



[SYMBIANT
TECHNOLOGIES, INC.]

[N-TIER ARCHITECTURE SERVICES]

[Technical Whitepaper]



Date	Revision	Description	Modified By	Audited By
10/6/2006	1.0	Initial Internal Release	Scott Frappier	Scott Frappier
10/9/2006	1.1	Changes to formatting and grammar. Added click and continue section.	Scott Frappier	Warren Young, David Hutchinson, Sean Solberg
10/23/2006	1.2	Style changes	Scott Frappier	Sean Solberg



Contents

Introduction 4

What is a 3-tier Architecture? 4

 Presentation Layer 5

 Business Logic Layer 5

 Data Management Layer 5

How do the nTier Architecture Services work? 8

How do I expand the functionality as a Partner/Customer? 11

Conclusion 12

Introduction

The Symbiant nTier Architecture Services provide a platform for the creation of 3-tier processes on the Microsoft Dynamics-NAV 3.01 to the most current version at the release of this whitepaper, Microsoft Dynamics-NAV 4.0 SP2. This technology enables Dynamics-NAV partners to use 3-tier processes to enhance the user experience. There is a basic set of transaction sets that comes with the product, but the solution has been designed for partners and customers to expand the functionality to their specific needs.

Symbiant Technologies is proud to present this technology to the market, and hopes that it will help your customers as it has helped ours.

What is a 3-tier Architecture?

SAP, Dynamics-AX, Peoplesoft, and other applications within the ERP industry typically use a software architecture usually referred to as a “3-tier architecture.” The 3-tier architecture concept is not new, as it was used on mainframes and has now been brought to the general PC platform.

The 3-tier architecture is composed of three layers:

- Presentation Layer
- Business Logic Layer
- Data Management Layer

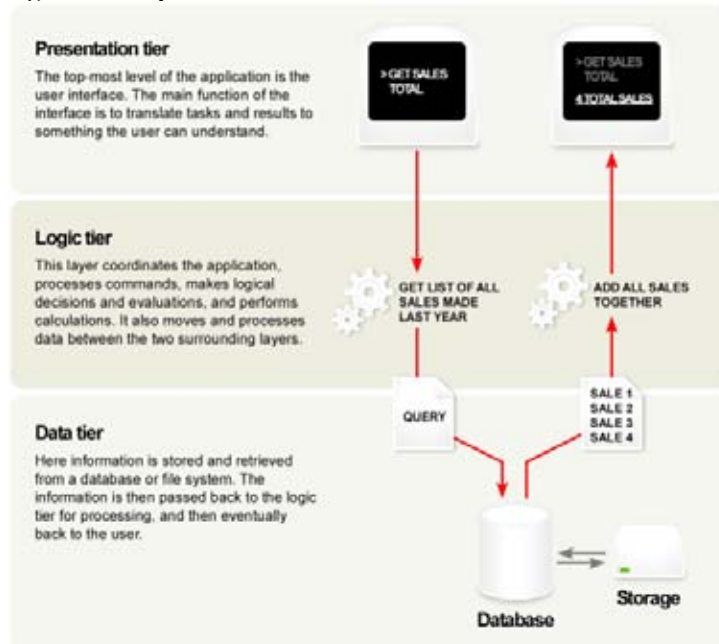


Figure 1 – A basic 3-tier application

Presentation Layer

The presentation layer is considered the top layer, in which the user interacts with a system. Screen layout, navigation, language integration and other user-based elements are controlled in this layer.

Business Logic Layer

The business logic layer is the driver of the entire system. This layer handles the requests and commands that have been sent from the presentation layer. It then takes those requests and runs business logic against them to provide answers to the presentation and data management layers.

Data Management Layer

This layer stores and retrieves all of the information within a database or file system.

How has Symbiant implemented a 3-tier architecture on a 2-tier client?

Microsoft Dynamics-NAV is technically a 2-tier, “thick” client application. The Presentation and the Business Logic Layers are encapsulated within the Dynamics-NAV executables, and the Data Management Layer is handled on the SQL Server or Native database environment. When a request occurs from the Dynamics-NAV client executable, the following occurs:

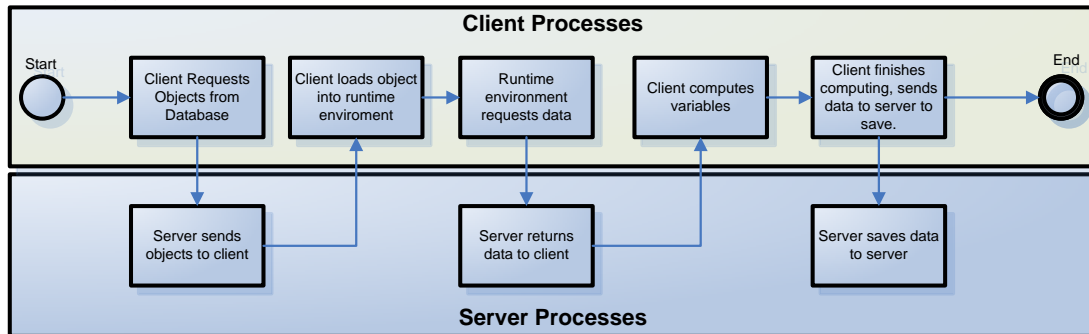


Figure 2- Basic request pattern within the Dynamics-NAV system

In this type of environment, the client is impeded by the business logic layer, as it must handle the processing of data that was sent from the server. The bottleneck that many companies run into is the fact that the slowest computer on the network will potentially slow down all other processes in the system. This occurs because the business logic layer requires CPU horsepower to complete the tasks that have been loaded, and a slower CPU will perform these operations at a slower pace.

The opposite could occur if the business logic layer was addressed in the database management system. The server would then become the potential bottleneck because of all of the processing that it may have to do in order to keep up with the requests coming from the clients. Luckily, we do not run into this situation with the Microsoft Dynamics-NAV product since we are running the business logic on the client.

While the 2-tier technology works great in small to midsize installations (1-50 concurrent users), higher end installations require more technologies to handle the concurrency issues that occur from multiple transactions and resource requests within a system. This is where the Dynamics-NAV application starts to run into issues, and why other solutions have been architected with a 3-tier environment.

Because of the way in which Dynamics-NAV is designed, many of the tables are in an *Entry No.* design. The G/L Entry, Item Ledger Entry, and Value Entries are perfect examples of tables within Dynamics-NAV that are designed in this fashion. The concept is simple; every entry that is inserted into these tables must be unique and serialized. This means that there should be no gaps between numbers when inserted into the tables. While this concept provides a very solid platform for auditing and transactional integrity, it also introduces new problems that are hard to overcome within the classical 2-tier architecture.

When a user posts a sales order, there are multiple transactions that occur within the given process. One of these steps (at the beginning of the transaction) is to lock the G/L Entry table and find the last G/L Entry No. During this process, the G/L Entry table has a lock on the last record in the database. If another user tries to access the last record (another posting in the system), they will be blocked until the lock is released. In the example above, the lock will not be released until the transaction completes.

There are a couple of ways to get around this type of issue. One option would be to re-design the process so that it has multiple COMMITs throughout. The design would have to encompass the fact that there could be deadlocks/errors throughout each portion of the COMMIT, and some type of programmatic rollback would need to be implemented.

Another option would be to hand off the business logic to a separate process so that the user is not encumbered by the transaction that they are trying to conduct within the database. The transaction would be queued for processing by another server, and if the process fails it would notify the user.

This is the path that Symbiant decided to pursue.

Microsoft Dynamics-NAV comes with the capability of running Navision Application Servers (NAS) that are basically Dynamics-NAV clients without the GUI capabilities. This type of client starts as a service, and allows processes to be offloaded to it for processing. Microsoft has used this executable for ADCS, Employee Portal, and other extended functional areas that require constant interaction with the database environment.

Since the NAS is basically an application/batch server, its technology could be leveraged to offload processes that client PC's have to do. Some of the benefits of this are:

- 1.) A fast pipe between the database server and application server can be created (1 GBps).
- 2.) A fast machine can be introduced on the network that can process each transaction.
- 3.) Processes could be queued serially, which means that posting processes that conflict with each other can be eliminated.
- 4.) Processes could be offloaded to multiple application servers to create a load balancing environment.

Since the benefits seemed to address many issues that are faced in large scale environments, a concept was created in which a NAS server would post sales documents in one of our customer environments. After implementation of the technology, the users were amazed. Instead of waiting 10-20 seconds for a posting to complete, the posting apparently would complete in less than a second. Since that initial concept, multiple areas of the application have been expanded so that the user experience is increased to a level in which users are not hindered by the ERP system.

The Symbiant nTier Architecture Services was born.

How do the nTier Architecture Services work?

The nTier Architecture Services work off of one basic principle.

Do not make the user wait for a transaction to complete before allowing the user to address another task within the system.

Most ERP systems work in a “click and wait” mode, in which the user tries to do an action within the system, the system computes this transaction (could take 5 seconds or 10 minutes) and the results are then returned to the user. Sometimes, this classical view of a transactional system can be changed so that the user does not need to wait for a process to complete before going to their next process in the system.

The nTier Architecture Services are a set of objects that exist in Dynamics-NAV that offload the processing from the client executable, to a dedicated application server to create a 3-tier environment. When a user clicks a button on the Dynamics-NAV client, the business logic processing is offloaded from the client machine to the nTier Application Server (nTAS). The nTAS then processes the transaction within the system. The client is only in charge of the validity tests that need to occur before submitting the transaction to the nTAS Server.

This type of architecture is what we have dubbed the “click and continue” architecture. The ERP system is not the bottleneck for the user, but allows the user to continue onto other tasks within the database. This saves time, allows faster responses to inquiries, and changes the age-old archaic mentality of “click and wait.”

The basic workflow of the nTier Architecture Services is as follows:

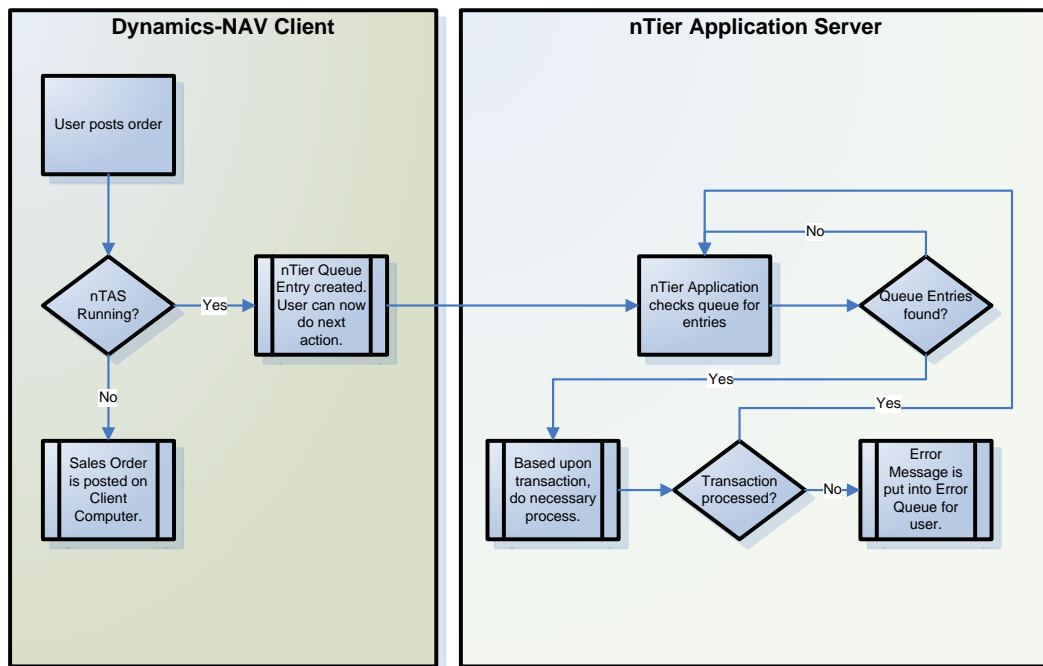


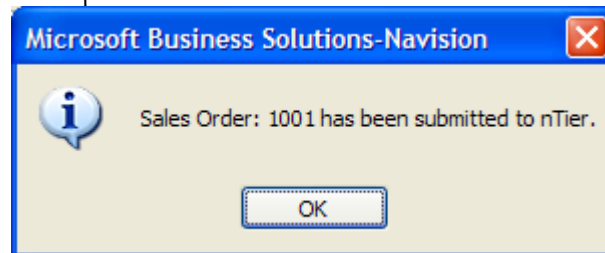
Figure 3 – nTier Architecture Services

An example of how a sales order posting occurs within an nTier enabled database is as follows:

- 1.) The order is entered by the user based upon the requirements given by the customer.
- 2.) The user releases the order, which in this case, runs the business logic on the client computer.
- 3.) The user then goes to a function to print out a document for the customer and hands the document to the customer (this could be a Performa invoice, shipment ticket, et cetera).
- 4.) After printing, the user Posts.
- 5.) The Post screen is returned; the user then selects the Ship radio button and clicks on OK.



- 6.) Almost immediately after the user presses the OK button, a confirmation dialog appears stating that the process has been submitted to nTier.



That's it! The user does not wait for the transaction to complete, and goes onto the next customer to help them with their request. This enables the user to give the best experience to the customer in this particular process, since they are not hindered by the ERP system. The handling of errors that occurred during a posting is either handled as an end of day function, or something that can be examined throughout the business day.

The concept above can be applied to almost all processes within the Dynamics-NAV application. Out of the box, the following functions will submit to the nTAS for processing:

- Sales Documents
 - Orders
 - Invoices
 - Credit Memos
 - Return Orders
- Purchase Documents
 - Orders
 - Invoices



- Credit Memos
- Return Orders
- Transfer Orders
- Warehouse Documents
 - Receipts
 - Shipments
- General Journals
- Item Journals
- Deposits
- Job Scheduler Jobs
 - Adjust Cost
 - Update Sales Analysis Views
 - Update Analysis Views

The design encompasses the following objects on top of the base objects modified:

Type	ID	Name	Comments
Table	37009001	nTier Parameter Setup Line	This table has the setups for which processes should be handled by a specific nTier parameter
Table	37009002	nTier Queue Log	This table contains all of the transactions and errors that have occurred during an nTier process
Table	37009003	nTier Queue	The nTier Queue contains entries that have yet to be processed by the nTAS
Table	37009004	nTier Posting Setup	This setup defined what documents can/cannot be transacted by the nTAS's
Table	37009049	Date_Time	This table provides the servers date/time stamp for transactions. It is a view that exists in SQL Server that needs to be created.
Form	37009001	nTier Queue	Form to view active queue entries.
Form	37009002	nTier Parameter Setup	Form to setup basic nTier Parameters
Form	37009003	nTier Parameter Setup Line	Form to setup basic operations for a nTier Parameter
Form	37009004	nTier Posting Setup	
Report	37009001	nTier Purch. Document Errors	Shows purchase document failures
Report	37009002	nTier Whse. Ship. Doc. Errors	Shows warehouse shipment document failures



Report	37009003	nTier Whse. Rcpt. Doc. Errors	Shows warehouse receipt document failures
Report	37009004	nTier Document Errors	Shows all document failures
Codeunit	37009001	nTier Queue Management	This codeunit manages the processing of the queue from the nTAS
Codeunit	37009002	nTier Job Scheduler Management	This codeunit is used when Job Scheduler automation is enabled for a nTAS
Codeunit	37009003	nTier Business Logic Mgt.	This is the heart of the application...it basically automates the tasks for posting, updates log entries with failure reasons, and submits entries to the queue
Codeunit	37009010	Job - Inventory Adjustment	Runs the Adjust Cost batch job via job scheduler
Codeunit	37009011	Job - Update Sales Analysis	Updates Sales Analysis views via job scheduler
Codeunit	37009012	Job - Update Analysis Views	Updates Analysis Views via Job Scheduler
Codeunit	37009019	Date_Time	Centralized function for SQL Server Date Time

How do I expand the functionality as a Partner/Customer?

Expanding the nTier Architecture Services involves the following:

- 1.) A queue entry must be created. Examples on how this is handled exist in codeunit 37009003 – nTier Business Logic Management. Look for the *SubmitSalesDocument* example. Make sure that this process is called by the client application when doing the action.
- 2.) After the queue entry has been created, the nTAS will process the record. In order to know “what to do” with that record, or if it requires a specific record to be passed, codeunit 37009001 must be modified to add hooks to the *ProcessQueue* and *FailOriginalRecordIfRequired* functions. These functions both call codeunit 37009003 when transacting within the system.
- 3.) Modify the front-end experience for the user. If you need to have documents printed, we recommend that you print them *before* submitted to the nTAS, as printer queue management is not something that the Navision Application Server can effectively handle.

The flexibility that this provides a partner is the capability to quickly and easily implement new solutions to help offset potential system bottlenecks, and increase the overall performance of the system.

Conclusion

The Symbiant nTier Architecture Services provide a platform to help a partner/customer develop a highly scalable, concurrent environment. Automating jobs, removing business logic processing from the client, and reducing user frustration allow for the solution to easily scale to new heights within the Dynamics-NAV environments.

Currently, Symbiant Technologies has three customers that are using this technology. One customer has 450 concurrent users and is using the pre-cursor to the nTier technology that is available today. The other two are running in 150 concurrent user environments, and have actively participated in the feature-set that has been provided.

The technology and concepts have also been considered in terms of future upgrades. Dynamics-NAV 5.0 will introduce a 3-tier environment that will allow us to move this processing to the middle-tier more effectively. While load balancing and user experience have not been the focus of the Dynamics-NAV 5.0 design team, Symbiant will have a solution to help with these needs and will continue to enhance the product as more features and technologies become available.

All recommended updates to the product will be considered, as partner/customer feedback is critical. Please feel free to submit new requests for functionality to the product, and hopefully it will be added so that customizations are not required against the base package to make it easier for future upgrades.

If you run into environments that have a high-degree of complexity or transactional volume, please feel free to contact us regarding this product offering. It could potentially save your company a lot of time, and alleviate performance frustrations so that partner/customer relationships can be enhanced!