XML Style Sheets and Socrates®

Streamlining Processes through Effective Data Integration and Transformation

WHITE PAPER
Cincom In-depth Analysis and Review
This white paper provides a step-by-step process of how to accelerate business processes through effective data transformation.

**Goal:**
The purpose of this technical white paper is to highlight a best-practice methodology that will optimize business-process changes through effective data transformation, making it a source of competitive advantage in your organization.

**Audience:**
As the name implies, this document is written for a technical audience, which would include the following titles within a customer organization:
- Technical managers
- Mid- to senior-level developers
- Technical end-user

**Introduction**
In the pursuit of operational efficiency, corporate IT professionals as well as end-users experience a variety of technical barriers to business transformation and growth. One such roadblock occurs when data is not responsive to what otherwise could be positive changes in system applications and infrastructure.

Through a step-by-step process that begins with building hierarchies for more efficient data integration, you can accelerate business processes by transforming the hierarchy XML output to meet the specification requirements of third-party systems.
Step 1

Creating a Hierarchy with Socrates – The Foundation for Efficient Integration with Other Applications

The hierarchy object within Socrates is a tool that gives the user a clean and very efficient way to produce the necessary output for integration without tons of scripting.

Hierarchies Defined

Hierarchies enable you to define knowledge structures showing partnet-child relationships of Knowledge Explorer objects. For example, in a manufacturing application, it could be used to define a Bill of Materials structure (Fig. 1) of the components and subassemblies. In a legal document drafting application, it could be used to show the document structure, where the components represent section(s), clauses and/or sub-clauses of the document.

Hierarchies can be designed to provide a snapshot (generative output) of the configuration and can be tailored to supply several different views of the same configuration. In fact, Socrates can support multiple hierarchies in one model. This is significant when multiple systems are being integrated to a Socrates solution, where the XML schema for one system can be very similar or quite different than the XML schema for another (i.e., XML schemas for reports, drawings, charts, engineering utilities, etc.).

Socrates makes it very easy to create a hierarchical representation of the data as well as generation of the XML that these systems need.

The Benefits of XML – Extensible Mark-Up Language an Industry Standard

Whenever you need to send data to an external application, XML is a perfect fit for the exchange format. The extensible mark-up language consistently answers to the issues that are present with IT Directors and related IT Professionals who find themselves at the crossroads when it comes to data transformation.

Configuration solutions like Socrates often are integrated to third-party applications such as reporting tools, CAD, CRM, ERP systems, etc. How configuration data is packaged for use by these systems is an important step in defining the integration process. Utilizing XML as the exchange format has many advantages.

• Documents are easily readable
• System independent, vendor independent standard
• Can be hierarchical
• Custom parser is not needed

Figure 1. Example of a Bill of Material
Step 2

Native Socrates Form – An Effective Tool for Internal Data Transformation

XML output in Native Socrates form is always useful if you are going to use other Socrates modules to import the data. This output contains such information as the name of the hierarchy used to generate the output, the object name, the type, instance properties, etc. Figure 2 shows an example of Socrates hierarchical output.

```xml
<?xml version="1.0"?>
- <dog>
  - <node name="BaseModel" hierarchy="DamperBOM" type="list" baseClass="Subassembly_List">
    - <value name="HCD">
      <prop name="Description" type="str" value="Rectangular Industrial Control Damper"/>
      <prop name="Item_Number" type="str" value="39670872"/>
      <prop name="Quantity" type="float" value="1.0000"/>
      <prop name="Price" type="float" value="250.00"/>
      <prop name="Cost" type="float" value="100.00"/>
    </value>
  - <node name="Frame" hierarchy="DamperBOM" type="list" baseClass="Item_List">
    - <value name="G12">
      <prop name="Description" type="str" value="Frame: Galvanized 12 Gauge"/>
      <prop name="Item_Number" type="str" value="39670140"/>
      <prop name="Quantity" type="float" value="1.0000"/>
      <prop name="Price" type="float" value="20.00"/>
      <prop name="Cost" type="float" value="0.00"/>
    </value>
  - <node name="Blades" hierarchy="DamperBOM" type="list" baseClass="Item_List">
    - <value name="Steel 3V G14">
      <prop name="Description" type="str" value="Steel 3V Galvanized 14 Gauge"/>
      <prop name="Item_Number" type="str" value="34539401"/>
      <prop name="Quantity" type="float" value="1.0000"/>
      <prop name="Price" type="float" value="0.00"/>
      <prop name="Cost" type="float" value="0.00"/>
    </value>
  - <node name="Axle" hierarchy="DamperBOM" type="list" baseClass="Item_List">
    - <value name="Steel 3V Axle">
      <prop name="Description" type="str" value="Steel 3V Galvanized Axle"/>
      <prop name="Item_Number" type="str" value="34539401"/>
      <prop name="Quantity" type="float" value="1.0000"/>
      <prop name="Price" type="float" value="0.00"/>
      <prop name="Cost" type="float" value="0.00"/>
    </value>
  </node>
- <node name="Axle" hierarchy="DamperBOM" type="list" baseClass="Item_List">
  ```

Figure 2. Example of Socrates Hierarchical XML Output

This format can also be very effective when the output is used to persist dynamic data for use in other Socrates modules or reconfiguration at a later time. The output also contains, of course, the selected value (data) for each node (Socrates object).
Step 3

Enhance Data Transformation and Broaden Efficiency – On-Demand Data Transformation

However, all of the Native information is not necessary when we are integrating to third-party systems. Each system to which you are integrating has its own exchange format (XML schema) definition. Since we need to use the data for systems other than Socrates, this “Native” form can be challenging. The structure (as seen in Figure 2) is very generic and makes extensive use of XML attributes.

Compare this Native structure to a standard XML structure, like the one below.

As you can see, the node names directly reflect the name or description of the data. Using a standard style sheet, the Native XML can be transformed into a more usable form for integration. A second style sheet can then be applied to further transform the standard XML to meet the XML schema definition for the third-party system. Moreover, this two-step process can be reduced to a one-step transformation by using a style sheet that will transform the Native XML directly to the required schema for integration.

Figure 3. Example of the Transformed Socrates Hierarchical XML
Summary

On-Demand Data Transformation – A Useful Tool for Complexity Reduction

Changes in technology continue to accelerate, and the easiest technique of reducing costs and driving business alignment is through simplification. Therefore, it is critical that solutions be flexible in order to move with these advancements. Implementing technologically proven design standards, such as XML and Extensible Stylesheet Language Transformations (XSLT), is vital to a solution’s longevity. Taking advantage of the design tools (i.e., hierarchies) that are available in Socrates will allow you to easily utilize these technological design standards (see figure 4). An additional benefit of implementing the use of Socrates hierarchies and Extensible Stylesheet Language Transformations (XSLT) is that the Socrates knowledge modules become properly partitioned for other functional areas of the solution. Thus, they become independent knowledge capsules that are insulated from other non-knowledge-based changes.

Therefore, changing integration schema or third-party application(s) does not immediately require an update to the Socrates knowledge module. In most cases, a simple modification to the transformation file (XSLT) is all that is needed.

Cincom’s Commitment

Cincom is committed to helping you maximize your investment in Socrates with tools that streamline business processes. For more information, please contact Jim Wilson at jwilson@cincom.com.

Figure 4. Diagram of the Transformation Design

---

**Figure 4. Diagram of the Transformation Design**
About Cincom

For nearly 40 years, Cincom's software and services have helped thousands of clients worldwide simplify the management of complex business processes. Cincom specializes in the areas of business where simplification brings the greatest value to managers who want to grow revenue, control costs, minimize risk and achieve rapid ROI better than their competitors. Cincom serves clients on six continents including BMW, Citibank, Boeing, Ericsson, Penn State University, Milacron, Siemens, Rockwell Automation and Trane.