Paper 100-2007

Speed It Up – Active Warehousing with SAS® Data Integration: From Batch to Real-Time

Eric Hunley, SAS Institute Inc., Cary, NC
Gary Mehler, SAS Institute Inc., Cary, NC
Nancy Rausch, SAS Institute Inc., Cary, NC

ABSTRACT

The global economy has fueled a need to "speed it up" to fulfill the demand for current, reliable data for decision support. The SAS Data Integration Server, with its active warehousing capabilities, can help you meet this need. This paper presents methods and best practices for using real-time techniques that are available in SAS Data Integration Studio:

- change data capture (CDC)
- message queues technology
- service-oriented-architecture (SOA) technologies.

INTRODUCTION

Data volumes are doubling within an organization every 12 to 18 months. Expectations of accurate and trusted information to be delivered to the right people, at the right time, and in the right format, continue to be paramount regardless of geographical location or industry of business. This is especially true in today's global economy where the lists of regulation and compliance requirements are long and complicated. As a number of corporate scandals have revealed, failure to comply now can result in serious penalties, including the loss of corporate integrity and shareholder confidence. Many people within an organization carry the weight of a freight train on their shoulders each and every day, knowing that decisions or recommendations they make can have a significant impact on the entire organization, and on them as an individual. This burden is also shared by those responsible for collecting, organizing, and providing the information to the decision makers or executives within an organization. Database Administrators, enterprise architects, and information architects all face the challenge of how can they speed up (do more, process more, deliver sooner) what they are already doing without jeopardizing the quality of information produced.

Let's look at the challenge from a slightly different perspective. The drive from Cary, NC to Orlando, FL is approximately 500 miles. You might have an ambitious goal to attempt to make it in 6 hours, but there are certain risks with these lofty goals. For example, traveling I-95 is a rather risky experience on any day, let alone at speeds that would be required to get you there in 6 hours. On the other hand, taking 2 days to travel that distance is not acceptable either. The challenge is: how can you get there in the shortest amount of time, without breaking any laws (at least major ones) and without injuring yourself or anyone else in the process? You might start by thinking about the type of car you might drive, the route you might take, or how many miles over the speed limit can you travel without seeing blue lights in your rear-view mirror. However, you are likely to conclude that there is only one way to meet the challenge: FLY! In this paper, we look for alternative ways, methods, or techniques to do what needs to be done, in the time that it needs to be done, and without jeopardizing quality of the deliverable or results.

This paper provides some approaches to meeting the demands that are placed on the data or information producers within an organization. It focuses on how to deliver information at the right time (real time as some might say) by leveraging methods such as change data capture, message queues, and a service-oriented architecture (SOA). This paper also illustrates how SAS Data Integration Studio can help facilitate or automate these processes to ensure optimal efficiency and reusability throughout your organization. The concept of active warehousing can help you address the following challenges to keeping information accurate and current:

- **Data volumes surge while time constraints tighten.** Organizations struggle to handle exploding volumes of data within time windows that are shrinking dramatically. Data warehouses and data marts can no longer be loaded within the time allocated, and the few seconds available to update and synchronize operational systems are no longer enough. Up-to-date information is unavailable, systems are slowed, deadlines are missed, and IT loses credibility.

- **IT environments are cumbersome and overly complex, making change difficult, and agility impossible.** Organizations have built environments with a number of systems that are incompatible, redundant, and underperforming. These kinds of environments hamper IT speed and agility, and make consolidations and modernizations nearly impossible.

- **Information that is shared across the organization is contradictory, inconsistent, and inaccurate.** The information provided across the enterprise about customers, products, suppliers, and others does not match from one system to another, or is fundamentally inaccurate. There is no "single truth" for operational or reporting use. Interactions with customers, suppliers, regulators, and others is based on inaccurate information, resulting in higher costs, lost credibility, damage to the business itself – or worse.

With SAS Data Integration, organizations will have a powerful, configurable, and comprehensive data integration solution that can access all data sources; cleanse, transform, conform, aggregate, and manage data; support data migration, synchronization, and federation projects; and support both batch and real-time solutions for the enterprise. SAS Data Integration provides a comprehensive solution to the challenges of:

- distributed and rapidly increasing data volumes
- inconsistently defined data managed across disparate IT systems
- the high expectations of data consumers across the enterprise who depend on data to be correct and complete where and when they need it.

SAS Data Integration does this in a timely, cost-effective manner, and gives the organization the ability to efficiently manage data integration projects on an enterprise scale.

Here are a few definitions that might be helpful:

SAS Data Integration Studio – a powerful visual design tool for the construction, execution, and maintenance of data integration projects – from building an enterprise data warehouse to migrating data from applications like SAP. SAS Data Integration Studio simplifies and speeds projects with an easy-to-use interface, extensive built-in transformations, and powerful productivity enhancements, all while providing a single point of control for managing complex enterprise data integration processes. SAS Data Integration Studio is easy to learn, collaborative, and lets you build reusable processes in order to speed data integration development, both now and in the future.

active warehousing – An Active Warehouse is a warehouse that is updated or refreshed as frequently as possible to ensure currency and accuracy of information used for Decision Support, Business Intelligence, and Analytical purposes.

change data capture (CDC) – the capture and delivery of changes made to source data.

real-time – refers to delivery of information in the timeframe that it is *really* needed, or some would say, in "right-time". It has also been said that real-time is anything that is faster than you can currently do it today. The fact of the matter is that the true definition of real-time is in the eye of the beholder.

message queues – A message queue is a reliable message delivery mechanism for handling data sharing in a user-defined format. There are several widely used messaging technologies currently available. The format of the message content can be completely user defined, or it can be a format that has been commonly accepted for a particular industry segment. Commonly used message queuing systems include IBM's WebSphere MQ and Microsoft's MSMQ.

service oriented architecture (SOA) – a technology-independent way for different applications to communicate with each other. On Wikipedia (2007), SOA is described as "a style of information systems architecture that enables the creation of applications that are built by combining loosely coupled and interoperable services. These services inter-operate based on a formal definition (or contract, for example, WSDL) that is independent of the underlying platform and programming language. The interface definition hides the implementation of the language-specific service." SOA-based systems can therefore be independent of development technologies and platforms (such as Java, .NET, and so on). Web Services are a common way to implement a service oriented architecture, and one that we'll discuss in this paper.

DATA INTEGRATION WITH CHANGE DATA CAPTURE

The basic premise of change data capture is to determine which values or records have changed in a particular source of information, capture those changes, and then deliver those changes to the appropriate environment. Change data capture can help reduce the volume of data that is extracted and processed from source systems. The reduction in data that must be processed helps to achieve better scalability, and can speed up data integration activities. The result is delivery of the desired information to the desired resources in a timeframe that meets expectations or service level agreements.

Change data capture is a technique that can be used to address many of the data management or data integration challenges that IT managers are faced with. These challenges range from addressing batch window challenges to providing live data feeds to support a workflow within a business application.

- **ETL and Data Warehousing** – In order to make data available more quickly, batch windows are going from days to hours and hours to minutes, and we are seeing latency levels that might be classified as near real-time or real-time—some might even say drip feeding data as it arrives into the warehouse. This might be a combination of change data capture and message queues that are discussed later in this paper. Not meeting batch windows can results in significant loss in revenue or exposure of risk to an organization. For example, when data is unavailable to a Business Analyst who is responsible for fraud detection activities, the result can be millions of dollars in losses for each hour of unavailability.

- **Cross System Data Consistency** – Many organizations, through mergers and acquisitions, must deal with some redundancy of applications, or replication of information, and yet must keep information consistent and accurate between the two environments. Change data capture can provide the mechanism for capturing transactions or updates from one system, and then using ETL type techniques to ensure that the same information is synchronized with the sister application. Inaccuracy or lack of timeliness of this synchronization can result in wrong information being used to make corporate decisions, or can result in poor communication with a customer from a call center representative.

- **Data Cleansing and Enrichment** – It is a well-known fact that the sooner data can be cleansed, or the earlier in the information supply chain data inaccuracies can be acknowledge and fixed, the less the impact on downstream systems or decisions. Change data capture techniques may also be integrated with data quality by determining new information and passing it to a cleansing service or applying a cleansing business rule. These same services or business rules can also be used for ETL and Cross System Data Consistency processes. As stated above, the farther downstream that bad data travels, the more costly it is to an organization. It takes more effort and expense to correct than if it was fixed at the time of entry. Also, any uses of the data along the way can result in inaccurate information being used to make corporate decisions.

TRADITIONAL BULK DATA INTEGRATION OR ETL

Typically the way to process data is to do a bulk data transfer from one or more source systems into a data integration or ETL process for transforming and loading the data (Figure 1). The data would then be transformed (or business rules applied) and then loaded into the target location. In many cases the data is loaded by dropping or truncating the table and then re-loading all the desired records. Because a lot of processing is still being performed, this does not help make data available more quickly. As a result, additional transformation processing was put in place to determine changed records (after the data was extracted) and introduced more advanced loading techniques like Type 2 Slowly Changing Dimensions where new records were added, changed records were updated, and obsolete records were deleted (or closed out). This certainly helps in addressing batch window restrictions as it reduces the burden on the loading process, but still does not change the fact that many unneeded records were extracted for one or more sources systems.

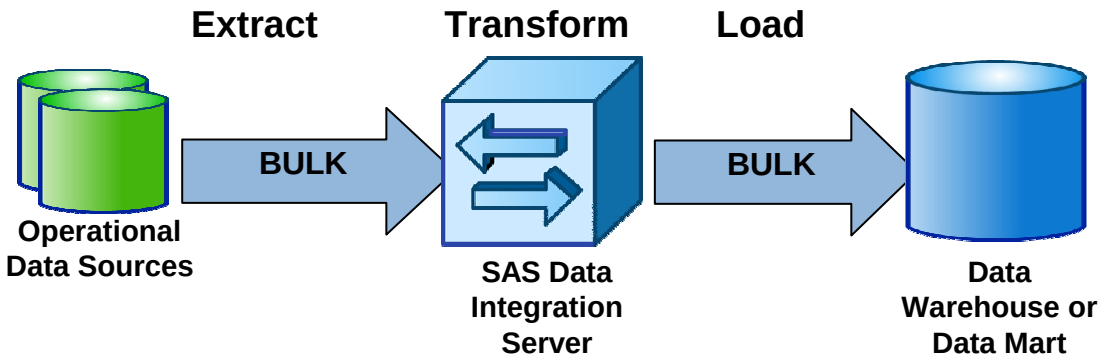

Figure 1. Illustration of Traditional Bulk Extract and Loading

Figure 2 illustrates a process flow diagram from SAS Data Integration Studio 3.4. In this example a complete data extraction, transformation, and load is completed. This may be acceptable in some situations, but considerations should be made based on the amount of data that is being extracted, how it is being processed, and where it is being loaded. If there are limitations on the batch window, additional techniques should be considered.

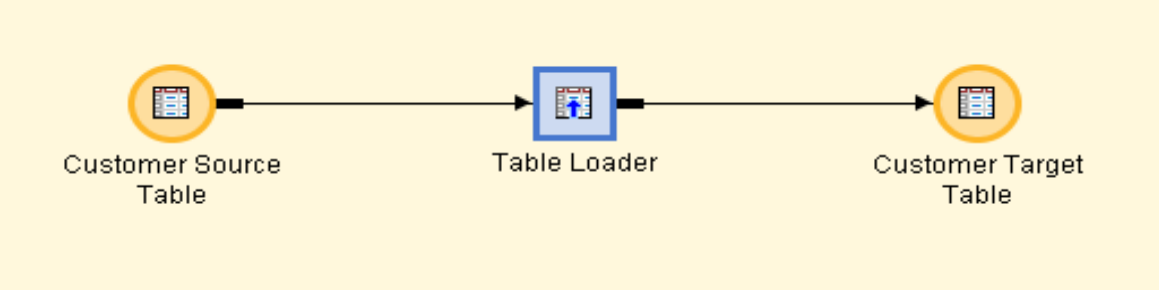

Figure 2. Traditional Bulk Data Integration or ETL – Complete Extract, Transform, and Load

Figure 3 illustrates a process flow diagram in SAS Data Integration Studio 3.4. In this example a complete data extraction, transformation, and advanced load techniques are used. In this situation a Type 2 Slowly Changing Dimension (SCD) Loader is being used to detect changes and process the records appropriately (Insert, Update, and Delete).

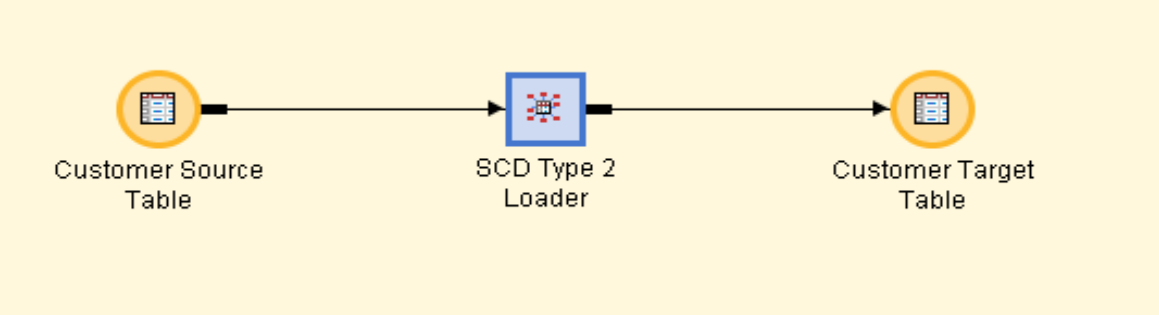

Figure 3. Traditional Bulk Data Integration or ETL – Complete Extract, Transform, and Advanced Load Technique

Figure 4 illustrates the properties for the Type 2 SCD Loader. In this example the Name, Address, Phone, and CustomerID fields are selected for change detection between the source and target tables. Records are processed based on the type of change (Insert, Update, Delete).

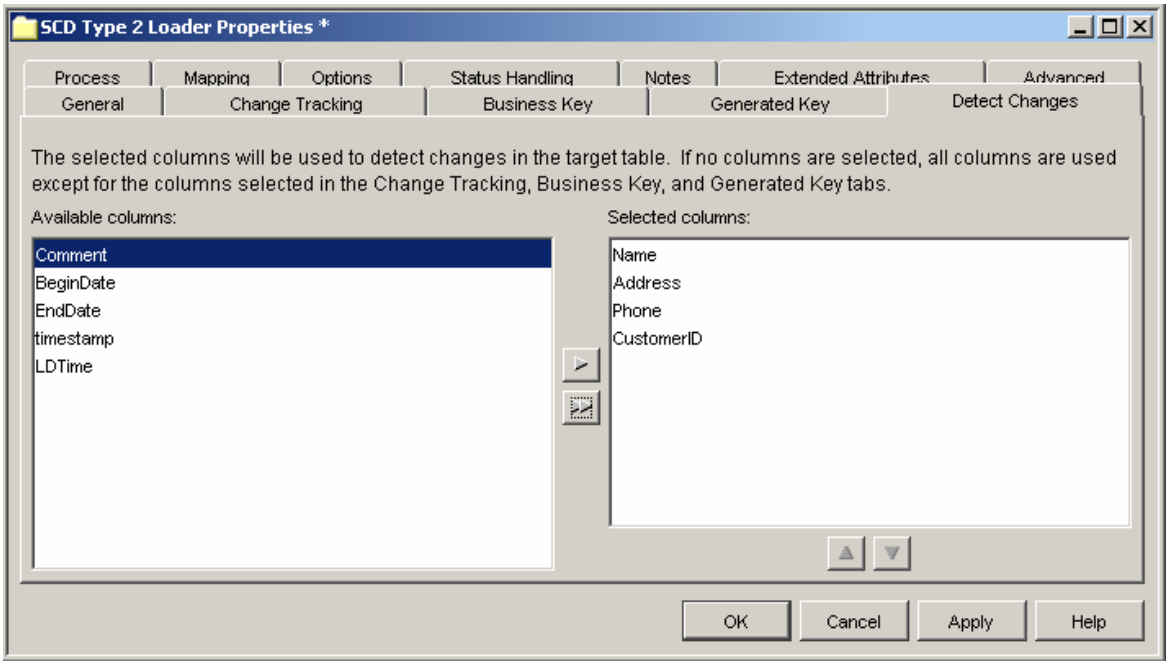

Figure 4. SCD Type 2 Loader Properties Window

CDC-BASED DATA INTEGRATION OR ETL

Taking the traditional Bulk Data Integration or ETL one step further is to look beyond simply reducing the amount of records that are moved from the transformation engine into the loading process. With change data capture (CDC) techniques, the same loading advancements can be leveraged while also reducing the total number of records that are extracted from the sources systems as well as the number of records that must be processed by the transformation engine (Figure 5). In this case, the processing burden is reduced at all three stages of an ETL process. First, only the changed records are extracted from the source systems. This certainly reduces the number of records that must be transmitted across the network, but it also reduces the time that the production systems are being taxed for both transactional processing and extraction routines to build downstream systems. Second, there is a significant reduction in the number of records that must be processed by the transformation engine. Depending on the complexity of the processes that must take place at this stage, this could reduce the overall process time by 40% - 50%. Finally, the loading or updating of the data warehouse or data mart does not require a complete refresh. Any of these benefits of CDC can be the deciding factor as to whether a batch window can be met or not or whether a service level agreement is met or not.

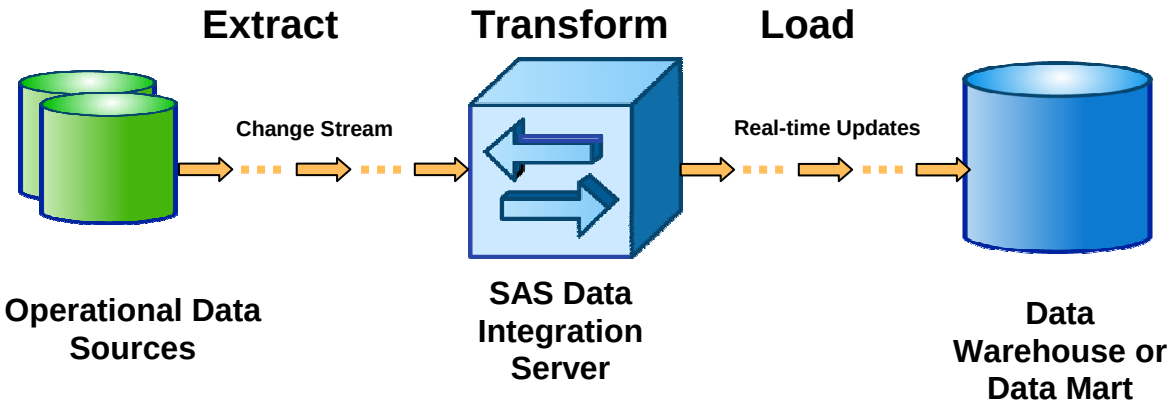

Figure 5. Illustration of Change Data Capture Process for Extract and Load

Figure 6 illustrates a process flow diagram in SAS Data Integration Studio. In this example, only changed records are extracted from the source table, and only those records go through the transformation engine; advanced load techniques are used. In this situation, a Type 2 Slowly Changing Dimension (SCD) Loader is being used to detect changes and to process the records appropriately (Insert, Update, and Delete).

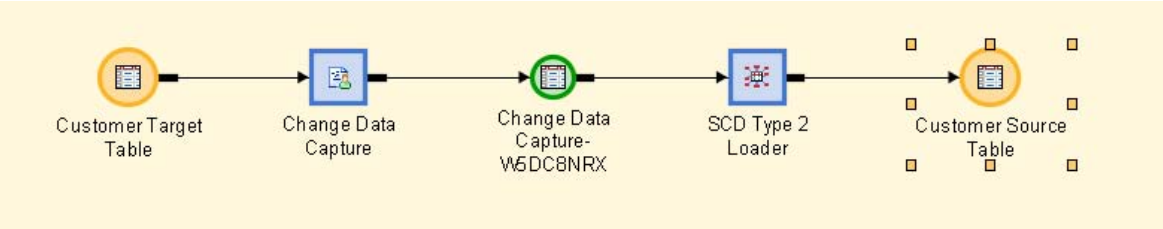

Figure 6. Extract, Transform, and Advanced Load Technique on Changed Records Only

Figure 7 illustrates the properties you might see for a change data capture transformation in SAS Data Integration Studio. It provides the parameters for representing Update, Insert, and Delete values for the inbound records.

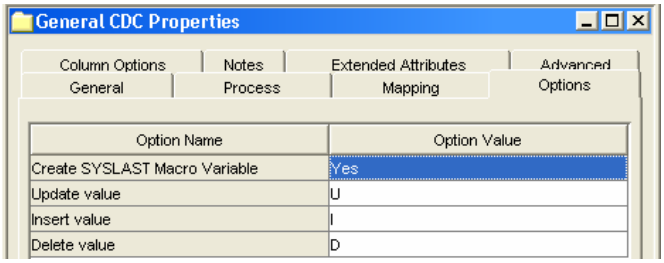

Figure 7. Example Change Data Capture Transformation Properties

From the illustrations and descriptions above, it is obvious that leveraging change data capture techniques can help meet the challenges and obstacles that IT (data providers, data architects, information architects) face every day. It can help meet some the objectives and service level agreements that are put in place between the business and IT; for example, by ensuring that all information is available or updated in the timeframe that is required by the business to do their jobs.

To conclude the change data capture topic, there are several options to consider when determining what approach to take. There are capabilities offered directly by relational database vendors that are built into their databases. Some might say these approaches are too invasive and might have negative impacts on the source systems. They might require changes to the way the database processes (triggers) or might require additional values (timestamps) to be added, increasing storage requirements. Another option is to use database logs or journals, an approach that is considered to be less invasive to the database and production systems. This approach might require specialized third-party tools, such as Attunity Software (www.attunity.com/attunity_stream).

DATA INTEGRATION WITH SERVICE ORIENTED ARCHITECTURE AND WEB SERVICES

A key benefit that SOA provides is the ability to reuse applications and data. SOA provides a framework for interoperability that allows users to build applications that can communicate with each other. This benefit becomes more important as the complexity and number of applications in use at an enterprise grows.

To illustrate this point, consider an organization that begins with a single set of custom processes to manage one aspect of their business. They then add a second area with its own custom processes. As the number of applications and custom tools written for each specific area grows, a complex network of application silos appears. This leads to redundant and inconsistent data, such as conflicting versions of customer data, employee information, product tables, and so on. When a business change occurs, such as an acquisition or merger, introduction of new product lines, or some other significant event, organizations then have to scramble to update their various custom applications in order to keep pace with changes.

SOA can help alleviate this problem, by providing a fundamentally different way of linking business information together. Applications become services that can provide information to each other. They are components that are located within a network of interconnected systems. Component developers create reusable parts that deliver data as needed, through a technology-independent protocol that uses messages to communicate. The parts become reusable and reconfigurable to meet the needs of the organization over time.

Figure 8 shows an example of a well-defined SOA process. Note that each calling application has access to the same set of services. This example shows the benefits of interoperability.

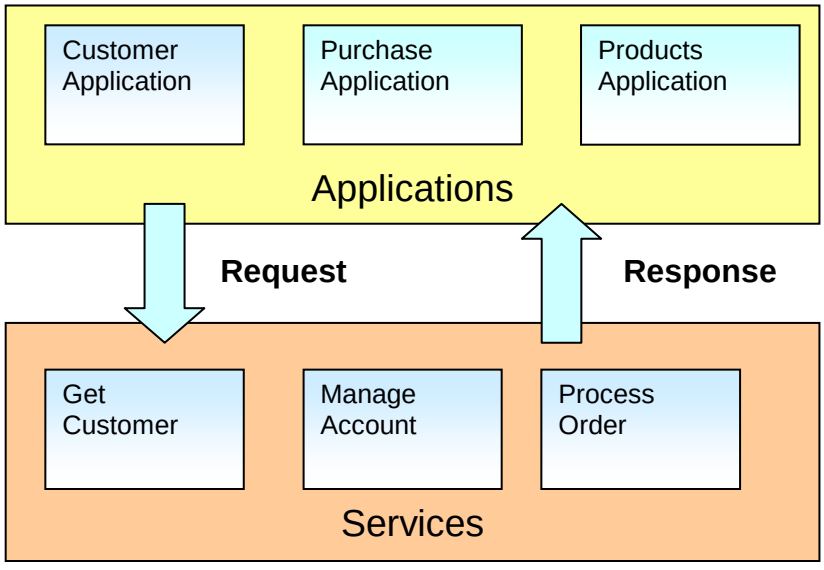

Figure 8. Sample SOA Process

A second big benefit of SOA architecture in the data integration space is the ability to get close to real-time data movement. Data integration is all about bringing data together. Traditional data integration uses batch processing, with nightly or weekly extracts to combine data. Moving to an SOA-enabled system, data warehouse architects and developers can move information more quickly, no matter where it comes from or where it is going. As a result,

participating databases and services in a SOA solution can receive new data more frequently than in a traditional batch window, thus providing more value to those using the source and target systems. SOA allows data integration specialists and data warehouse architects to support real-time data warehouse solutions.

HOW IT WORKS: SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

SOA architectures most commonly use Web services as the building blocks. Web services communicate using messages sent in the XML language and in a particular format. In the SOA world, the Simple Object Access Protocol (SOAP) provides an agreed-upon protocol for the format of the XML document so that the receiver understands what the content of the message is, and how to read it. SOAP contains the entire message. It consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body. The envelope specifies the name of the message and the data types that the message can understand. If a header is provided, it can be used to indicate that some intermediate processing can be done on the message before it reaches its final destination, such as security processing. The body contains the main part of the SOAP message, its content.

Figure 9 shows an example of a SOAP message designed to look up the meaning of a word in a dictionary. Notice the tags: <SOAP-ENV:Envelope>, <SOAP-ENV:Header>, and <SOAP-ENV:Body> that make up the parts of the SOAP message.

```
<SOAP-ENV: Envelope
   xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”
   SOAP-ENV:
      encodingStyle=
          “http://schemas.xmlsoap.org/soap/encoding/” >
      <SOAP-ENV:Header>
            <t:Transaction xmlsn:t=”your-URI”>
              .. some stuff in here
            </t:Transaction>
      </SOAP-ENV:Header>
      <SOAP-ENV:Body>
           <m:DictionaryLookup:m=”your-URI”>
                 <word>Interesting</word>
           </m:DictionaryLookup>
      </SOAP-ENV:Body>
</SOAP-Envelope>
```

Figure 9. SOAP Example Showing Envelope, Header, and Message Content

HOW IT WORKS: WEB SERVICE DESCRIPTION LANGUAGE (WSDL)

SOAP protocol describes that “what”, that is, the message and its contents. An application however still needs to know what format to use in making a request to a service, and how to interpret the messages that come back. This information is described using an XML document called a WSDL document, which contains a description of the Web service's interface. A WSDL file describes in XML, to a client of a Web service, how to call the Web service. The WSDL contains the URL (location) of the Web service, the set of operations (methods) exposed through the Web service, and the types (structure or schema of the XML that’s used in the messages to communicate with the Web service).

The WSDL file describes two methods: **Discover** and **Execute**. The **Discover** method enables the client to get a list of the services that are available and the parameters that each service takes. The **Execute** method calls the Web service with the parameters and data streams necessary to communicate with the Web service.

HOW IT WORKS: PUTTING IT ALL TOGETHER

Figure 10 shows how the pieces fit together in a SOA-enabled environment. In order to locate the services that are available, the calling application calls the Discover method for a particular URL. This provides the caller with the set of available Web services at that location. Providers of Web services (Service Provider) publish the interfaces to the service to the hosting server. This information tells the caller how to interact with the service. The calling application can then call the service using the execute method. The SOA protocols convert the application messages to and from XML, and wrap their communication in a SOAP envelope.

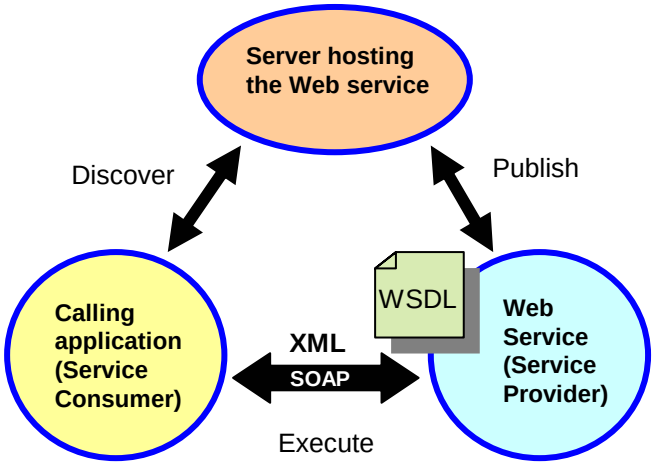

Figure 10. Connecting All the Pieces of SOA Together

CREATING WEB SERVICES IN SAS DATA INTEGRATION STUDIO 3.4

SAS Data Integration Studio 3.4 introduced the ability to easily create a Web service from any SAS Data Integration Studio job. It uses the technology contained in SAS BI Web Services and SAS Stored Processes to create the Web service. When a job is deployed as a Web service, a stored process is created in a special form that supports the standard Web service interfaces (SOAP and WSDL). The SAS Data Integration Studio job then becomes a Web service method that can be called from any Web service-enabled application, using standard SOAP protocols.

Figure 11 shows how you can create a Web service from a SAS Data Integration Studio job. Once a job is created, to make it a Web service, simply select the "Web Service" menu item from the job item in the tree:

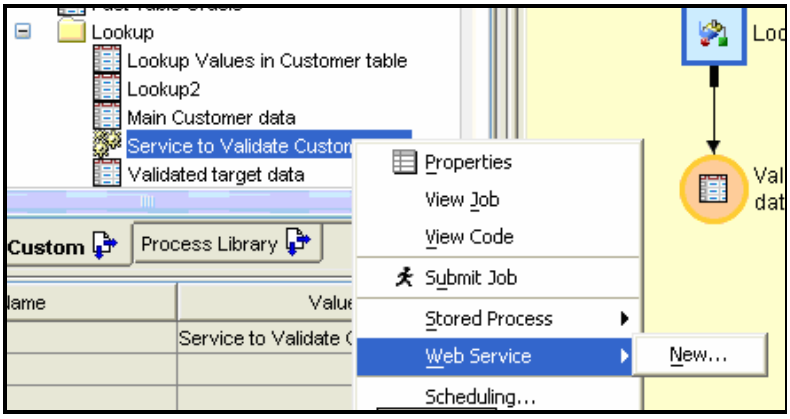

Figure 11. Creating a Web Service from a Job

A wizard walks you through the steps needed to create the service. After you name the service, you can then select the stored process server that will host the Web service, as shown in Figure 12.

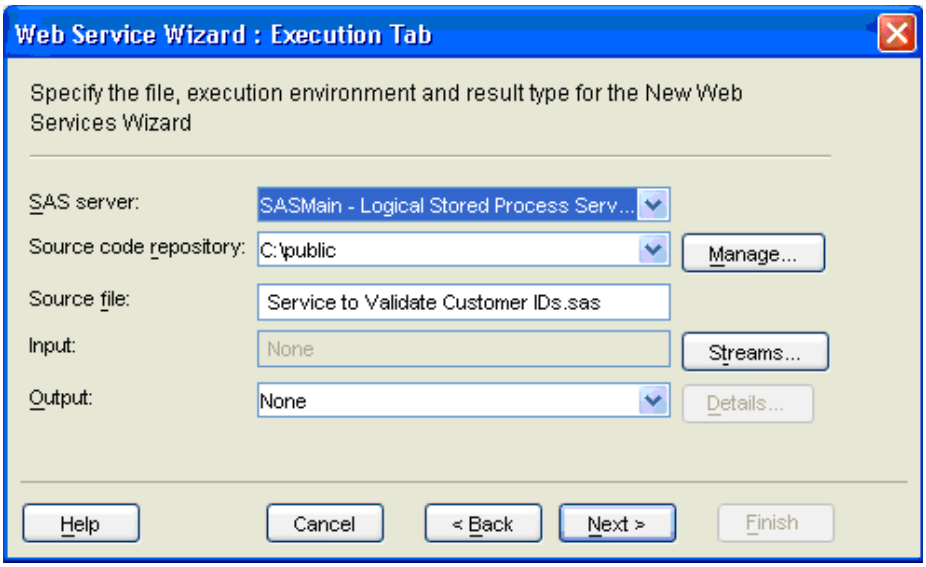

Figure 12. Wizard Page Showing How to Select the Server That Will Host the Web Service

Just as in any application, information must often be transferred between the calling application and the receiving application. The SOA protocols enable creators of the Web service to specify exactly what information the Web service supports. A SAS created Web service can transfer information in any or all of three ways: one or more input data streams, one output data stream, and through input prompts. All information is transferred using XML format.

It is easy in SAS Data Integration 3.4 to specify which input or output in any job should be defined as a Web stream. To do so, simply right-click on any table in the job, and select "Web Stream", as shown in Figure 13.

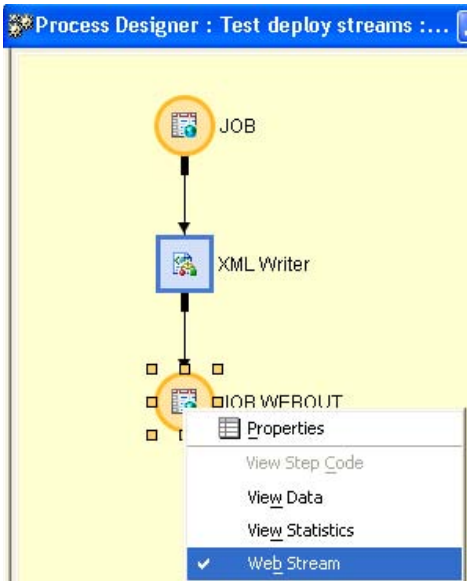

Figure 13. Flagging a Table as a Web Stream When Used in a Web Service

An icon will display on the table when it is being used as a Web stream in a Web service as shown in Figure 14. This flag has no impact on the job or table unless it is operating as a Web service, so you are free to mark any tables you need to as Web streams.

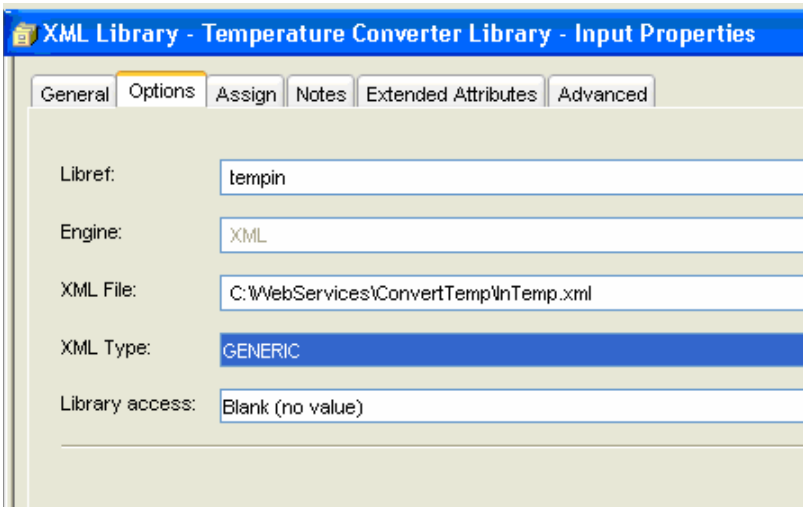

Figure 14. An example of an XML Library Definition

In order to qualify as Web streams, the tables must be defined as XML tables and connected to an XML library that uses the XML libname engine. The XML libname engine understands Web services, and when used in a Web service, is able to send and receive streaming data. SAS Data Integration Studio 3.4 introduced a libname engine template for the XML libname engine. Figure 14 shows an example XML libname definition using the new XML library template.

Streaming is a suitable transfer mechanism for communicating tabular data between applications, but sometimes you also need to provide configuration or runtime information to the Web service. For example, suppose you only want to run the application on a particular ZIP code. The caller of the application needs a way to provide specific pieces of information to the Web service. This is also defined in the SOA protocols, and is handled through prompting.

Creators of a Web service can define any number of values that they can accept from the user at runtime, by defining input parameters on their interface. In SAS Data Integration Studio, this is easily managed through the Parameter tab. An example is shown in Figure 15. To create a parameter, right-click on the open job, and select Job Properties. Select the **Parameters** tab, which will display all parameters that are defined for the job.

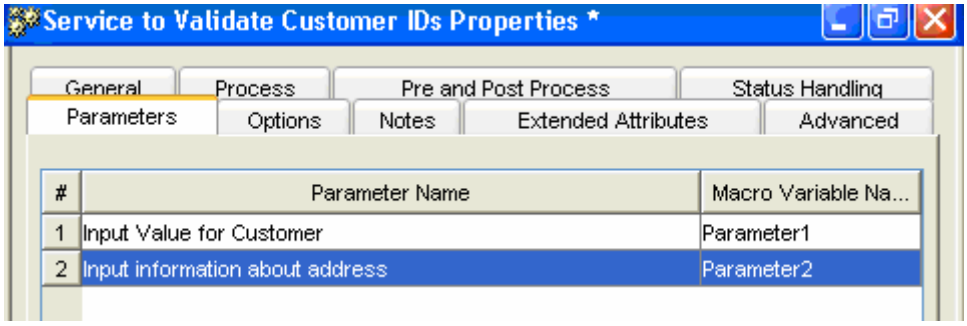

Figure 15. Parameters Tab for a Job That Will Be Deployed as a Web Service

To create a new parameter, click the **New** button, and a dialog box will appear (Figure 16) that enables you to define the prompt that the user will see and its type and structure. You can also provide a default value, which will be provided in the prompt field as an editable value when you call the Web service.

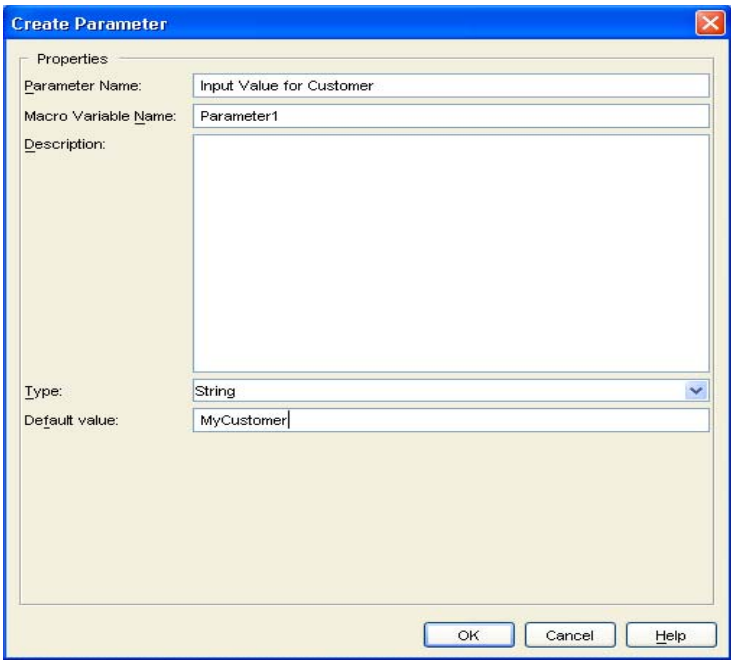

Figure 16. Parameters Properties Panel

SOME SOA CONSIDERATIONS

SOA isn't always the best choice for every situation. One limitation of SOA-based systems is that Web services, the applications that are called in a SOA environment, are stateless. *Stateless* means that you make a request, and get back a response. When you make the next request, the Service is re-set back to where it was before your first request, except for any data that is persisted during execution of the service, such as adding database records.

A second limitation is that Web services are inefficient when large data volumes are involved. In order to communicate in a Web service, messages must be converted to XML to be sent, and then converted from XML when they are received. XML is not a particularly efficient representation for data, which means that there is a lot more information that is being transferred in an XML encoded message, and there is time spent in translating from the native format to XML, and from XML back into the native format for the data.

It is important to note that performance degradation might be seen, as data volumes grow between native binary- and XML-based data transfers. Essentially, the more data you have, the bigger the difference between the native binary and XML interfaces.

In summary, SOA architectures using Web services are becoming increasingly popular as the data integration needs of organizations increase in complexity. SOA offers key benefits in terms of increased interoperability and real-time data access. The downside of using a SOA-based architecture is a decrease in performance as data volumes grow; so traditional ETL methods will remain necessary as back-end processing to Web services, or for high speed data management with large data volumes. An enterprise might want to consider an investment in SOA architectures in order to improve the overall performance and scalability of their systems.

DATA INTEGRATION WITH MESSAGE QUEUES

Another way to ensure that the most current information is available for decision support is to leverage data that is available through message queuing systems. Also known as message oriented middleware, software packages providing this type of capability include IBM's WebSphere MQ and Microsoft's MSMQ. Messages can be placed into a queue by one application, and retrieved at a later time by another. A benefit of this approach is that the two systems don't need to be always connected at the same time, yet delivery of the data into the second application is ensured.

To contrast this operation with Web services, the key difference is that the Web service is synchronous; information is sent or requested when the Web service is called, and then waiting occurs until that specific request is satisfied. Message queues can perform similar functions, but work asynchronously; data can be posted on a queue, but no specific waiting is done at that time. When the recipient of that data does read the message, it takes the needed action and can post its result back to another queue. This approach has a number of benefits including the ability to determine the response time needed on a case-by-case basis. In addition to determining the response time,

efficiency can be added by determining how many pending messages can be processed at a time. We'll discuss these choices with an example that illustrates how to work with a message queuing system in SAS Data Integration Studio.

A good use of the near real-time properties of a message queue is to perform complex analysis for a large volume of incoming requests for a product like home equity loans. In this scenario, we want quick turn-around, but don't need it to be immediate. Also, if the analysis is so complex that it takes significant processing time to complete, this is better done on a back-end server when processing time is available.

For data integration with a message queuing system, we need the ability to read and optionally write to message queues themselves. SAS Data Integration Server 3.4 provides these capabilities through transformations that provide the ability to read and write to a message queue. Figure 17 shows a SAS Data Integration Studio process flow that utilizes both of these types of transformations.

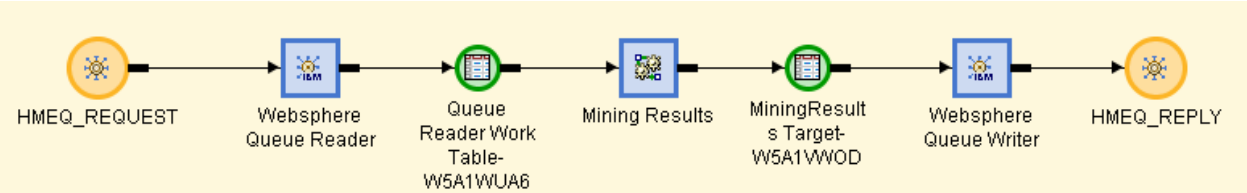

Figure 17. A SAS Data Integration Process That Reads a Request from One Message Queue and Writes Results to a Second Message Queue

In this example, messages are read from a specific location, an IBM Websphere MQ message queue called HMEQ_REQUEST. These messages contain a specific structure that includes a number of personal facts about the individual requesting the loan. These personal facts are input to a SAS Data Mining model through the Mining Results transformation, which performs a sophisticated mining analysis to determine that individual's creditworthiness. This mining model returns specific scores that are then posted back to another Websphere MQ message queue called HMEQ_REPLY. Another application can later read these replies to determine the next level of processing for that individual's request.

This process flow looks like any other flows in SAS Data Integration Server except that the source and target are message queues instead of being physical data locations like a SAS data set or a database table. This allows for the same logic and mining model to be used in near real-time environments, utilizing message queues.

HOW IT WORKS: READING ONE OR MORE MESSAGES

To directly equate a Web service use of this basic process flow to that involving message queues, we'd expect to process a single message from the request queue and generate a single reply to the reply queue. While this would work, it could be much less efficient than a similar flow that reads a large set of requests from fixed physical storage like a SAS data set or a database table. When we process a large set of requests, we get the benefits of traditional ETL processes that run in batch mode. In batch processing, large amounts of data can be processed by a single process or in parallel as when grid computing is utilized. Even when using only a single process, efficiency results from allowing a single process to start and remain running for the duration of time needed to process all incoming data. In this mode, the cost to start up and shut down the process is a very small part of the overall processing requirements.

In contrast, when a process is run for a single piece of data, the start up and shut down costs can become most costly and inhibit high throughput levels. If the turnaround time is reasonable, and a number of messages accumulate on the message queue, near-batch levels of throughput can be maintained utilizing message queues. The number of messages read during a single process flow can be tuned in the settings of the Message Queue Reader transformation (Figure 18).

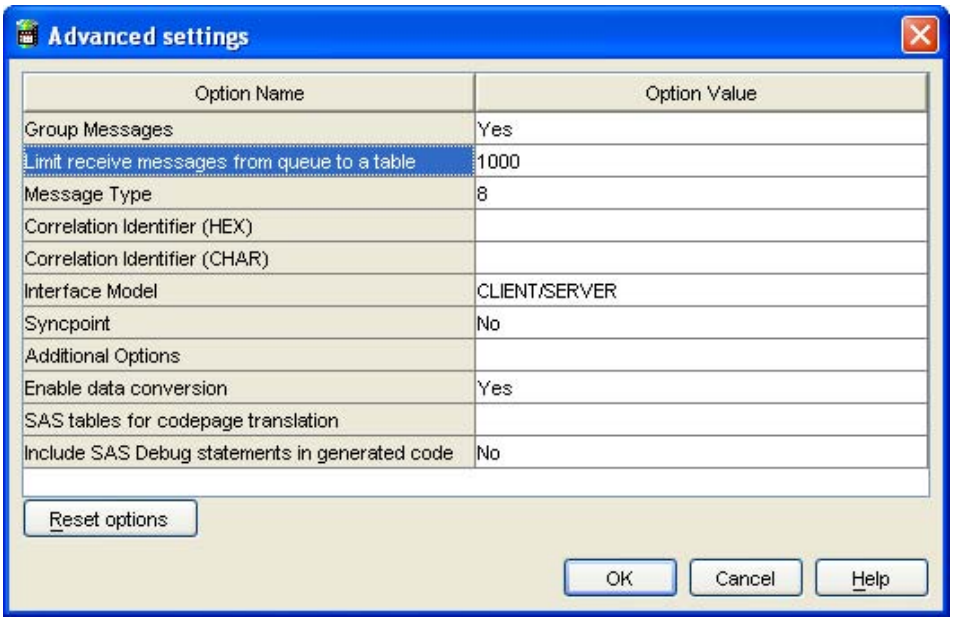

Figure 18. Setting the Number of Messages Read from the Queue in the Message Queue Reader Transformation

In Figure 18 we see the value for "Limit receive messages from queue to a table" is set to 1000 messages, which will enable any number of messages up to a maximum of 1000 to be read at one time. If more messages are present in the queue, the remainder will be left in the queue until the next read operation occurs. There is a trade-off here in that a very small number can result in quick turnaround for any given message. As the volume increases, overall load on the system processing messages can rise and cause a very slow system relative to the throughput.

The opposite effect can occur if the value is set very high and there are a large number of messages waiting to be processed. If there are 1000 messages waiting, and each one requires considerable processing time, it is likely that even the first completed reply isn't posted until all of those messages have been processed. Referring back to Figure 18, all messages have to pass through the Mining Results transformation before any replies are sent to the Reply message queue.

As a side benefit though, it is likely that the system will be efficiently processing those 1000 messages since they'll be processed in a batch-like mode. In addition, if the processing requires parallel or grid execution, having a large number of messages to process enables the parallel resources to be efficiently utilized.

HOW IT WORKS: CHOOSING WHEN TO READ MESSAGES

Another consideration for message queue processing involves the frequency and method that is used to start the process flow shown in Figure 17. A good place to start is determining whether there is a service level agreement in place that governs how long a message can wait before processing begins. In our example, we might know that the home equity loan requests need to be processed every ten minutes. This would mean that replies for most requests are returned to the individual making the inquiry within about 20 minutes. In this scenario, the traditional batch ETL process scheduling system would work well. Instead of the most traditional batch scenario of running a process a single time each night, the scheduling system could run the message queue process every ten minutes. As long as the overall process completes in the ten minute interval before the next process starts, this should allow all messages to be processed in a timely and efficient manner.

In Figure 19, the scheduling interface in SAS Management Console shows a ten minute interval between successive runs of the message queue process flow. The time event provided by Platform Computing's LSF Schedule will cause the process flow to be started at five minutes past each hour, every day, and at ten minute intervals following that.

Figure 19. Message Queue Scheduling Interface in SAS Management Console

HOW IT WORKS: ON-DEMAND PROCESSING WITH TRIGGERS

If fixed time intervals are too slow for the level of processing turnaround that we require, message triggers should be considered. A *trigger* is a mechanism for the message queuing system itself to launch the message queue process in SAS. It provides a broad range of configuration settings for the launch process.

Triggering can be complicated to configure, so you should refer to appropriate documentation for the message queuing system you're using. The following discusses triggering basics for IBM Websphere MQ. Triggering on Websphere MQ requires an additional queue called an Initiation queue (Figure 20). This is the queue that will actually launch the SAS process when a suitable message is placed there.

Queue name	Queue type	Definition type	Open input count	Open output count	Current queue depth
DEADLQ	Local	Predefined	0	0	0
HMEQ_INITIATION	Local	Predefined	0	0	0
HMEQ_REPLY	Local	Predefined	0	0	0
HMEQ_REQUEST	Local	Predefined	0	0	0
REPLY	Local	Predefined	0	0	0
REQUEST	Local	Predefined	0	0	0
TRIGGERQ	Local	Predefined	0	0	0

Figure 20. WebSphere MQ Showing Queues - the HMEQ_INITIATION Queue Causes the Trigger to Fire

Another added requirement for Websphere MQ triggering is a process definition (Figure 21). This is the actual process that is launched by Websphere MQ when a trigger is executed. Having a registered process definition is a prerequisite for starting a SAS process via a trigger.

Process name	App type	Application ID
RUNSAS	Windows NT	c:\sas\mq\runsas.bat
RUNSAS_HMEQ	Windows NT	c:\sas\mq\runsas_HMEQ.bat

Figure 21. WebSphere MQ Showing Registered Process Definitions

Finally, a listener is defined to handle the connection between the incoming Initiation message and the defined SAS process (Figure 22).

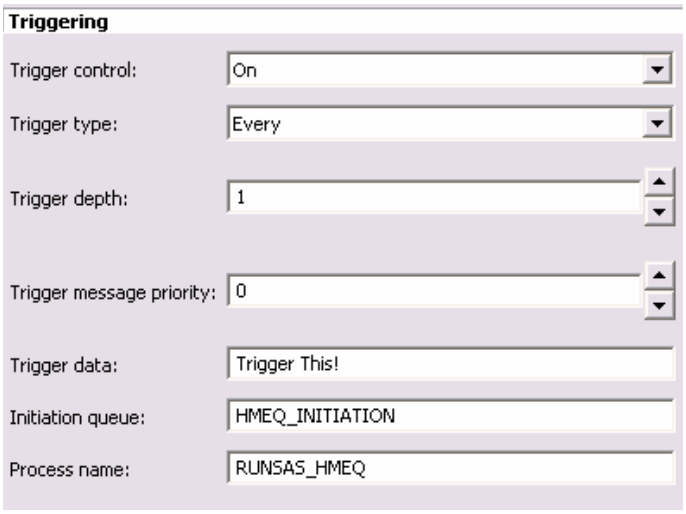

Figure 22. Setting Trigger Parameters in WebSphere MQ

A wide range of configuration options are available to customize triggering for your specific needs. In the example shown in Figure 22, triggering is On, and it is set to launch the process named RUNSAS_HMEQ for every single message that arrives. These settings work well for quick turnaround for messages.

Using an immediate trigger with a relatively large limit on the number of messages to read during each process execution can result in quick turnaround and good system utilization. In the example above, up to 1000 messages can be processed whenever anything is detected in the message queue. In addition, because the Initiation queue can ensure that no more than one processing job is triggered at a time, this avoids concerns in the scheduled mode that the next process could start before the previous one completes.

This type of configuration also effectively implements a load-management system that isn't possible with Web service implementations. In Web services, a large number of requests could be processed concurrently, but this could be beyond the capabilities of the underlying hardware to complete effectively. Triggering from message queues can easily limit the number of concurrent sessions, the frequency of checking for incoming messages, and the volume of messages to process during each interval.

In summary, we've discussed how message queues can be used to provide very timely response to data or processing needs. As shown above, message queues can be read and written in a straightforward manner, very similar to standard processing that is done for traditional physical tables like SAS data sets or database tables. Polling mechanisms like scheduled executions and triggers complete the management infrastructure to enable processing to be done in a managed and controlled environment. You should keep message queues in mind when timely response is needed and when request volume can be variable or potentially very large. Message queue processing in your data integration environment can bring the best of batch ETL throughput and efficiency to your active warehousing environment.

CONCLUSION

This paper covered three mechanisms for dealing with growing information volumes and increased use of this information. Data volumes continue to increase, while the time we're willing to wait for this information to become ready for analysis regularly shrinks. Active warehousing makes it possible to incorporate more recent information much more quickly than traditional, batch-only ETL processing allows. The common thread of the three approaches covered in this paper is to focus on the specific information needed so the processing of that information can be done as quickly as possible.

Change data capture allows you to work with traditional sources like relational databases, but you only need to handle data that has been changed. Instead of processing a full extract from the database, we only have to worry about the much smaller set of new and updated records. Downstream processing of this limited set of data can be completed much more quickly than working on full extracts, so we can present the most up-to-date picture much more quickly than in the past.

Web services provide an extremely dynamic intercommunication environment so that up-to-the-second transactions can be incorporated into the SAS analysis environment. This allows nearly instantaneous use of the newest transactional information in your organization, so that analysis results are always based on the most current data. While the overhead associated with each transaction is higher than in batch, the arrival of these transactional records is spread across the 24-hour day so that factor can have a minimal impact in many cases.

Message queues provide a middle ground of very quick turnaround when a message-oriented middleware system is used. If this already forms a communications hub in your organization, receiving periodic or nearly instantaneous updates mean that very current data is always available for analysis.

The SAS Data Integration Server environment can help you leverage each of these approaches. As shown in the examples in this paper, these are straightforward enhancements to your existing batch ETL logic. Sources like change data capture databases, XML streams to Web services, or messages from a message queuing system are able to be read right into your data integration environment. You'll be able to implement your active warehousing system quickly, so you should start thinking now about the ways to get maximal benefit from your completely current data.

REFERENCES

Wikipedia. 2007. Service-Oriented Architecture. Wikipedia, The Free Encyclopedia. (Accessed February 27, 2006). Available at en.wikipedia.org/wiki/Service-oriented_architecture.

RECOMMENDED READING

Ankorion, Itamar. "Change Data Capture – Efficient ETL for Real-Time BI". *DMReview*, January 2005. Available at www.dmreview.com/article_sub.cfm?articleId=1016326.

Wikipedia. 2006. Change Data Capture. Wikipedia, The Free Encyclopedia, Accessed February 27, 2007. Available at en.wikipedia.org/wiki/Change_data_capture.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Eric Hunley
Product Management
SAS Institute Inc.
Cary, NC 27513
eric.hunley@sas.com

Gary Mehler
Research and Development
SAS Institute Inc.
Cary, NC 27513
gary.mehler@sas.com

Nancy Rausch
Research and Development
SAS Institute Inc.
Cary, NC 27513
nancy.rausch@sas.com