shortened to 28 or less characters.*

Physical Layer

The Physical layer is composed of three sub-layers that contain the physical representation of the data sources, the first layer of data abstraction and Tibco Discovery Models. The physical "Metadata" structurally looks exactly like the data source; however it does not store any data. It is a virtual representation of the data source. The "Formatting" resources provide the first level of data source insulation that occurs which allows developers to transform attribute names, configure caching and rebind sources from Development, UAT and Production. The "DiscoveryModel" provides a place to store models generated by Tibco Discovery. Even though the "Physical" views have been deprecated in this release, the developer still has access to and may generate "Physical" views which provided a one-to-one mapping between the physical "Metadata" and these views. The purpose of these views has been subsumed into the "Formatting" resources sub-layer. The one potential use-case where they may still be useful and applicable is for CRUD operations. The "create, read, update, delete" operations may only operate against views that do not contain any new or derived fields. Generally speaking, the generated "Physical" views are the closest to the physical metadata that would never contain any new or derived fields.

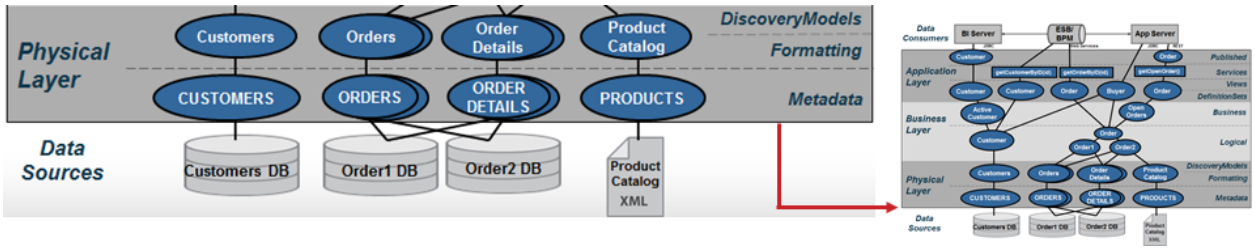

Figure Ten: Physical Layer

Naming Conventions

Naming conventions are established at this layer so that folder ordering within Tibco is visually consistent with the presented diagrams.

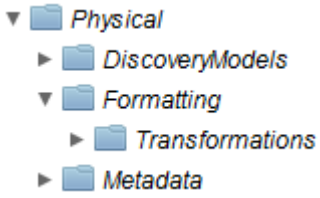

Figure Eleven: Physical Layer Naming Conventions

- ***Data Source Location*** – In general, the data sources get imported to the Physical Metadata folder. However, optionally you may choose to put this in a common folder such as /Shared/Common/DataSources. It is recommended to create additional sub-folders with the names of the data sources under this common folder. This allows greater control and flexibility when defining privileges and using the best practices generation scripts.

- ***Metadata*** – The naming conventions will be exactly the same as the data source because that is how Tibco does the import of the metadata. The screenshot in the example below shows a subset of the metadata inventory and catalog.

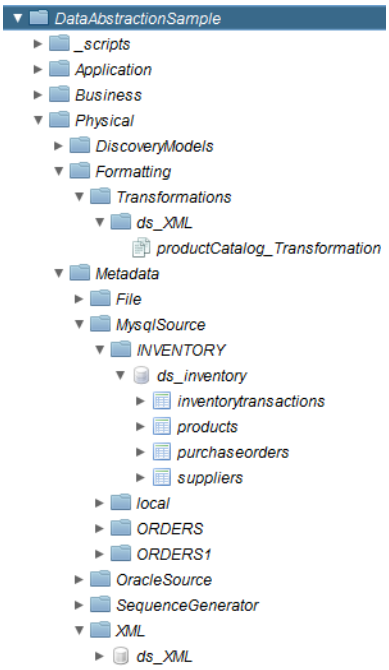

Figure Twelve: Example of Introspected Objects within the Physical Metadata layer

- ***Formatting*** – The naming conventions may vary based on customer requirements. In the example below the entity name "CUSTOMER" uses the upper case convention while the attribute names use the camel case convention.

```
SELECT
    CAST(CUSTOMERS.CUSTOMER_ID AS INTEGER)          CustomerId,
    CUSTOMERS.COMPANY_NAME                          CompanyName,
    CUSTOMERS.CONTACT_FIRST_NAME                    ContactFirstName,
    CUSTOMERS.CONTACT_LAST_NAME                     ContactLastName,
    CUSTOMERS.BILLING_ADDRESS                       BillingAddress,
    CUSTOMERS.CITY                                  City,
    CUSTOMERS.STATE_OR_PROVINCE                     StateOrProvince,
    CUSTOMERS.POSTAL_CODE                           PostalCode,
    CUSTOMERS.COUNTRY_REGION                        CountryRegion,
    CUSTOMERS.CONTACT_TITLE                         ContactTitle,
    CUSTOMERS.PHONE_NUMBER                          PhoneNumber,
    CASE WHEN CUSTOMERS.FAX_NUMBER = ''
            THEN NULL
        ELSE CUSTOMERS.FAX_NUMBER
        END                                         FaxNumber,
    CUSTOMERS.MOD_DT                                ModifiedDate,
    CAST(current_date AS DATE)                             CurrentDate
FROM
/shared/BestPractices/DataAbstractionSample/Physical/Metadata/OracleSource/c
isOraDemoDS/CISORADEMO/CUSTOMERS
```

Metadata

The Physical Metadata is imported from the customer's data sources.

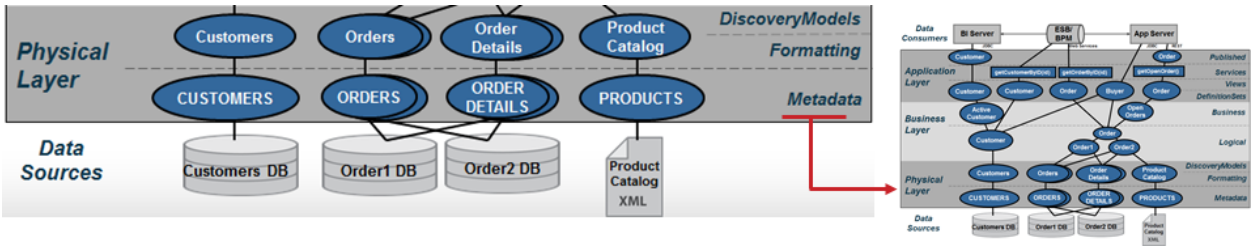

Figure Thirteen: Physical Metadata Sub-Layer

In Tibco, the tool, "New Data Source" wizard walks the Tibco Developer through a series of screens to import metadata for a source. It generates the physical metadata. There are no manual steps involved here. Physical sources can be Relational (tables, views, stored procedures, SQL Statements), Web Services, Flat Delimited Files, XML Files, Java Functions, Packaged applications such as SAP and much more. Besides capturing table and column structures, this wizard performs introspection of indexes, primary keys and foreign keys. This metadata can optionally include cardinality statistics on source tables which can be used by Tibco's cost-based optimization phase. Tibco allows for re-introspection of these sources either manually or automatically. This is important from the standpoint of a Data Abstraction Layer.

Formatting

The Formatting resources are derived from the Physical Metadata which represents the customer's data sources.

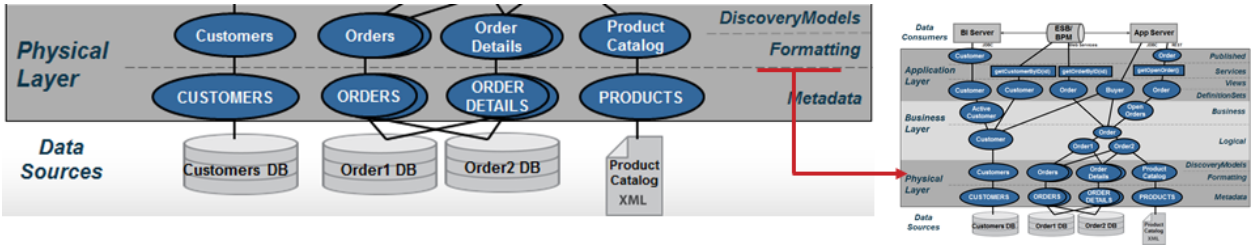

Figure Fourteen: Formatting Views Sub-Layer

The objects in this layer will map one-to-one with the Physical Metadata objects. The purpose of the resources in this layer is to provide an insulation layer that allows for transformation, rebinding of sources and caching. The views standardize the data elements into the terms or data dictionary established by an organization's business logical views. This is often referred to as "data canonicals." This layer also insulates applications and upper level views from lower level change for example when new columns are added to a data source. For applications to take advantage of those changes, the columns must be percolated to the upper views and published externally. However, this kind of change can be managed in the form of new versioned data services that provide a new signature of data while maintaining the integrity of existing applications. These views are typically SQL-based which provides for easier maintainability and understanding by the data architects.

Mapping

The general guidelines for this layer are as follows:

- **One-to-one mapping** – These views provide a one-to-one correspondence between the physical metadata and the formatting views. Generally, the names and types of data elements that the physical metadata are mapped against correspond to their "logical/canonical" name and type. An example of one-to-one mapping is shown below where the physical column names are mapped to their corresponding logical names:

```
/project/Physical/Metadata/ORDERS → /project/Physical/Formatting/Orders
    One-one-mapping with Name Aliasing:
        OID                     → OrderId
        CID                     → CustomerId
        ORDER_DT                → OrderDate
        PONUM                   → PurchaseOrderNumber
        SHIP_ID                 → ShipToId
        FR_CHRG                 → cast (FR_CHRG as numeric(12,2)) FreightCharge
        WEIGHT                  → Weight
        (FR_CHRG/WEIGHT)        →  CostPerWeight
        MTHD_ID                    → case MTHD_ID
                                        when 1 then 'AIR'
                                        when 2 then 'GROUND'
                                        else 'OTHER'
                                    end as ShipToMethod
        "ORDERS1"               → cast ('ORDERS1' as varchar) DataSourceName
        CURRENT_TIMESTAMP       → cast (CURRENT_TIMESTAMP as timestamp) StartTime
        NULL                    → cast (NULL as varchar) EndTime
```

- **Simple mappings** – Actions that are performed in this layer should be kept simple and should refrain from performing joins. The following is a list of actions that can be accomplished at this layer:
    - **Name aliasing** – Mapping the physical name to its logical/canonical counterpart. E.g. ContactFirstName is mapped to the alias Contact_First_Name.

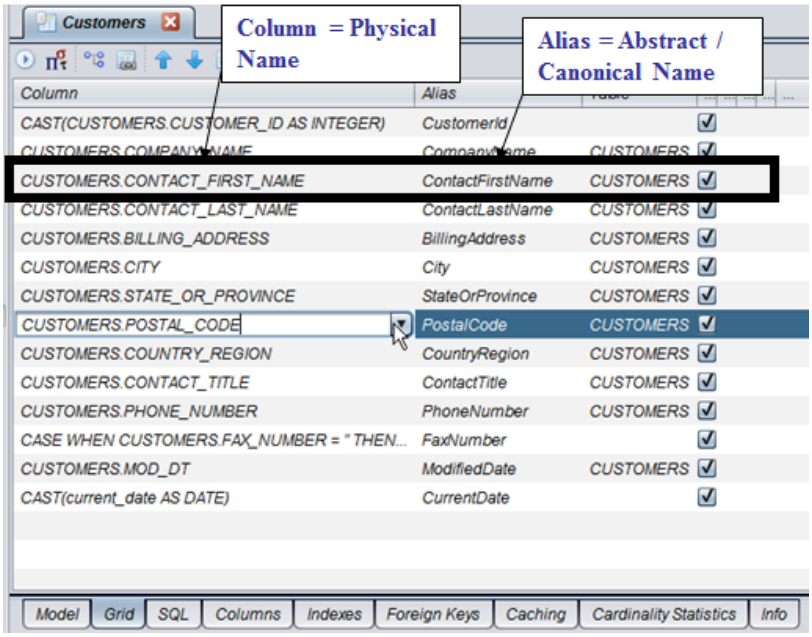

Figure Fifteen: Column Mapping Example

- **Data type casting** – Data type casting is a form of transformation whereby the type of the physical column is cast to a different type as in “cast (FR_CHRG as numeric (12,2)) FreightCharge”.

- **Simple derived columns** – Derived columns are typically columns that can be calculated from existing columns. For example, a full name can be derived from the first name, middle name and last name.

- **Value formatting** – Value formatting provides conditional logic to return a different value in place of the original value. An example would be to asses a description field and return categorization of the data. Refer to the “case” statement below:

```
case MTHD_ID
       when 1 then 'AIR'
       when 2 then 'GROUND'
       else 'OTHER'
end as ShipToMethod
```

- **New columns** – An example of a new column introduced at this level is one where the data does not exist in the source. The data is provided at the time of this view creation through a “static” definition or a “system function”.

    (i) An example of a static definition is to provide the data source name. For example, if you are mapping more than one system, it might be advantageous to the Application Developers to know where a particular row is coming from (its data lineage).

(ii) Another example is the use of a system function such as CURRENT_TIMESTAMP. A custom function could also be invoked to populate the new column with data.

- **Null mapping** – It may be necessary to establish the alias column in this layer, yet it has no corresponding physical data element to map to. In this case, it is permissible to map the alias to a NULL value. However, it is also necessary to cast the null to a specific type. E.g. CAST (NULL AS VARCHAR).
- **Light Data Quality** – Cleaning up known bad data using SQL functions, validation tables, etc.

Caching

The formatting sub-layer makes the most sense to perform initial caching of a source if necessary because the logical names, type casting and any transformation have already been applied thus reducing the downstream processing requirements. Caching of data sources can be done for various reasons:

- Protect against maintenance windows and downtime of underlying source
- Protect underlying data sources from excessive utilization
- Location transparency – guard against slow network bandwidth by caching data local to the processing

Rebinding

When promoting resources from development to test/QA/Staging and production, it may be necessary to rebind the formatting views to a different physical metadata source. This may be as a result of a schema or path change in the data source. This can be accomplished in several ways including the following: manually, package import, custom scripts, using the Tibco Promotion and Deployment Tool (PDTool), or the rebind scripts provided as part of the Best Practices Scripts through a Tibco Professional Services engagement.

Transformations

The general guidelines for this sub-layer are as follows:

- **Flat structures** – Flat structure like those from databases, delimited files, custom java cursors and etc. are simply mapped into the “Formatting”. There is no additional transformation that is needed within this layer.
- **Hierarchical structures** – Hierarchical structures like those from XML files, REST and Web Service sources require a thin layer of transformation to turn their hierarchy into a tabular output that can be consumed in the “Formatting” layer. ***Figure Sixteen*** below demonstrates a technique where a “Transformations” folder is created under the subject-area folder. In this example, the subject-area folder is called “ds_XML”.

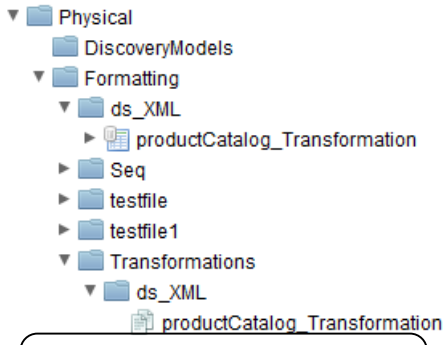

Figure Sixteen: Discovery Models Sub-Layer

The name should be short, yet descriptive of the data source. The "Transformations" folder contains an XML-to-Tabular transformation using one of several strategies:

- *Basic XSLT Transform* – This transformation accepts an XML or WSDL source as input and generates a tabular
- *XSLT Transform* – The XSLT transformation lets you define how the XML data should be transformed using a graphical editor in Tibco Studio. It accepts an XML or WSDL source as input. Complex transformations are possible by writing custom XSLT. The only requirement is that the XSLT must produce tabular data with the following structure:

```
<results>
        <result>
                <column_one>a</column_one>
                <column_two>b</column_two>
                <column_three>c</column_three>
        </result>
        <result>
                <column_one>d</column_one>
                <column_two>e</column_two>
                <column_three>f</column_three>
        </result>
</results>
```

- *Streaming XSLT Transform* – The Streaming transformation lets you define how the XML data should be transformed using a graphical editor in Tibco Studio. This transformation is useful for transforming a large amount of XML data. This transformation does not require the entire XML source document to be realized in memory like the "Basic and XSLT Transform" do.
- Transformation Editor Transform – the Transformation Editor (release in 6.2.3) will provide the ability to graphically construct XML to Tabular format. The capabilities go far beyond this use case.
- XSLT Procedure – The XSLT Procedure has no graphical editor but allows a developer to write or paste into the XSLT panel any XSLT 1.0/2.0 compliant syntax. The parameter output may be a complex cursor (tabular) format.

Discovery Models

The Discovery Models are imported from the customer's data sources Tibco Discovery.

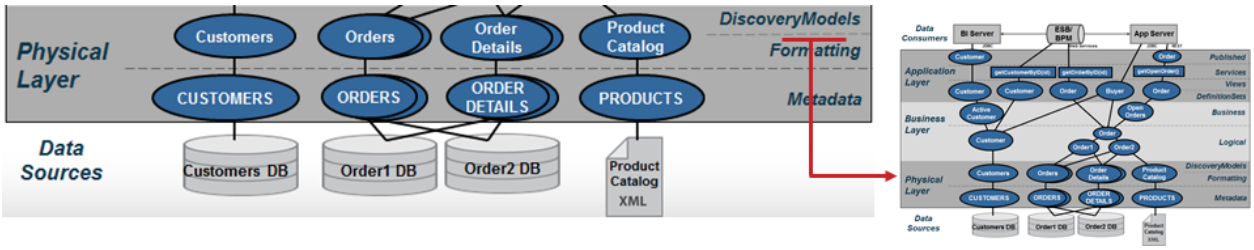

Figure Seventeen: Discovery Models Sub-Layer

Tibco Discovery is a product that reveals hidden connections in your enterprise data and uses that knowledge to help you quickly build rich data models that enable data virtualization and reporting efforts. Working within CIS and Tibco Studio, you can use Discovery to scan data and metadata from across your data repositories, whether it is applications, databases, or data warehouses. You can then use Discovery's robust set of graphical tools to create data models based on system metadata and the discovered relationships. From the data models, you can then create views in Tibco Information Server (CIS).

Tibco Discovery is an add-on product to CIS and uses the Tibco Studio user interface and many of the built-in features within CIS. Tibco Discovery is delivered as a software executable which contains all of the software needed to implement Tibco Discovery in your IT environment.

Business Layer

Business layer views and services are grouped into subject areas defined by an organization's enterprise information model. The business layer is a logical or canonical representation of the key business entities and supports federation of data across multiple data sources. Often data modeling tools such as ER/Win and ER Studio are used to create a logical data design. These models can be used as the basis for the views and data dictionary at this level. Naming of objects should reflect the logical entity and attribute names determined by data modelers. Because this layer serves to federate multiple, like views together to form a single unified result set, naming used in the underlying formatting layer should be consistent with the Business Layer.

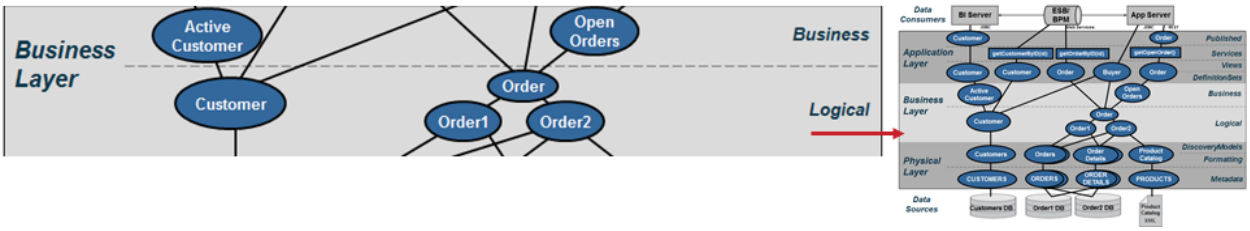

Figure Eighteen: Business Layer

Naming Conventions

Naming conventions are established at this layer so that folder ordering within Tibco is visually consistent with the presented diagrams.

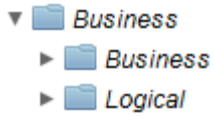

Figure Nineteen: Business Layer Naming Conventions

- ***Naming*** – Naming of objects should reflect the logical names determined by data modelers. Use sub-folders to establish groups of views according to business areas and objectives. It will make it easier to find views with like concepts in a sub-folder.

- ***Sub-folders*** – It is recommended that developers create subject area sub-folders in order to aid in the maintenance of the application over time. It reduces the number of resources in any one folder and helps other developers who are not familiar with the application to identify with these subject areas.

Logical

The Business Layer - Logical resources are grouped into subject areas. The Logical views are a projection of joins, transformations or views from the Formatting sub-layer. This layer may serve to federate multiple, like views together to form a single unified result set. It is vital that this layer only access other logical views or formatting views and 'never' access the physical metadata. Going directly against physical metadata would break the data abstraction insulation and cause problems during deployment.

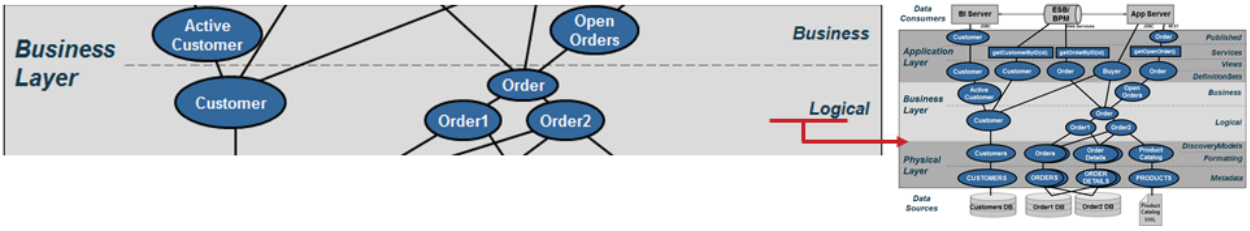

Figure Twenty: Logical Views Sub-Layer

The Business Layer in general may be a combination of Views and SQL Procedures. Views do not have logic other than joins, filters or parameters. SQL Procedures provide logic and are much like stored procedures in databases. SQL Procedures can access logical views or abstract views. The diagram below shows an example of joining two views from the formatting views sub-layer to form a single logical view of Order.

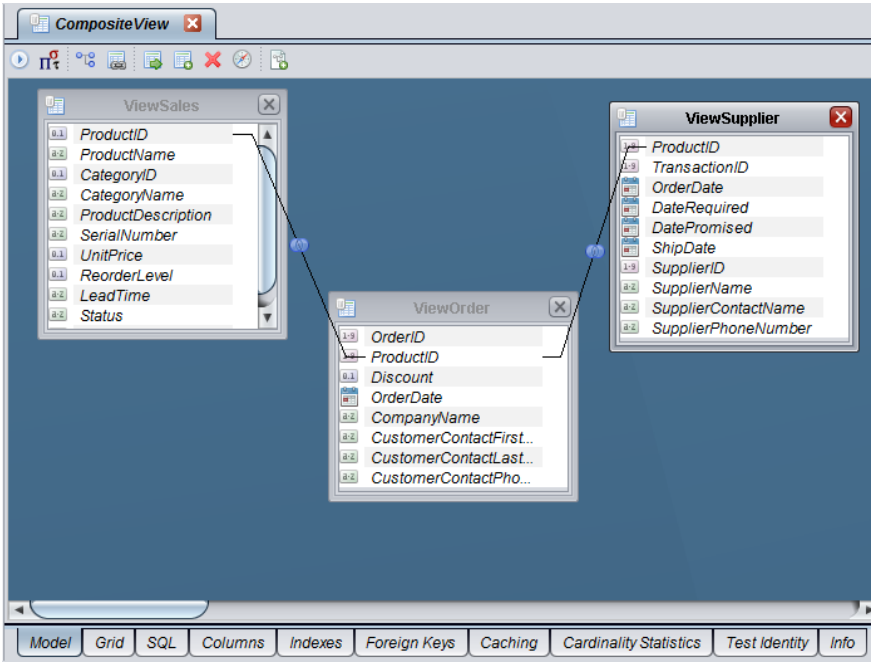

Figure Twenty One: Logical Business View Example

Within the logical views it is sometimes necessary to create federated Views that either join or union disparate data sources together. Tibco provides a mechanism to create a union of similar views that return data sets from each underlying source and merge them into a single result-set. It provides the capability to push-down filters and execute the respective queries in parallel.

It may also be necessary for the logical views to pivot columns to rows. For example, when a data set provides columns of information for periods like 3, 6, or 9 months, CIS will need to pivot those columns to rows in order to apply aggregation and roll-up of data in the business views. It makes more sense to execute the pivot in the Logical Views so that one or more business views can take advantage of the pivoted view.

Typically, no “where clauses” are applied in this view. A possible exception to this statement is when you require the ability to view current vs. historical data in which case a where clause with a sub-query and MAX(timestamp_var) would be needed to get only the current data from a warehouse

Business

Within the Business Layer it is sometimes necessary to create Business resources from the Logical resources. The purpose of business resources is to provide filtering, aggregation, custom logic and unions of other business or logical views. This layer is a refinement of the business strategy and use cases. Filtering is achieved by applying “where-clauses” which narrow the scope of the data and return specialized sets of data based on business rules. Additionally, data can be aggregated at this layer.

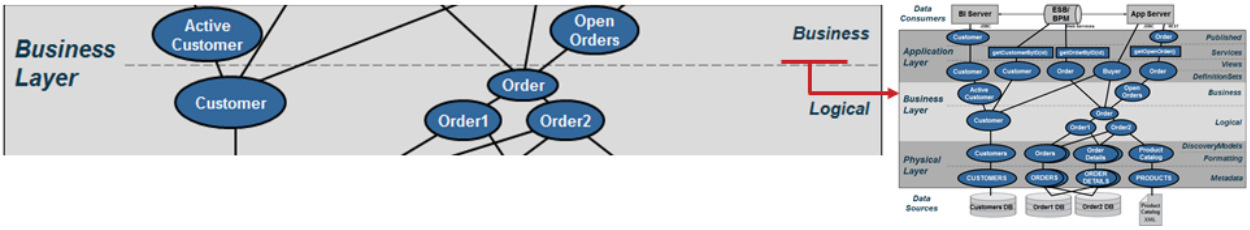

Figure Twenty Two: Business Views Sub-Layer

Be careful about the use of custom business logic in the context where views invoke procedures. Procedures are opaque and the optimizer cannot push the query to the database. In many cases, it is possible to break down the logic of a procedure and implement it as one or more views. However, there are times that a procedure makes the most sense. The following example below shows how to narrow the scope of a query and create specific subsets of data. This view would be called “Open_Orders” and provides a client service such as “getOpenOrders()” an easy way to return a result set.

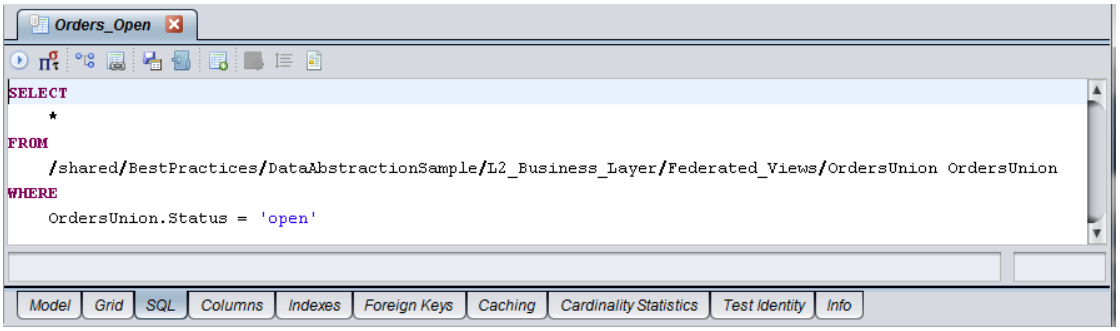

Figure Twenty Three: Business View Example

Application Layer

Application Layer views and services prepare the final output into the context required by the client application. It is recommended that these views and services match up to client API structures (object model, XML complex types and SQL tables). In other words, the client structure dictates the naming conventions and view contents. If XML shaping is required, then Tibco provides XQuery, XSLT or SQL procedures with XML functions for shaping XML. If SQL result sets are needed, then a SQL procedure can be used to provide the output as a cursor. Tibco SQL views or SQL procedures can be published as database resources. Web services can be published through top-down (a.k.a. contract-first) design or bottom-up (auto-generation) design methods.

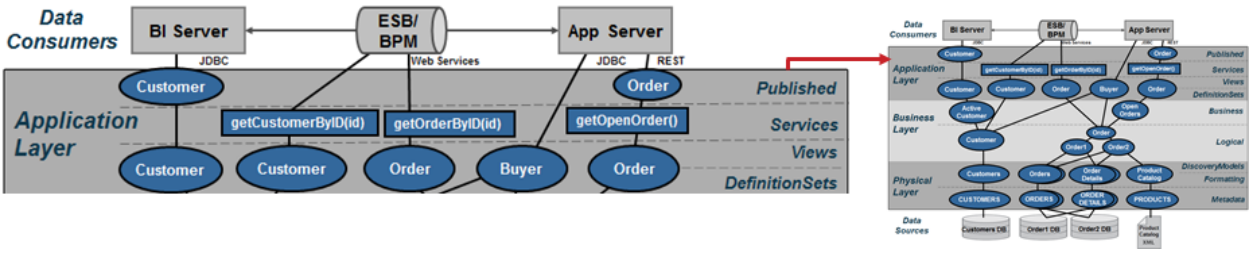

Figure Twenty Four: Application Layer

Naming Conventions

Naming conventions are established at this layer so that folder ordering within Tibco is visually consistent with the presented diagrams.

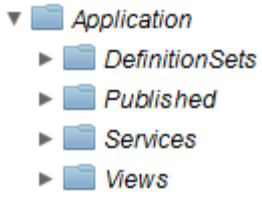

Figure Twenty Five: Application Layer Naming Conventions

- ***Sub-folders*** – It is recommended that developers create subject area sub-folders in order to aid in the maintenance of the application over time. It reduces the number of resources in any one folder and helps other developers who are not familiar with the application to identify with these subject areas.

- ***Naming*** – Naming of views and attributes should reflect the names determined by the application. If the application is new and can accept the same names that are being used in the Business Layer, then use those. If the application pre-dates Tibco and cannot be changed and thus requires Views and Column names to be in a specific format and naming convention, then use the Application Layer to re-shape the Business Layer views into views required by the application. Apply whatever convention that is needed. The example view below "ORDERS" maps the logical names into the application-centric view using the name aliasing to abbreviations mentioned earlier.

```
SELECT
    Orders.orderID                  ORDER_ID,
    Orders.customerID               CUSTOMER_ID,
    Orders.orderDate                ORDER_DT,
    …
    Orders.dataSourceName           DATA_SOURCE_NM,
    Orders.validStartTimestamp      VALID_START_TS,
    Orders.validEndTimestamp        VALID_END_TS
FROM
/shared/BestPractices/DataAbstractionSample/Business/Logical/Orders Orders
```

Definition Sets

Application Layer – Definition Sets serve the purpose of proving the containing the web service "contract-first" WSDL and Schema definition sets. Contract-first web services will point to the WSDL definition set. Contract-first implementation procedures will point to the WSDL definition set schema. The XQuery Mapping Editor and Transformation will use the WSDL and Schema definition sets to map results into XML.

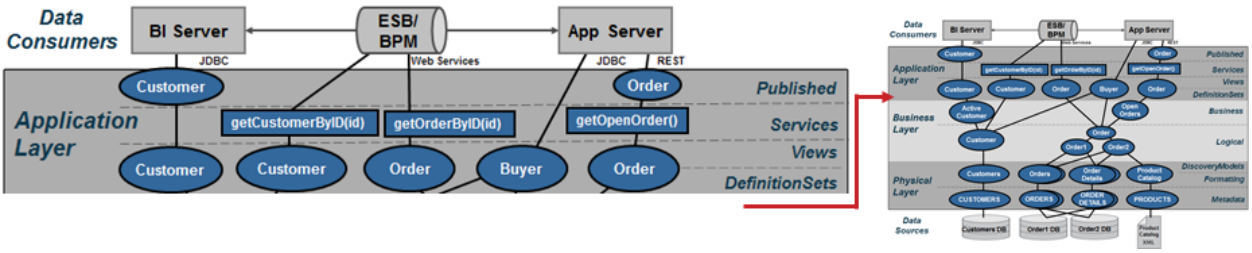

Figure Twenty Six: Definition Sets Sub-Layer

Purpose:

- **Centralize Definition Set location** – Provides a central location to store all WSDLs, Schema and SQL definition sets.

Views

Application Layer – Views serve the purpose of preparing the final output into the context required by the client application. It is recommended that views match up to client API structures (object model, XML complex types and SQL Tables). In other words, the client structure dictates the naming conventions and view contents.

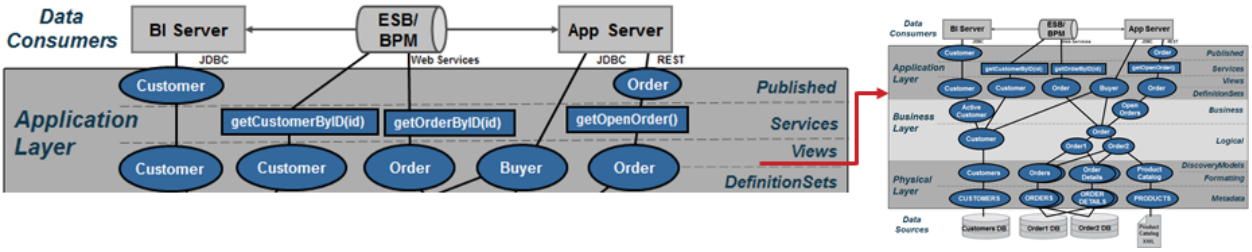

Figure Twenty Seven: Client Views Sub-Layer

Purpose:

- **Client API Abstraction** – The Application Layer is the client API abstraction layer and names of items should reflect the needs of the client API object.
- **Business to Client Mapping** – This layer serves as the business to client mapping and may involve a projection of joins, transformations or views from the business layer.
- **Client Caching** – Data can be materialized (cached) intelligently at this layer to provide the best possible consumer response and throughput.
- **Pivot** – A common pivot is converting rows to columns. An example of pivoting rows to columns is a timeframe like 3, 6 or 9 months periods for a given unit of measure and pivoting the data to columns so that a UI graph can display those time periods more easily and efficiently.

Services

Application Layer – Services serve the purpose of providing additional procedural logic on top of the Application Views in preparation for delivering information to the client application. This might require SQL Procedures or XML-Shaping procedures if XML shaping is required. CIS provides XQuery, XSLT or SQL Procedures with XML functions for shaping XML. If SQL result sets are required that are different from the logical views then a SQL Procedure can be used to provide the output as a cursor. Any of these procedures can be published as web services. CIS SQL Views or SQL Procedures can be published as database resources. This is commonly known as bottom-up design. SQL Procedure may also be used for ordering, grouping and aggregating results. SQL Procedures may expose parameterized methods in order to conform to a governance policy by the customer.

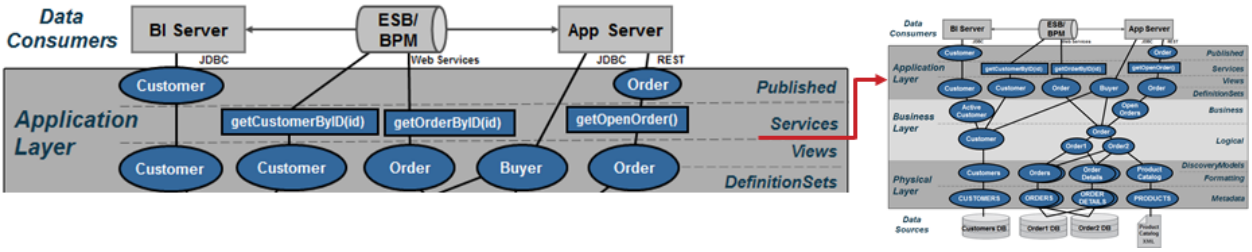

Figure Twenty Eight: Client Service Sub-Layer

Another option congruent to Application Views is the use of Tibco Studio to create a “Top-Down” or “Contract-First” design where you start with a customer WSDL and map CIS resources into that WSDL. There are three facets to consider when implementing web service that include:

- Contract-First Design – Tibco Studio imports the WSDL which contains a schema definition and operations. Tibco Studio will generate stub procedures when contract-first is selected. The developer may implement the procedure by invoking views, procedures or transformation editor procedures. This is known as “top-down” or “contract-first” design.
- Bottom-up Design – Tibco Studio can publish a view or procedure as a web service operation and it will generate the necessary schema definition and WSDL. This is referred to as “bottom-up” web service generation.
- Mapping – Provides a way of mapping Tibco resources such as views and procedures to a specific XML WSDL or Schema Definition:
    - XQuery Mapping Editor – The traditional approach to mapping views or procedures to XML is to use the XQuery mapping editor to produce an XQuery procedure that may be published to a web service.
    - Transformation Editor – Starting with Tibco 6.2.3, Tibco Studio provides a transformation editor to map resources into XML.

Purpose:

- **Data Manipulation** – Data from the Application Layer views can be manipulated by order by's, group by's and aggregation but the names from the client views should be retained for consistency throughout the organization.
- **Narrow Results** – Narrow results by selection criteria and parameterized queries
- **Publish Database Views or Web Services** – The Application Layer contains candidates for publishing as a view or web service. Shape web service results into XML.

Published

Application Layer – Published views serve the purpose of proving the "contract" with the consuming application. These views are type cast and contain exactly the same name as the underlying view it was created from. There is no other logic in these views. Only type-casting should be performed in these views.

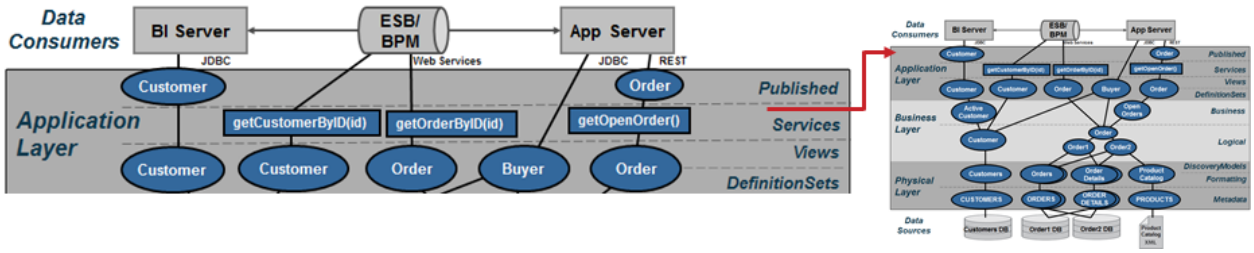

Figure Twenty Nine: Client Views Sub-Layer

Purpose:

Client Contract – The Published sub-layer provides a contact with the application consumer by type casting the columns in the views or by providing an XML schema for web services or REST access. Create procedures for mapping Tibco resources into target XML schema for web service delivery. The reason this is important is that some front-end tools will import the Tibco Published Data Services metadata into their tool and validate it on execution. If the views underneath change either by name or by type, there exists the potential for the application to throw an exception at run-time. By type-casting the views, the developer insures the contract will stay intact during execution.

7 Data Abstraction Scalability

Data Virtualization Active Cluster helps IT organizations improve their Data Abstraction implementations in the following ways:

- ***High availability*** – High availability in Active Cluster is achieved by ensuring that all Data Virtualization cluster nodes are redundant. If any one of the CIS cluster nodes fails or is removed from the cluster, the remaining nodes continue servicing the requests without requiring administrative intervention. Cluster node redundancy ensures that no data is lost and operations continue. From a client standpoint, there is no change in activity in the case of failure.

- ***Scalability*** –Active Cluster helps your organization scale the Data Virtualization environment on demand and based on your immediate needs. Dynamically add cluster nodes to address increased load requirements. Cluster nodes are automatically synchronized to maintain identical metadata. All cluster nodes are 'viewed' by clients as a single server; the clients connect to a load balancer which in turn connects the client to the next available cluster node.

- ***Reduce RPO*** (Recovery Point Objective) ***and RTO*** (Recovery Time Objective) – Active Cluster ensures your RPO and RTO objectives are met. The repository can be enabled for HA/DR (High Availability/Disaster Recovery) through native or third-party capabilities (for example, Oracle RAC or VERITAS). Migration/recovery of metadata occurs automatically, minimizing impact on CIS.

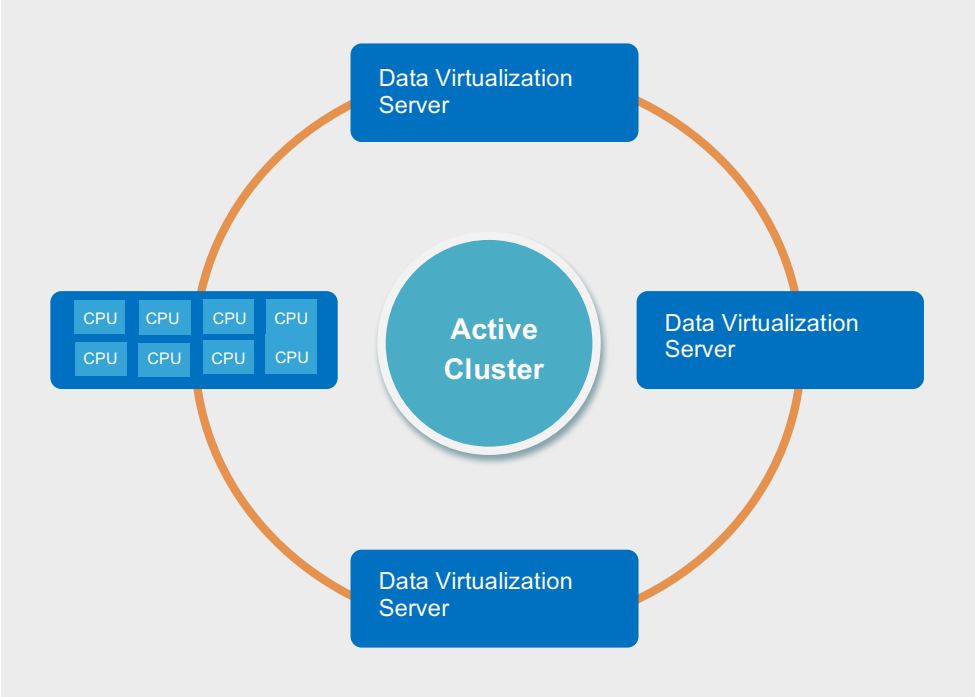

Figure Thirty: Data Abstraction Scalability

8 Summary of Key Benefits

A data abstraction layer as enabled by Tibco's Data Virtualization platform and data abstraction reference architecture provides business and IT with a number of key agility and total cost of ownership benefits including:

- **Faster development** – Leverage existing views and services to build new applications faster. Onboard new data sources faster.
- **Reduced complexity** – Use abstracted data structures that business consumers understand.
- **Simplified change management** – Insulate sources and consumers from change.
- **Higher performance** – Data Virtualization optimizes performance and scalability through query optimization, caching and clustering, and more.
- **Better control** – Unify data design and implementation within a complete, enterprise-scale data virtualization platform.

Phased implementation – IT can incrementally create the data abstraction layer over time, while immediately realizing data service reuse.

9 Practical Next Steps

Enterprises can get started on the journey to achieving the key agility and total cost of ownership benefits described above with a few simple steps. The key is getting started quickly with a manageable project that enables learning and a foundation for progress.

- **Set achievable goals** – Start with projects and a focused team. With success, broaden Business and IT team involvement to expand usage across departments toward the ultimate enterprise level deployment.
- **Determine levels of abstraction** – Are the four recommended layers right for your organization? Do you need greater depth within one or more layers? Tibco Professional Services can help answer these questions and get you started on the right path.
- **Determine modeling and mapping approach** – Should you use top-down, bottom-up, or some of both?
    - Top down – You have a vision and you want to find the data to fulfill it. This is often referred to as Contract-First design. In this approach Tibco allows you to start with your own WSDL and map Data Virtualization services to your contract.
    - Bottom up – You know what your data looks like, now how do you make it usable by others. In this approach, Tibco allows you to generate or publish resources such as SQL view and Web services directly from the Tibco introspected sources.
    - Both – Mix and match appropriately according to domains and needs.

Start now! – Don't over analyze. Getting started now with small steps is the best way to learn, progress, and gain value.