Wonderware® FactorySuite™ InTrack Deployment Guide

May 2002

Wonderware Corporation

Contents

Chapter 1
InTrack Overview 1-1

Manufacturing Management 1-2

InTrack's Role 1-3

Simple Discrete Manufacturing 1-3
Electronic Assembly 1-4
Consumer Packaged Goods 1-4
Web and Roll Processing 1-4
Hybrid Manufacturing 1-5

Chapter 2
System Architecture 2-1

Network Overview 2-2

Chapter 3
System Requirements 3-1

General Requirements and Recommendations 3-2

Multiple Processors 3-2
Disk Speed 3-2
Networking 3-2
Client Performance 3-2
Recommended Accessories 3-2

Database Requirement Calculations 3-7

Memory Requirements 3-7

Chapter 4
Installation Instructions, InTrack 7.11 4-1

Automatic Data Migration 4-1

Chapter 5
Usage Guidelines 5-1

Functional Guidelines 5-2

Project Management Guidelines 5-3

Coding Guidelines 5-4

Using Visual Basic vs. InTouch 5-5

Chapter 6
User Characteristics 6-1

Process Control Engineers 6-2
Production Supervisors 6-2
Production Operators 6-2
Production Support Personnel 6-3
Chemist, Cooks, Scientist 6-3

Chapter 7
FactorySuite Component Integration 7-1

InSQL 7-2
SPCPro 7-3
InControl 7-3

Chapter 8
Elements of InTrack 8-1

The Process Model 8-2
OLE Automation 8-3
Structural Objects 8-4
Calendar Object 8-4
Customer Specification Object 8-4
Dataset Template Object 8-5
Disposition Code Object 8-5
Location Object 8-6
Machine Object 8-6
Machine Type Object 8-7
Material Object 8-7
Operation Object 8-8
Route Object 8-8
Security Group Object 8-9
Setpoint Template Object 8-9
User Object 8-9
User Certification Object 8-10
Work Instructions Object 8-10
User Defined Tables 8-10
Activity Objects 8-11
Creating Lots 8-11
Adjusting Lots 8-12
Starting Lots 8-13
Consuming Lots 8-14
Completing Lots 8-17

Closing Lots ... 8-19
Moving Lots ... 8-20
Merging and Splitting Lots ... 8-20
Shipping Lots ... 8-20
Lot Status ... 8-21
Data Collection Using Dataset Templates, User Variables, and User Tables ... 8-21
Adding Comments ... 8-23
Monitoring Machine Downtime ... 8-24
Database Locking ... 8-25

Query Objects ... 8-27
Pre-built Queries ... 8-27
General Query ... 8-27

Chapter 9
Reporting ... 9-1

Table Relationships ... 9-2

InTrack Views ... 9-2

Using a Report Server ... 9-3

Chapter 10
Purging and Archiving Data ... 10-1

Purging ... 10-2
Purge Options ... 10-2
Purging Unshipped Lots ... 10-2

Archiving ... 10-3

Database Concerns ... 10-4
Performance ... 10-4
Database Size ... 10-5

Chapter 11
Terminal Server Architecture ... 11-1

Special Notes on Terminal Service Enviornments ... 11-2
Archiving must be performed only on the TSE server ... 11-2
Installations must be performed using Add/Remove Programs ... 11-2
Different log file names are required when running the trace utility on multiple nodes simultaneously ... 11-2

Sizing the Termial Server ... 11-2
Test Hardware and Software ... 11-2
Client Startup ... 11-3
Performance Measurements ... 11-4
Transactional Measurements ... 11-9
Conclusion ... 11-12

Chapter 12
Connecting to an Oracle Database 12-1

Glossary G-1

CHAPTER 1

InTrack Overview

InTrack is the tracking component of FactorySuite, a fully integrated suite of software for industrial automation.

A powerful Manufacturing Management Information (MMI) system, FactorySuite provides a full package of automation tools:

- **Visualization (InTouch)**
- **Factory Database (IndustrialSQL Server)**
- **Resource and Work-In-Process (WIP) Tracking (InTrack)**
- **Batch Management (InBatch)**
- **Windows NT-based Control (InControl)**
- **Internet/Intranet Visualization (FactorySuite Web Server)**
- **Connectivity**

The components can be used in different combinations to collect and manage production information, create graphic visualizations of the production environment, analyze plant process, track and improve production operations; control machines and processes, and manage batches.

FactorySuite can increase productivity in continuous, discrete and batch manufacturing environments. It operates effectively at every level of the enterprise, from factory control to interface with business systems.

Four versions of the suite are available for developing applications in different manufacturing environments.

FactorySuite 2000 runs on the Microsoft Windows NT 4.0 (SP5) operating system.

InTouch and IndustrialSQL Server clients also run on Windows 95/98 SE.

Contents

- Manufacturing Management, 2
- InTrack's Role, 3

Manufacturing Management

Wonderware's FactorySuite is designed for effective automation of the factory floor. The software manages production, materials handling and process control with a suite of tools designed for runtime execution and full collection of history data.

Three key areas of manufacturing management are identified in a "Triangle of Manufacturing" concept which applies to many industries:

- **Production Execution and History**
  Production execution defines the resources used to produce a product, and includes the plant operations, machines, operators, documents such as specifications and other elements required for plant operation and production. Production History is the storage of information about these resources for evaluation and future planning.
- **Material Execution and History**
  Material execution defines the products or materials that are combined to produce intermediate or final products. Material execution also involves the consumption or destruction of materials in the production process, such as the logs cut apart in a sawmill. Material history records the consumption, storage, movement and production of the materials.
- **Process Execution and History**
  Managing the process ensures that the correct parameters are used by the control systems. Process controls such as setpoints and configuration parameters are applied during the manufacturing process. The actual values and measurements during manufacturing are monitored throughout the process.

All three areas of information management are used by most industries. Depending on the industry, the three components may have greater or lesser weight in the total automation solution. For example, in the pharmaceutical industry, special emphasis on the monitoring of materials and final products may be required.

Failing to address all three areas may hinder the manufacturing management solution.

InTrack's Role

As the tracking component of FactorySuite, InTrack focuses on the management of data related to Production and Material execution and history collection. Used with the InSQL databases used to record and analyze ongoing production date, InTrack also plays an important role in Process management.

InTrack and InSQL are effective at solving problems together: InTrack provides the context information, which allows isolation of a large volume of historical process information stored by InSQL. In addition, the SPCPro and Alarms components can enhance the management of production information.

A few examples of how InTrack can be used in different industries are provided below.

Simple Discrete Manufacturing

Simple discrete manufacturing entails production using a Bill of Materials with two to four levels. Products are classified as configured-to-order, make-to-stock, or engineered-to-order. Consumption of materials is usually based on lots.

Usually in this domain, the Bill of Materials is downloaded to InTrack from an order entry system or an ERP system. In many cases, a custom BOM can be created for each order.

In some make-to-stock plants, the number of products is small and BOM information can be managed directly in InTrack's database using the modeling component, Model Maker. Manufacturing routes in these plants often are complex and products can move through multiple operations before completion.

Production and material information includes the machines used, the materials consumed, the operations performed and the operators qualified to perform them, off-line testing requirements, product customizations or downgrades, and customer specifications.

The InTrack database stores important information such as yield, operation queue time, in-process time, machine availability, lot movement and material consumption.

Process data is less of a factor in this domain. Process information may be needed to monitor environmental systems, furnaces, clean rooms and automatic testers. InTrack is effective in monitoring continuous assembly lines used for manufacturing discrete products. Generally, when time between operations is short, such as fast-moving assembly lines, InSQL can be used to record information easily and quickly, so the more time-intensive InTrack transactions may not be needed for ongoing process data management.

Electronic Assembly

Similar to discrete manufacturing, the assembly of electronics usually involves serialized parts and a higher level of detail from the tracking database.

Serialization can be accomplished though serialized lot quantities or by using a separate lot for each unit (lot size of one).

If units move through the route with lot quantities greater than one, serialization provides an efficient means of tracking transactions. Executing serialized transactions individually may not complicate database management and slow the system.

Many electronic assembly facilities produce a large number of parts and require careful attention to individual models and specific route transactions. In high volume systems, the database server should not be burdened with extensive queries or other transactions that slow system performance.

Many electronics applications use barcode readers to identify parts. Due to the advanced level of automation and high transaction rates required for the manufacture of electronics, multiple InTrack servers may be used at the same site.

Consumer Packaged Goods

Consumer packaged goods usually involve a large number of input materials, as well as abundant variations of end products and byproducts.

Since the same basic product may required hundreds of packaging variations, the number of bill of materials to be maintained is significant.

In most cases, margins are slim and the collection of process data is essential to maintain a high level of productivity. Data collection may be used to reduce waste, or to ensure that the products meet governmental regulations..

Process data is usually collected both by automated measuring devices and by operators recording information manually. InTrack can be combined with either a data historian or a statistical process control (SPC) package. Measurement data must be linked to the lot produced. Ongoing collection of data is used to create production genealogy information that is essential for determining product quarantines and avoiding product recalls.

Web and Roll Processing

Web processing focuses on material handling and the conversion of one product to another. In some cases, there is no material consumption. The system must monitor machine usage and machine conditions, since machine time is critical.

As material is cut or divided into other smaller components, genealogy tracking from parent lot to intermediate lot to final product is required. The database must store information on high volumes of material output, with possible unit conversion. Although routing is usually straightforward, reporting can be complex because of multiple intermediate materials.

Although processing rates can be fast, tracking of web and roll products is often slow paced. One database server can accommodate a large number of processing machines.

Hybrid Manufacturing

In a hybrid manufacturing facility, several different types of production occur at the same time. Discrete, continuous and/or batch processing might be used in separate areas of the facility.

Integration among the different manufacturing areas is essential. Each area may require unique controls or data management, which could entail the use of different tools throughout the plant. The transfer of controls among the different processing areas can be managed effectively using different components of FactorySuite. In a hybrid manufacturing plant, it is possible to have all components of the FactorySuite in operation.

For additional considerations on integrating InTrack with other FactorySuite components, refer to Chapter 5, Practical Application.

CHAPTER 2

System Architecture

Like all FactorySuite components, InTrack is designed to operate on personal computer networks using standard Microsoft Windows software and standard network technology.

Operations are performed using one or more servers to store database tables and record database transactions. User applications reside on client computers and use data retrieved from the server.

Additional servers can be added to enable multiple server processing (for large operations), to accommodate additional FactorySuite components, such as InSQL, that perform additional analysis of transactional data, or for specialized functions such as reporting.

Both SQL Server and Oracle technology can be used for server and database management.

Because FactorySuite is designed to use standard server components, hardware and tools, users are afforded the flexibility of choosing from a wealth of hardware, software applications, connection devices and other peripherals. FactorySuite has consistently shown an ability to adapt fully and integrate well with evolving and emerging technologies.

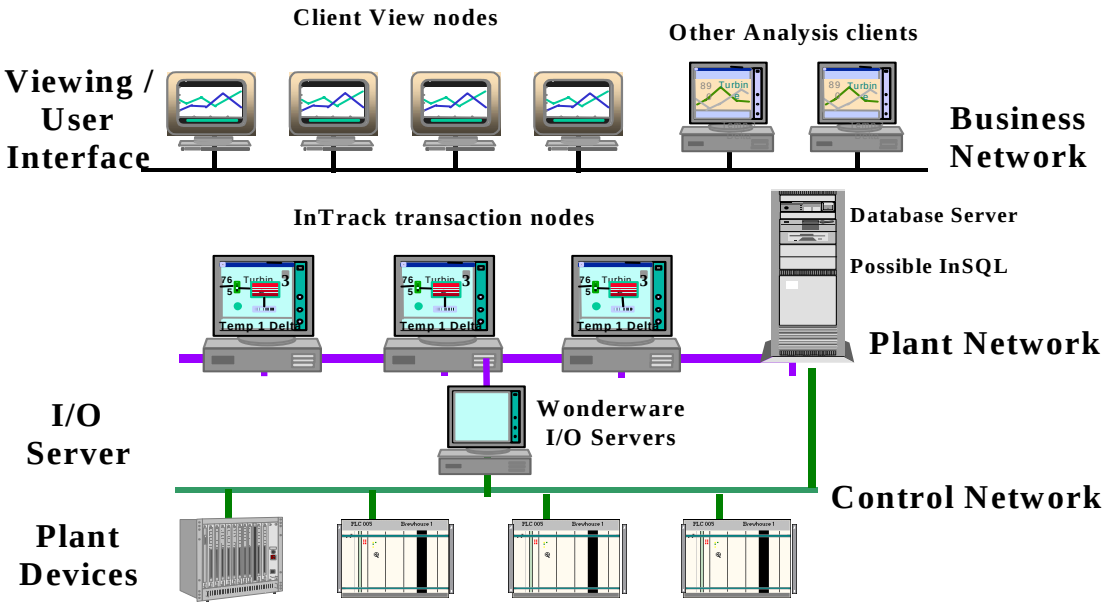
FactorySuite also accommodates connectivity with a wealth of external systems and terminal services. For example, two-way communications with commonly used factory PLC devices enables downloading and uploading of information directly from automated equipment on the factory floor.

Additional integration is possible with other systems, such as ERP.

Naturally, FactorySuite systems vary greatly because of the virtually infinite combinations of factors that impact different industries and their needs, including plant size, workload, level and degree of automation, and product type.

Network Overview

The following diagram illustrates the components and levels involved in a basic FactorySuite system:



CHAPTER 3

System Requirements

InTrack is designed to run on a client/server network. Before you can install and use InTrack or ModelMaker, your system must be equipped for Microsoft SQL Server or Oracle database management.

Currently, InTrack is supported on the following database platforms:

- Microsoft SQL Server 7.0
- Microsoft SQL Server 2000
- Oracle 8.0.5
- Oracle 8.1.6

This chapter outlines general system requirements for each of the database platforms.

Contents

- General Requirements and Recommendations, 2
- System Requirements — Microsoft SQL Server 7.0 Platform, 3
- System Requirements — Microsoft SQL Server 2000 Platform, 4
- System Requirements — Oracle 8.0.5 Platform, 5
- System Requirements — Oracle 8.1.6 Platform, 6
- Database Requirement Calculations, 7

General Requirements and Recommendations

The following requirements and recommendations apply to all platforms.

Multiple Processors

All InTrack platforms support the use of multiple processors. Using multiple processors can effectively enhance system performance.

- **SQL Server 7.0** can use up to four processors. Additional processor support is available with SQL Server 7.0 Enterprise Edition.
- **SQL Server 2000** can use up to four processors. Additional processor support is available with SQL Server 2000 Enterprise Edition.
- **Oracle 8.0.5** and **Oracle 8.1.6** supports the use of multiple processors.

Disk Speed

Disk speed is critical for terminal server performance. Small computer system interface (SCSI) disk drives, especially devices compatible with Fast SCSI and SCSI-2, have significantly better throughput than other types of drives.

Networking

The system should be equipped with 10/100 Mbps network adapter card (TCP/IP protocol).

A high-performance network adapter is recommended. Using multiple adapters can significantly increase network throughput.

Client Performance

Depending on the platform, InTrack client machines operate with either Windows NT or Windows 2000 Professional software. Because clients are established to perform different tasks, requirements can vary significantly for individual machines.

Additional memory or disk space may be required if:

- The InTrack client will be used to process more than 1 transaction every 5 seconds.
- Queries will be required to return more than 200 records.
- Complex graphics are used for applications.
- The client runs additional support applications, such as Crystal Reports, WWLogger, and SPCPro

A dual processor will not improve the performance of a client machine

Recommended Accessories

The following accessories are recommended for all server and client machines:

- Super VGA Monitor
- CD-Rom Drive
- Microsoft Mouse or equivalent
- UPS (Uninterrupted Power Supply).

System Requirements — Microsoft SQL Server 7.0 Platform

Server

Hardware

The minimum hardware requirements defined by the database manufacturer are listed below. Hardware recommendations for a typical InTrack installation for this server also are shown. For additional information on calculating the storage and performance requirements, refer to the next section, Database Requirement Calculations.

	Minimum	***Typical***
Pentium Processor	166 MHz	450 MHz Pentium III
RAM	32 MB	256-512 MB
Storage Space	64 MB	2-3 GB

Additional memory may be required, depending on individual system requirements. Hard disk requirements vary significantly based on individual network configurations and additional software installed.

Software

Minimum software requirements for SQL Server 7.0 are:

SQL Server 7.0 Enterprise Edition or Standard Edition

Microsoft Windows NT Server version 4.0 or Windows NT Server 4.0 Enterprise Edition, with Service Pack 5 or later.

Microsoft Internet Explorer 4.01.

NOTE: Client Access Licenses are required for all SQL Server operations.

Client

Hardware

Hardware requirements for effective clients are:

	Minimum	***Typical***
Pentium Processor	166 MHz	333 MHz Pentium III
RAM	32 MB	128 MB
Storage Space	64 MB	500 MB

Software

Client machines should include the following software:

Microsoft Windows NT version 4.0 or Microsoft Windows Professional 2000

SQL Server client software and utilities

Network connection

Optional InTouch, Microsoft Office, Crystal Reports and associated programs needed to perform client tasks

System Requirements — Microsoft SQL Server 2000 Platform

Server

Hardware

The minimum hardware requirements defined by the database manufacturer are listed below. Hardware recommendations for a typical InTrack installation for this server also are shown. For additional information on calculating the storage and performance requirements, refer to the next section, Database Requirement Calculations.

	Minimum	***Typical***
Pentium Processor	166 MHz	450 MHz Pentium III
RAM	32 MB	128 MB
Storage Space	64 MB	2-3 GB

Additional memory may be required, depending on individual system requirements. Hard disk requirements vary significantly based on individual network configurations and additional software installed.

Software

Minimum software requirements for SQL Server 2000 are:

SQL Server 2000 Enterprise Edition or Standard Edition

Microsoft Windows NT version 4.0 or Windows NT Server 4.0 Enterprise Edition, with Service Pack 5 or later.

Microsoft Internet Explorer 4.01.

NOTE: Client Access Licenses are required for all SQL Server operations.

Client

Hardware

Hardware requirements for effective clients are:

	Minimum	***Typical***
Pentium Processor	166 MHz	333 MHz Pentium III
RAM	32 MB	128 MB
Storage Space	64 MB	500 MB

Software

Client machines should include the following software:

Microsoft Windows NT version 4.0 or Microsoft Windows Professional 2000

SQL Server client software and utilities

Network connection

Optional InTouch, Microsoft Office, Crystal Reports and associated programs needed to perform client tasks

System Requirements — Oracle 8.0.5 Platform

Server

Hardware

The minimum hardware requirements defined by the database manufacturer are listed below. Hardware recommendations for a typical InTrack installation for this server also are shown. For additional information on calculating the storage and performance requirements, refer to the next section, Database Requirement Calculations.

	Minimum	***Typical***
Pentium Processor	166 MHz	450 MHz Pentium III
RAM	48 MB	256-512 MB
Storage Space	64 MB	2-3 GB

Additional memory may be required, depending on individual system requirements. Hard disk requirements vary significantly based on individual network configurations and additional software installed.

Software

Minimum software requirements for Oracle 8.0.5 are:

Microsoft Windows NT Server version 4.0 or Windows NT Server 4.0 Enterprise Edition, with Service Pack 5 or later.

Microsoft Internet Explorer 4.01.

Client

Hardware

Hardware requirements for typical application users are:

	Minimum	***Typical***
Pentium Processor	166 MHz	333 MHz Pentium III
RAM	32 MB	128 MB
Storage Space	120 MB	500 MB

Software

Client machines should include the following software:

Microsoft Windows NT version 4.0 or Microsoft Windows Professional 2000

SQL Server client software and utilities

Network connection

Optional InTouch, Microsoft Office, Crystal Reports and associated programs needed to perform client tasks

System Requirements — Oracle 8.1.6 Platform

Server

Hardware

The minimum hardware requirements defined by the database manufacturer are listed below. Hardware recommendations for a typical InTrack installation for this server also are shown. Minimum requirements for NTFS systems (default block size of 2 KB) are:

	Minimum	***Typical***
Pentium Processor	166 MHz	166 MHz
RAM	96 MB	256-512 MB
Storage Space	1 GB	2-3 GB

Software

Minimum software requirements for Oracle 8.1.6 are:

Microsoft Windows NT Server version 4.0 or Windows NT Server 4.0 Enterprise Edition, with Service Pack 5 or later.

Microsoft Internet Explorer 4.01.

Client

Hardware

Hardware requirements for effective clients are:

	Minimum	***Typical***
Pentium Processor	166 MHz	333 MHz Pentium III
RAM	32 MB	128 MB
Storage Space	100 MB	500 MB

Software

Client machines should include the following software:

Microsoft Windows NT version 4.0 or Microsoft Windows Professional 2000

SQL Server client software and utilities

Network connection

Optional InTouch, Microsoft Office, Crystal Reports and associated programs needed to perform client tasks

Database Requirement Calculations

The following guidelines for calculation of database storage space on the server apply to SQL Server and Oracle systems.

The InTrack database server should be a dedicated machine responsible only for maintaining the database. It should not be used as a file server, mail server, domain controller, or a backup domain controller.

If the database server has a high activity level, use of a second server for reporting is recommended.

Depending on the activity level of the database server, one would be wise to replicate the data to a second server that will act as the reporting server.

Memory Requirements

Since the server is only running the database software, memory should be based on the number of database transactions processed.

The server machine should not be used to support extensive graphics.

NOTE: Never run a screen saver (other than a blank screen) on the server.

Use the following calculations to estimate the number of transactions processed in the LotBaseLog. Typically the largest table in the database, LotBaseLog will provide a good general indication of the memory requirements.

Calculate the daily subtotals for the number of lots processed daily.

+ Transactions to Process Lots at each Step:		
Number of lots created per day	1,000	
Average number of steps per lot	× 8	
If start and complete	× 2	
Sub Total 1	16,000	
+ Transactions to Process Collections:		
Number of lots created per day	1000	
Average number of collections per lot	× 4	
Sub Total 2	4000	
+ Transactions to Process Consumption of Materials:		
Number of lots created per day	1000	
Average number of consumes per lot	× 12	
Sub Total 3	12000	
= Grand Total	32000	transactions per day
÷ Seconds in a day (8 hrs x 3600)	28800	one production shift
TRANSACTION RATE =	**1.11**	**transactions / second**

NOTE: This is not the actual number of transactions that SQL Server or Oracle will process. The database will have to process 5–15 times this many transactions.

If the calculated transaction rate is less than 2:

Any server grade PC with a single processor and 128 MB are sufficient.

If the calculated transaction rate is between 2 and 10:

Calculated rates between 2 and 10 will require a DBA to tune the database and properly allocate memory. Additional memory is recommended.

If the calculated transaction rate is greater than 10:

A rate above 10 will require a powerful database server dedicated to collecting data, plus a second reporting server to handle report queries.

Carefully review the number of queries to be issued in any high transaction environment.

Memory requirements will range between 256 MB and 2 GB. Remember that more memory is the easiest and typically best way to improve performance.

If the transaction rate exceeds 20, a redesign of the system to reduce transactions will be necessary.

Performance also can be improved by increasing processor speeds by adding dual processors, adding a second drive controller to handle the transaction log, or using additional database tuning options.

CHAPTER 4

Installation Instructions, InTrack 7.11

IMPORTANT: Install InTouch 7.11 before proceeding with InTrack installation.

After you have updated your InTouch system, you are ready to install InTrack Release 7.11:

1. Insert the CD into your CD drive. The setup.exe program will start automatically.
2. The InstallShield program will prompt you for verification. Click Next to continue.
3. The License Agreement window will appear. Click Yes to continue.
4. The installation program will replace existing files.
5. Continue to click Next at each prompt.
6. At the last prompt, click Finish.

The revised files will be copied to the appropriate InTrack directories and registered accordingly.

You will be prompted to reboot your system after the installation of common components has been completed.

Automatic Data Migration

When you start ModelMaker the first time after upgrading, a system display will prompt you to migrate your data. You MUST choose to migrate your data in order to use the new version of InTrack.

Your current database will be migrated and updated to accommodate the expanded features of Release 7.11.

CHAPTER 5

Usage Guidelines

Before setting up your InTrack system, refer to the important usage guidelines on the following pages:

- **Functional Guidelines**
  Basic considerations about your system set-up
- **Project Management Guidelines**
  Considerations about implementing your InTrack system
- **Coding Guidelines**
  Notes on coding conventions
- **Using Visual Basic**
  Comparison of Visual Basic and InTouch as the InTrack OLE container

Contents

- Functional Guidelines, 2
- Project Management Guidelines, 3
- Coding Guidelines, 4
- Using Visual Basic vs. InTouch, 5

Functional Guidelines

Database Connections

Only one database connection can be established per node. Multiple applications can use a single connection.

However, two applications on the same node cannot connect to two different InTrack database servers.

Error Handling and Debugging

InTrack does not log error messages by default. The application programmer is responsible for trapping errors and developing error-handling procedures.

During the debug process, the ITRTrace.exe program can help identify development problems. This program lists all calls made from InTrack, including error messages. However, it creates a large file and is not recommended for use during normal production.

The application should be written to capture and log errors to the WWLogger or another similar log file.

All method calls return an error code or have a parameter for an error code.

To retrieve the message associated with the return code, use the Database.GetErrorMessage(ReturnCode, ErrorMessage) method to retrieve the error as a string. This string can be logged to a log file to facility error tracking. Log the calling script and other pertinent information to determine the cause of the error.

Using a debug method in the application to log variable status information is recommended, especially when using InTouch. Insert a discrete (or integer) tag that can be monitored throughout the application logic. When the debug tag is TRUE, log a text message to a debug file (this may be the same as the error log) describing the action completed or the action to take.

Building a debug feature in the application will make troubleshooting errors much easier. The cost effectiveness of creating the debug feature will depend on the complexity of the application and the programmer’s familiarity with InTrack error messages.

Project Management Guidelines

Managing an InTrack project is similar to managing any other automation project.

Start with a set of requirements, a project plan, a schedule with some intermediate sign-off dates, documentation requirements, and other automation planning documents relevant for your installation.

Additional time may be needed to implement the InTrack system for ERP integration, complex reporting requirements and developing applications that require long-term maintenance.

ERP Integration

It may be necessary to coordinate the InTrack installation with the supplier or developer of your ERP program. Allow extra time if you need to coordinate work with representatives of two or more different companies.

System Administration

Make decisions on your specific system architecture, server management software, database management procedures and administrator support early in the project.

Reporting Requirements

Reporting requirements should be outlined at the start of the project. Information needed to generate reports must be built into the initial model.

The system must be modeled to accommodate all the details that need to be reported.

Adding information to the model to accommodate additional report can be costly after the application is finished.

Modeling

The model will determine most of the actions governed by the application and the format of the information. A poor model can lead to a significant increase in the number of transactions and the complexity of the reporting.

When building the application, remember that you are using a database model with limited storage capacity.

Attention to tracking the most effective information for problem solving is critical. Modeling a system that requires excessive database transactions to gather required information or using transactions to gather nonessential information can affect system performance, resulting in slower production tracking and wasted system resources.

Data Archiving and Retrieval

When determining the scope of the project, include a mechanism for archiving or purging data from the production database.

The removal of historical data should be included with the delivered application. The ability to recover purged or archived data also may be a major consideration in some industries.

Data archiving is necessary to avoid overloading the database with obsolete data. Archiving is not difficult to manage when planned properly.

Coding Guidelines

Most programmers or development groups adhere to coding conventions.

Standards may include naming conventions, project organization standards, change control methods, etc. Use existing coding standards and internal conventions whenever possible.

Managing Modular Code

When multiple developers are working on a single application, it is common to enforce a practice that each code segment must stand alone.

This practice may cause excessive database activity and lead to serious performance problems with InTrack.

Developers must remember that every .Load and .GetXX method call initiates one or more queries to the database. If one section of code has loaded the object to be used in a second section of code, the object should not be loaded a second time.

Coordinate the units of a complex application and review them for duplication or excessive use of system resource.

Memory Allocation

The creation of every OLE object requires memory and processing time.

Since the amount of memory required to manage an empty object requires is fairly small, frequently used objects should be created on initialization of the application and released only when the application is closed.

Recreating an existing object with the same name is the preferred method for resetting an object to its initial state.

Query objects may take up the most memory when in use. Because they contain multiple records from the database, queries should be released or recreated after the records they contain are no longer required. Releasing unneeded queries will reduce memory usage significantly.

Using Visual Basic vs. InTouch

Most InTrack implementations use Microsoft Visual Basic as the InTrack OLE Automation container because of the many debug options available.

The total number of lines of InTrack-specific code will be about the same for both Visual Basic and InTouch.

The decision depends on the project development time frame, the expertise of the integrator and the project requirements.

In some instances, using Visual Basic and InTouch may prove to be the most efficient in terms of development.

For applications with a lot of user interface graphics and straightforward InTrack code, InTouch will enable faster development.

If the InTrack scripting is complex and the user interface will be simple, such as using ActiveX objects and other basic tools, then Visual Basic will be faster.

Also, if the integrator is familiar with Visual Basic and not InTouch, there may not be an advantage to using InTouch for the graphical user interface. If the integrator knows InTouch and not Visual, InTouch would be preferred since Visual Basic is more complex to learn. If the integrator knows both, use each product for its distinct advantages.

Exclusive Visual Basic Methods

A few methods are available only to Visual Basic users. Although all tasks and operations can be accomplished in both environments, some methods had to be modified to support InTouch.

In the Database object, the Database.GetUDA(ClassID, ObjectKey, UDAName, Value) method is only supported for Visual Basic. The Database.GetUDAValue() method will work in both.

The structural object DatasetItem has a limitation in InTouch: the method DataSetItem.GetRanges(InstHigh, HighHigh, High, Low, LowLow, InstLow) will not work.

Use the DataSetItem.GetLimitValue(LimitID, ReturnCode) to retrieve the limits one by one, or use the DataSetTemplate.GetItemRealRanges() or DataSetTemplate.GetItemIntegerRanges() method.

InTouch Numeric Data Types

When creating data items for data collection tables or other user-defined tables in InTouch, the data type for real numbers must be double instead of a real.

The method call for saving data to a user-defined table will fail if the real data type is used in InTouch.

CHAPTER 6

User Characteristics

After your InTrack system has been developed and implemented, several types of employees will use and access ModelMaker and the runtime application:

- Process Control Staff and Control Engineers
- Production Supervisors
- Production Operators
- Production Support Personnel
- Chemists, Cooks, Scientists

System security considerations may determine the modeling of users and the privileges assigned to different groups. If operations your production process operations require special user training, the user certification object can be helpful in restricting access to the system at certain route steps.

The level of reporting and queries made to the system will impact InTrack performance. If a large supervisory staff will monitor operations frequently, or if multiple managers will require simultaneous access to data, additional system resources or the use of multiple servers may be recommended.

The overall computer proficiency of the users also will determine the features and level of graphics you will need to develop in the user interface of your runtime applications.

A basic description of common InTrack users is provided in this chapter.

Contents

- Process Control Engineers, 2
- Production Supervisors, 2
- Production Operators, 2
- Production Support Personnel, 3
- Chemist, Cooks, Scientist, 3

Process Control Engineers

Process control engineers configure and maintain the system. They will be closely involved with process design.

Generally process control engineers, or similar personnel, will define, design, implement, test and maintain the control system.

They should be involved with initial planning and well informed about operator interface specifications, plant floor information requirements, standard operating procedures, production reporting needs and interfaces to external systems. InTrack commonly interfaces with Material Resource Planning (MRP) systems, Laboratory Information Systems (LIMS), Preventative Maintenance systems and other third-party or existing production support systems.

Familiar with the system design, the process control engineer may be assigned to serve as InTrack system administrator.

In addition to managing ongoing system operations, the engineering staff will be responsible for archiving of historical data, installing and testing new software upgrades and solving technical issues that they arise.

Production Supervisors

Production Supervisors are responsible for plant floor production.

Supervisors will use the software system to schedule production, monitor production status and generate production reports. They will need access to system information and reporting functions in order to identify, diagnose and solve production problems.

Production supervisors typically are responsible for managing operators (users). Their role will include outlining system security, maintaining user profiles and monitoring ongoing user access to the system.

As the first in line to receive operator questions, production supervisors should have sufficient knowledge of the overall InTrack system so they are well equipped to handle operator training and some troubleshooting during normal operation.

Production Operators

Production Operators run the plant based on the production schedule and standard operating procedures for the plant.

Operator will use InTrack to control and monitor the status of the process equipment. They must be trained to identify, diagnose and solve exception alarms that occur.

Operators will be the primary users of the interface, responsible for entering production information and entering production data (such as machine failure reasons, data collection measurements, and other input). They need to know how to recognize how to respond to important alarm, access work instructions and other system information, and interpret customer specifications or other data that affects production.

Production Support Personnel

Production planners, industrial engineers, test technicians, QA managers, technicians, maintenance engineers, receiving and shipping clerks, are among the personnel accessing the system and enter information pertaining to their job function.

These individuals may be required to enter information about new materials, standard operating procedures, receipt of new material lots, shipments of production orders and possible new formulations.

Chemist, Cooks, Scientist

Chemist, cooks and scientist will use the system to develop, test and verify standard operating procedures, formulations and recipes for new products.

CHAPTER 7

FactorySuite Component Integration

In most installations, InTrack will be used with other components of the Wonderware FactorySuite.

InTouch is the FactorySuite visualization component. In a FactorySuite solution, an InTouch application will be the main engine for developing runtime applications. InTouch will be used to host the visualization and logic for InTrack. Since InTouch is an essential part of InTrack operation, review the InTouch documentation to understand the basic functions and features

Depending on your tracking and ongoing reporting needs, InTrack may be combined with one or more additional FactorySuite components:

- InSQL
- SPCPro
- InControl

Naturally, it is important to plan how these components will be deployed and used in the production process.

Additional consideration should be given to how InTrack interacts with these components and how the combination of components can be designed for maximum system performance and effectiveness of your automation efforts.

Considerations for component integration are provided in this chapter.

Contents

- InSQL, 2
- SPCPro, 3
- InControl, 3

InSQL

Wonderware's data historian, InSQL uses SQL Server as the database. When using InSQL, InTrack should use SQL Server for database management.

The InSQL and InTrack databases can reside on the same server or on separate servers, depending on the transaction rates for both products. If either InTrack or InSQL will have a high transaction rate, the use of separate servers is recommended.

InSQL will require as much server processor time as needed to maintain the specified collection rate. High InSQL collection rates will demand extensive processor time, which will slow processing for other programs . If both InTrack and InSQL use the same server, InSQL will take priority over processing and InTrack clients will experience slow response times during periods of peak InSQL demand.

If the InSQL load is low and InTrack transactions are high, both programs can be combined both on the same server. InTrack may experience slower performance for periods of peak InSQL activity, but overall performance degradation for InTrack clients will not be excessive.

As a general guideline, both products can operate on the same server if demands on both are low, or if InSQL demands are low and InTrack demands are moderate. However, for high InSQL demands, the two should not be combined on one server.

InSQL is a time-based storage system. The main link between InTrack's history tables and InSQL is based on time. Transaction time in InTrack is stored in the LotBaseLog or ResourceBaseLog TimeLogged and ActualTime columns. The timestamp issued when a transaction occurs is stored in the ActualTime column. User-specified time is stored in the TimeLogged column.

To coordinate the two systems, a typical approach uses the time range of two InTrack events as the range for displaying InSQL data. For example, you may perform a trend analysis based on the time a certain Lot is processed on a machine. You can query the LotBaseLog and StartLog to find the time of the first start transaction on the machine for the Lot. Next, query the LotBaseLog and CompleteLog to find the time of the last complete transaction on the machine for the Lot. These two times can be used to set the start and end time of the trend.

To use tags for trending, use InSQL's name manager to build groups of tagsthat match the machine and/or material names from InTrack. The named group enables you to display a list of tags associated with the machine or material associated with the selected LotID.

InSQL's ActiveDataGrid ActiveX object is a good tool for displaying data from any database table such as data collection tables, user defined tables and external tables.

SPCPro

SPCPro is Wonderware's statistical process control product.

SPCPro uses SQL Server or MS Access as the database. When integrating SPCPro and InTrack, use SQL Server.

As with InSQL, it may be advisable to use separate servers for InTrack and SPCPro operations.

If SPCPro has a high transaction rate, it should reside on its own server.

In most cases, the SPC client takes a greater toll on performance than the database server. Consequently, both databases can typically reside on the same server. The SPCPro database schema does not contain a large number of tables, and the SPCPro tables can be placed in the same database as the InTrack tables.

SPCPro is also a time-based storage system. For example, you can produce a SPC chart using the time span of an InTrack LotID. Use the start and end times of the InTrack transactions and to call the SPC chart for the dataset desired.

To link the two databases, create SPC Products with the same names as the InTrack materials.

For better integration, use an InTrack data collection template to store saved SPC sample numbers. The collection template can store the LotID and dataset name as the sample name, with optional machine name. This will create a direct link to the LotID running in InTrack when each sample point was collected.

InControl

InControl is Wonderware's controller product.

InControl has an excellent programmable engine for handling scalability issues with InTrack.

Multiple InTrack transaction engines can be managed by one InControl engine. The InControl engine accepts commands from multiple client applications and determines which InTrack transaction engine is available for processing the command(s).

All results, processing time, failure descriptions, etc., are returned to the calling client application.

By building intelligent symbols in InControl and making good use of InControl's programming, the client application can decrease the complexity of InTrack's OLE automation engine.

CHAPTER 8

Elements of InTrack

InTrack simulates, or models, the factory environment to track information throughout the production process. Objects in the program are based on common physical components, such as machinery, operators, and raw materials.

Data management for each InTrack object follows the logical measurements required for its physical counterpart. For example, InTrack machine objects are stored in database tables with information on capacity and maintenance intervals. Tables for materials store data for unit of measure, description, lifespan, etc.

The route object manages the associations of the individual objects – how materials are moved through the process, the operations performed at each production step, the destination of products or byproducts, etc. The route simulates the activities of each physical assembly line.

The model of the production process is the framework used to determine how runtime information will be collected and processed in the database. The graphic user interface, InTrack's ModelMaker, enables the set-up of the factory floor model. OLE Automation also can be used to create the model.

InTrack's Runtime component enables the creation of a custom software program specific to the data collection and reporting needs. A wide variety of operator interfaces, runtime displays, and data management tools can be developed with InTouch visualization software or automation tools such as Visual Basic. InTrack provides ActiveX controls and integrates well with other FactorySuite components, PLC controllers, and other automation tools.

Runtime components do not have a standard user interface – InTouch is generally used to create and customize the displays used by the end-users. Like other FactorySuite components, InTrack provides a full set of tools that can be used by in-house programmers or system integrators to build effective tracking systems.

In addition to the ModelMaker and Runtime tools, InTrack's powerful OLE Automation server enables comprehensive database management during runtime.

This chapter provides an overview of the InTrack elements and how they can be used effectively.

Contents

- The Process Model, 2
- OLE Automation, 3
- Structural Objects, 4
- Activity Objects, 11
- Query Objects, 27

The Process Model

The model you develop provides the basic framework for subsequent tracking activities. Careful attention to your transaction and reporting needs is required to build the tracking system. The model defines the level of detail for database storage. Determining the type of transactions to store and the volume of database reporting is critical for using the data effectively.

Process modeling used to track a product may not require the same level of detail as the typical process flow diagram used to design the actual production lines. Avoid modeling every step through which a product must move on the route. For example, in continuous or automated discrete manufacturing, there is little queue time between the actual steps in the assembly. Unless the individual pieces are identified uniquely, they are indistinguishable among other pieces in the same lot. Executing a tracking transaction for individual units in a continuous lot would complicate the database without providing any usable tracking data.

Grouping transactions in multiple quantities will provide a reasonable level of detail and more functional information. If the product is not stored, queued or other wise available among the different operations, the individual operations should not be modeled as unique steps.

In classic InTrack models, a “route” is defined as a sequence of operations. These operations are a direct representation of the manufacturing steps for product assembly. As products move among the different route steps, they are queued at the next operation before being started. After an operation is complete, the operator has the ability to route the product to another operation based on the disposition of the product. This concept maps into discrete manufacturing very well.

For other types of manufacturing, the classic model may not be practical. Many factories employ extensive automation in the manufacturing process. Materials moves quickly through individual automated operations, so tracking may not be practical at each unique operation. The route model may only need to record pointers when a lot of material enters or leaves an automated process, tracking only overall quantities, lot movement, dead time, etc. Even though the product may pass through a series of automated operations, the process model may be more effective with a route comprising only a few basic steps where lot tracking is necessary. In automated processes, InSQL is an effective tool for recording any relevant sensor data generated directly from the machinery.

In automated processes, material consumption may be continuous or aggregated. This consumption can be posted against the production lot before the route is completed to avoid overloading the InTrack database with intermediate transactions. Again the real-time details of consumption, flow, feeds or other rates can be managed effectively with the InSQL Historian.

Using InTrack as the transactional production engine and delegating the real-time collection of data to the InSQL historian will enable management of a large volume of data while making the most efficient use of system resources.

NOTE: Refer to the ModelMaker User Guide for complete information about the elements used in the process model and the type of database transactions that are commonly tracked within the route.

OLE Automation

The InTrack OLE automation server enables manipulation of the database during runtime.

OLE automation relies on two basic types of transactions: structural objects are used to manage the model, and activity objects manage the lots.

Each object in the InTrack database has a corresponding OLE object, which can create, delete or change the database objects. Nearly all ModelMaker objects and functions are included in the OLE automation server.

OLE objects provide flexibility for accomplishing the same results using different parameters or methods.

This chapter provides an overview of the structural objects and activity objects managed by the OLE automation server. For complete details on the objects, refer to the OLE Reference Guide.

In addition, a brief description is provided for the Query objects used to retrieve significant data from the InTrack database.

Structural Objects

Structural objects define the users, materials, machines, and the relationships among them.

Traditionally, ModelMaker was used to build the database model. However, the structural objects in the model can be built using OLE calls.

Objects may have subordinate objects that link to subordinate tables. Typically, an OLE object is associated with a single database table, with individual columns for each of the object's properties.

Some table properties can include other OLE objects with multiple columns. For example, an amount object (that includes properties for quantity and unit of measure) is often used as a property for another object. When the main table links to other tables, the structural OLE object will have an OLE property for each subordinate table. The subordinate OLE objects will have properties that link to the columns in the subordinate table. The only exceptions are the Material object, which includes information from the Material and BillOfMaterial tables, and the Route object, which includes information from the Route and StructureStep tables.

Users can add new properties to the basic tables. These user-defined attributes (UDAs) are added as columns to the base table. These additional columns will not become new properties of the OLE object. However, the UDA column values can be accessed (using the GetUDAValue and SetUDAValue methods for the OLE object). Refer to the OLE Reference Guide for complete information on the properties and methods for each object.

Calendar Object

The calendar is a set of tables that contain information about years, months, days, and shifts. Currently, no OLE object is available for defining the calendar. InTrack supports the use of only one calendar, called "Manufacturing." Use ModelMaker to schedule shifts, workdays and holidays. Only four shifts can be created in ModelMaker, although you can create an MES file containing additional shifts and import it to the database.

Customer Specification Object

The customer specification object (CustomerSpecification table) is a simple object with a name, version and description.

It is used to associate a WIP lot with a specific customer. A customer specification can be used to model modifications to standard processing for a customer, by applying material overrides for customer specifications at an operation. The data collection, work instructions, machines, or operation properties can be modified depending on which customer specification is specified.

An existing customer specification can be assigned to a WIP Lot during creation or anytime later.

Customer specification records can be used to maintain a list of customers to populate a selection for the ship command. (Because the Sublot.Ship method does not validate customer name against the database, the customer specification table can be used to designated specific customers.)

UDAs also can be added to record shipping addresses and contact information for customers.

Dataset Template Object

The dataset template object (DSTemplate table) is used for collection of runtime data, stored in user-defined tables. The basic dataset template table contains a name, version, description, the name of the user-defined table, a link to a list of sample names (optional), and a link to a list of data items (columns in the new table). Data collection can be activated at runtime. (Refer to the Activity Objects for additional information.)

InTrack stores the name of the data collection table and its column names in the DSDataItemHeader and DSTemplate database tables.

A sampling plan must be created and stored in the DSSampleHeader table. The sampling plan can be fixed or variable. A fixed plan is based on a specific number of samples (with predefined sample names). A variable sampling plan enables the automatic generation of sample names at runtime. A variable sampling plan with zero samples and zero percent of material allows optional data collection.

After creating a sampling plan, a list of data items defines the information to be collected for each sample. Data items can vary in type, one data item can require a numerical value, another a text (string) value, etc.

SQL Server limits the number of columns and bytes in a row note (SQL Server 7.0 allows 1024 columns and 8060 bytes per row). String data types default to a column width of 80 characters, so column widths should be shortened to avoid exceeding the limits. When the table is created, InTrack adds 11 additional columns with seven primary keys. For faster retrieval, indexes should be added to user columns that will be used in query expressions.

To save time creating similar dataset templates, a Parent template can be created. “Children” assigned to the parent template will use the same data table. The child template can have its own sample names and sampling plan (DSSampleHeader entries), but it uses the same data items (columns) as the parent template.

Disposition Code Object

The disposition code object (DispositionCode table) is a simple object with a name, type, and description. Four disposition code types (good, rework, reject and scrap) are used to model the movement of material and differentiate transactions. During database initialization, a default code of “OK” (good) is created and used for routes.

Material moves through the model using a Complete or Close (at the last step) command. The disposition code passed to the command determines the next destination for the material and validates if the amounts specified are within tolerances.

The InTrack system treats all disposition types the same and continues to monitor the material. InTrack will stop tracking materials with disposition codes assigned with the “scrap” type. Lot and sublot tables are decremented by the quantity scrapped. If the entire amount is scrapped, the sublot record is deleted. In all cases, the entire amount of the lot will continue to be tracked in the database.

Location Object

The location object (Location table) stores only a name and description. The location is used as a key field to identify sublots and physical position of machines.

The location also can be used in the model to manage materials available at a route step. When a location is assigned, material to be consumed must be available at the location.

EXAMPLE

Product lot A requires 20 units of material X. In the database, material X can be found in lots X1 (25 units) and X2 (10 units) at locations L1 and L2, respectively. If the route models material X for consumption at location L2, there is not enough material available to satisfy requirement of 20 units of consumption. Lot A cannot be processed beyond this step, although there is more material X in location L1.

To enable consumption from any location, do not assign a location for route input materials.

The location model in InTrack is a single level model. To add a hierarchy of locations, a user-defined attribute for "ParentLocation" can be created. Use the ParentLocation User-Defined Attribute to point to the next location level. Adding UDAs for parents enables creation of an unlimited number of location levels in the hierarchy.

Machine Object

The machine object (Machine table) links to many other objects in the model. Machine objects are assigned a name, machine type, location, capacity, as well as optional descriptive information. Assigned to a machine type object, the machine inherits machine tasks that are monitored in the MachineTaskStatus table.

Machines can be assigned to operations (OperationMachine table) to monitor the lots processed on a given machine and to update the machine tasks. The MachineQuantity table maintains a list of lots currently running on the machine.

Machines also can be assigned data collection templates (MachineDatasets table) to record machine information, such as calibration measurements.

Machines are assigned an operational status and a maintenance status in the MachineStatus table. As machine tasks or machine repairs are completed, the MachineStatus table is updated. The operational status indicates if a machine is in use, idle and ready to be used, or offline undergoing repairs. The maintenance status indicates if the machine is currently ready, in planned maintenance (PM) mode running a machine task, awaiting PM because a machine task has expired, in repair mode running an unscheduled maintenance repair, or waiting an unscheduled maintenance repair.

The combination of machine types and machines enables modeling on two levels. Both objects support UDAs that can further enhance the stored information. The machine object also has data collection capabilities. This flexibility enables the design of complex machine architectures and the modeling of complicated machines with replaceable tools or parts.

Machine Type Object

The machine type object (MachineType table) serves as the building block for machine objects. It stores a name, description, type, and capacity units of measure. It also can contain failure reasons, failure symptoms, and machine tasks.

The machine type defines a certain category of machine, such as a packaging machine. The machine type must be defined either as batch or serial. Batch machines start one or more lots that must be completed before additional lots can be processed. Serial machine can start a second lot with another lot running.

Failure symptoms (MachineTypeFailureSymptom table) record machine problems. The maintenance staff can enter failure reasons (MachineTypeFailureReason table) to indicate the reason for the problem.

Machine tasks (MachineTypeMachineTask table) are scheduled maintenance activities monitored on a periodic interval, based on the number of units processed, number of lots processed, machine running time, or clock time.

A machine type object must exist before a machine object can be created. All machine objects linked to a machine type inherit the defined tasks, failure reasons, and failure symptoms.

Material Object

The material object (Material table) records information on all materials that will be tracked by the system, including raw materials (input materials), intermediate products, final products, co-products and byproducts. Material records also can be used to record other resources such as water, electrical power or labor hours.

Materials are assigned a name, version, description, life span, type, and a unit of measure. Materials are modeled as either lot or bulk. Lot-based materials use a LotID to identify individual shipments of the same material. Bulk-based materials are tracked only by name and are identified by location.

After a material is assigned to a route, it is considered a “product.” The Material name and version is copied from the Material table to the BOMMaterialName. A record also is created in the MaterialStructure table.

Both lot and bulk materials can be associated with a route. Since bulk materials are tracked only by name, the LotID for a bulk WIP lot must be recorded as material[version].

After a material is associated with a route, a bill of material (BOM) can be built. In most cases, the standard quantity of the parent material is one and the consumable material list is scaled accordingly.

The BOM (BillOfMaterial table) is used to model material consumption (BOMItems table), substitute materials (BOMSubItems table), and output materials (BOMItems table) known as co-products and byproducts. In the BOM, tolerances for material consumption and output also can be modeled, as a default value. All material consumption and output enforcement is performed at the route step.

In the model, material objects are used to create material or customer specification overrides. Overrides can modify any attribute of an operation to permit product customization. After a route has been defined, both material and customer specification overrides can be applied. At runtime, InTrack loads the original operation information. Material overrides are applied first, followed by customer specification overrides. Override information is stored in the MaterialOverride table and associated tables (OvrDataSets, OvrOperationMachine, OvrSetpointTemplates, and OvrWorkInstructions).

User-defined attributes are frequently applied to material objects to enable storage of additional information, such as cost, chemical and/or physical properties, preferred supplier information, MSDS data, etc.

Operation Object

The operation object (Operation table) defines the processing activity and its requirements. It includes a name, version, description, time information, and lot processing information. Each route step (RouteStep table) is tied to a single operation.

Operation properties control the movements of lots through the route steps. The flow lot property indicates if an entire lot quantity must be processed together or if sublots can be processed.

Operations can be designated as “start and complete” to measure queue time separately from total processing time, or “complete only” of only total processing time should be tracked. Start and complete operations require two transactions to move a lot through the route step. Operations must be defined start and complete if machine tracking or user certification are required.

The list of machines associated with an operation is stored in the OperationMachine table. Because the list of available machines is processed as an “either/or” list and not as a series, the same unit can be processed only by one machine in an operation. However, different units of the same lot can be processed on different machines. To specify multiple machines for the same unit, separate operations are required.

Data collection is defined and enforced at the operation level. Collection of the number of samples defined in the dataset template must be completed before the lot can be completed. The list of data collection templates associated with an operation is stored in the OperationDataSets table.

Multiple work instructions can be associated with an operation (OperationWorkInst table).

Operations may be associated with a list of setpoint templates (OperSetpointTemplates table) that store operational setpoints for equipment settings.

Route Object

The route object (Route table) defines the movement of lots through the system. Each object has a name, version, description, and information about the individual steps (RouteStep table) in the route. Route steps are associated with operations to be performed.

The movement of lots from one route step to the next is defined by the disposition code. A lot with a disposition code of good or rework will be moved to the next route step defined in the RoutePath table. If the lot assigned a disposition of scrap, the information is stored in the StepDispCodes table.

Based on the disposition code assigned, lots can move from one step to another in the same route, from the step to inventory, from the step to scrap, or from the step to another step in a different route. If material is moved into inventory or on to another route, a different material can be output. All material output must be modeled in the BOM and associated to the route (StepOutput and StructureStep tables). If the material changes and the parent material is decremented, the material is defined as a co-product. If the material changes and the quantity of the parent material remains the same, the material is defined as a byproduct.

As material is moved through the route, tolerances can be used to limit the amount of material assigned a new disposition code. Step-to-step disposition codes pertain to all products (materials) produced on the route, while material output disposition codes pertain only to one product (material) produced. Refer to the overview of Activity Objects (lots) in the next section for additional information about the treatment of materials moving through the route.

Material consumption is defined at the route step (StepInput and StructureStep table). Tolerances can be assigned to limit the amount of material consumed. Only material included in the BOM can be defined for consumption or material output. Each product (material) produced on the route has a unique list of material inputs.

If the operation assigned to the route step contains setpoint templates, the corresponding setpoints can be assigned as well. Each product (material) produced on the route has its own list of setpoint values. These setpoint values can store information on machine settings, recipes, processing information, etc.

Security Group Object

The security group object (SecurityGroup table) organizes user privileges. The object is defined by name, description, and list of privilege assignments. During the creation of the InTrack database, three security groups are created: ModelMakerReadAll with read-only privileges; ModelMakerReadWriteAll with full privileges except database initialization and archiving; and RuntimeAll with runtime transaction privileges. The list of privileges for a security group is stored in the SecGroupToPriv table and can be accessed through the Privileges object.

Setpoint Template Object

The setpoint template object is a simple object defined by a name, version, description, and a link to the setpoint items.

The collection of setpoint items in the template stores target values for each measurement, with optional high and low limits.

A setpoint template is associated with an operation. When the operation is linked to a route step, individual setpoint items are applicable to each product (material) produced on the route.

User Object

The user object (Users table) is a straightforward object with a name and a hidden password. Users can have an assigned list of direct privileges or can be linked to a security group and its privileges. Users also can be assigned user certifications (IssuedUserCert table) .

An InTrack user is not a part of the database security system. An InTrack user cannot connect to the database through any means other than the InTrack connection configured in the InTrack Database Setup Utility. The database server registers all InTrack connections based on an the users assigned by an administrator with database management privileges. InTrack administrators create the table structure, data collection tables, and user-defined tables.

The database UserName, database type, and other information are stored in the InTrack.registry. The database password is encrypted in the \FactorySuite\InTrack\InTrack\bin\config.dll file. When the InTrack database is first created, the database UserName and password defined from the InTrack Database Setup utility are placed in the InTrack User table with all InTrack privileges. To avoid security breaches, this initial entry should be deleted and replaced with a unique InTrack administrator ID and password. After the initial administrative record has been secured, non-administrative InTrack users cannot connect directly to the database or access any tables outside the InTrack program.

When a user is assigned to a security group, the group's privileges are inherited. Additional user privileges can be added or revoked for individual users, overriding the group privileges. These added and deleted privileges are stored in SecUserAddedPriv and SecUserDeletPriv. The group assignment is stored in SecUserToGroup. All privileges have a privilege code and can be returned to a Privileges group object and read individually with the Privilege item object.

User Certification Object

The User certification object (UserCertification table) is a simple object identified by name, description, and time interval. User certifications are assigned to a user, valid for the assigned interval.

User certifications can be assigned to operations, providing additional security during runtime. Only users with an active certification can perform the operation. Because operations are linked to route steps, the user must have an active certification for the operation to process a lot.

Work Instructions Object

The work instructions object (WorkInstructions table) are used to provide operator access to documents. The work instructions records is identified by name, version, description, label, and notification information.

The document can be stored be in a wide variety of formats commonly found in the industry: text, spreadsheet, graphics and others. At runtime the documents are displayed using Document Viewer.

Work instructions can be stored as text typed directly into the database, as a file embedded in the database (snapshot), or as a linked file (path only stored in the database). If the file is linked, any updates to the original document will be retrieved when the work instructions are viewed.

Work instructions can be assigned to an operation or a machine type machine task. The Instruct Dialog object is used to view both modeled work instructions and any non-modeled file of an acceptable type.

User Defined Tables

Additional tables can be created in the InTrack database using the user-defined table (UDT) object, defined by name and a list of columns.

By default, InTrack creates a Name column assigned as the primary key to the table. During runtime, the name of the row must be specified with the SaveRow method. When a user-defined table is created, the table columns and formats can be assigned, default values, string lengths, and indexing can be assigned.

InTrack assigns a class ID to the table (ClassDefinition table) so that information can be retrieved using the generic query object described in the following section. To perform querys on data in a user-defined table (other than the Name column), the column should be indexed.

Activity Objects

Activity objects store historical data in the database. Activity OLE objects include Lot, Sublot, and DataSetSample. The Machine object also has methods considered activity-based.

Every activity transaction will insert a record in the LotBaseLog, which tracks changes to materials, or the ResourceBaseLog, which records changes to resources (machines and users). Some transactions can have an entry in both tables. When both tables are affected, the BaseLogIDs will be different in the two sets of tables. Together, these two history tables and the associated subtables enable you to reconstruct the entire history of a lot or machine. Current information about a lot or machine is stored in the Lot, Sublot, SerialNumber, MachineQuantity, MachineStatus, and MachineTaskStatus tables.

The Lot object has a key of LotID. It is used for tasks pertaining to the entire lot such as creation, change of status or adjustments of various properties.

The Sublot object has a combined key of LotID, Route, RouteVersion, RouteStep, and Location. In the database table, this combined key is distilled to LotID and a SublotIndex maintained by InTrack. The sublot object is used for the majority of the transactions against a lot or partial lot.

The DataSetSample object has a combined key of template name and version. It is used to write or update data in the user-defined dataset template.

The Machine object has a key of Machine name. It is used to batch multiple sublots on a machine and manage maintenance tasks and repairs.

Basic functions of the activity objects are outlined below.

Creating Lots

All InTrack material stored in the database is contained in a lot object, with corresponding number of sublots that represent the lot quantities distributed across the plant floor. After a lot has been reduced to a quantity of zero, the sublot can be removed from the database.

InTrack manages three categories of lots BULK, LOT and WIP. All three are stored in the same database tables and differentiated by the values stored in the columns.

In a BULK lot, the MaterialName and MaterialVersion are the same as the LotID. For example, a MaterialKey of VulcanizedRubber[V123] would have a LotID of VulcanizedRubber[V123]. Only one lot record for a material can be maintained in the database. Bulk material can be spread across many Locations, Routes, and RouteSteps using multiple sublot records, but it can never be separated from its parent lot.

A LOT-based lot has a unique LotID of up to 40 characters assigned to it. These lots can have many records in the database for a single material. Lot-based material can be spread across many Locations, Routes, and RouteSteps using multiple sublot records. These sublots can be detached from the parent.

WIP is any lot's sublot record with an assigned Route and RouteStep. For Inventory, the lot's sublot is assigned a Route and RouteStep of "NONE". Because of this relationship, Inventory can become WIP and WIP can become Inventory. WIP lots can apply to both bulk and lot materials, but they require an association with a modeled Route.

Two OLE automation objects can be used to create Lots in the database:

```
%Lot.Create(SublotKey, Material, %PrimaryAmt, %SecondaryAmt,
   Priority, CustSpec, VendorID, VendorLotID, %DueDate);

%Sublot.Create(Material, %PrimaryAmt, %SecondaryAmt, Priority,
   CustSpec, VendorID, VendorLotID, %DueDate);
```

Both Lot.Create and Sublot.Create create the same records in the database. The SublotKey is a parameter in the %Lot.Create method and a property of the %Sublot object. The Create object can be used to either create BULK, LOT or WIP records in the database. For each create, a new record is inserted into the lot and sublot table. Since this function does an insert, the SublotKey to be created cannot exist in the database.

To add or receive new vendor lots to an existing lot, the %Sublot.Receive method can be used to update existing lots or sublots:

```
%Sublot.Receive(%PrimaryAmt, %SecondaryAmt, VendorID, VendorLotID);
```

The receive method of the sublot object will update only an existing lot/sublot record in the database. The Material, Priority, CustSpec and %DueDate parameters are not allowed on this method, since they are properties of the existing lot/sublot.

When working with BULK materials, if lot does not exist in the lot/sublot table, use the Lot.Create or Sublot.Create method. If the lot exists, use the Sublot.Receive method. These two methods Create and Receive are mutually exclusive and cannot be interchanged. If you are unsure of the status of a LotID in the database, you can use a planned failure mechanism to process the material. To create a planned failure algorithm, issue a Sublot.Create call. If this fails with a duplicate object key, issue a Sublot.Receive. This is just as efficient as performing a query of the lot database.

Lot.Create, Sublot.Create and Sublot.Receive will be logged as a CREATE/RECEIVE in the LotBaseLog and inserted into the CreateLog. Depending on the call, new entries will be added to the Lot and Sublot table or existing entries will be updated with new quantities.

Adjusting Lots

All properties of an existing lot can be changed except the create date. The lot object has a series of Set methods that will change the properties of the lot including the LotID, material, and expiration date.

To change the primary quantity of a Lot, the Sublot object must be used because it contains the properties for the StartedAmt and QueuedAmt.

Exercise caution when changing the LotID or material in a Lot. The change in name is not carried through the history of the object. Consequently, a report on the history of transactions on a lot can become complex if the LotID changed multiple times. Any changes to a Lot or Sublot UDA also will also be logged as an ADJUST command in the LotBaseLog and as a new record in the AdjustLog.

Starting Lots

If a start and complete operation is assigned to a Route and RouteStep, all sublot at the route step must be started before they can be completed.

There are four ways to start sublots:

%Sublot.Start

The Sublot.Start method is the most commonly used start command. It moves the specified quantity from the QueuedAmt to the StartedAmt, with no restriction on the size of the sublot. You also can start multiple sublots for an operation. The Start command cannot be used on a complete only operation.

```
%Sublot.Start(%PrimaryAmt, %SecondaryAmt, OverrideQueueTime);
```

%Sublot.StartOnMachine

Sublot.StartOnMachine adds the Machine parameter to the method call. Only one lot can be started on a machine at a time. The machine capacity is checked when issuing the start command, so you can start only quantities less than or equal to the remaining available machine capacity remaining. The method also checks the statuses of all machine tasks to ensure that none have expired. The MachineQuantity table in the database stores the Sublot and Machine relationship for started amounts.

```
%Sublot.StartOnMachine(%PrimaryAmt, %SecondaryAmt, OverrideQueueTime,
   Machine);
```

%Machine.Start

Machine.Start allows a sublot to be started on a machine, used primarily for batch processing. One method accepts a StartItem object with the properties of SublotKey, PrimaryAmt, SecondaryAmt and OverrideQueueTime. A second method uses a StartItems object, which is a collection of individual StartItem objects.

```
%Machine.Start(%StartItem);
```

```
%Machine.Start(%StartItems);
```

All start commands will write records to the StartLog and LotBaseLog.

Specifying a machine will write a record to the ResourceBaseLog, MachineStatusLog, and MachineQuantity tables and update the MachineStatus table.

Consuming Lots

Consumption of material develops a material genealogy in the database. All consumption is executed with a Sublot object.

There are two basic methods for consuming materials:

- If the WIP lot is not serialized, use the Sublot.Consume method.
- If the WIP lot is serialized, use the Sublot.Assemble method.

Two simplified consume methods of the Sublot object are commonly used for non-serialized WIP lots:

```
%Sublot.ConsumeStdQty(SublotKey);

%Sublot.ConsumeQty(SublotKey, ConsumeQty);
```

The first method consumes the standard BOM quantity for the SublotKey specified. If the lot does not have enough material, the method will generate an error.

The second command allows you to specify a quantity to consume for the specified SublotKey.

Note that for both methods, the material being consumed is implied by the SublotKey specified. The material to be consumed must be modeled as an input material for the RouteStep

These simple methods can be restrictive. Two additional methods can be used for greater flexibility when consuming materials.

```
%Sublot.Consume(%ConsumeObject)

%Sublot.Assemble(SerialNumber, %ConsumeObject)
```

These two commands differ only by the addition of a SerialNumber parameter in the Sublot.Assemble method.

A serialized WIP lot must perform consumption with the assemble method and material must be consumed for each serial number in the lot. Serialized inventory can be consumed with either method.

Serialized WIP lots use the Sublot.Assemble method. Non-serialized WIP lots used the Sublot.Consume method.

The methods also can use additional general objects to further define consumption:

- **%ConsumableMaterials**
  Collection of %ConsumableMaterial Objects
- **%ConsumableMaterial**
  Contains Material, StandardQty, Unit of Measure, and %ConsumeItems
- **%ConsumeItems**
  Collection of %ConsumeItem Objects
- **%ConsumeItem**
  Contains SublotKey, %PrimaryAmt, %SecondaryAmt, ForceConsumption, and UpdateQuantity
- **%SerialNumbers**
  Collection of SerialNumbers
- **SerialNumber**
  SerialNumber (40 character string) or comma separated list of serial numbers

The %ConsumeItem and %ConsumeItems objects are most commonly used. If the inventory does not exist and the material has not been modeled, the %ConsumeMaterial or %ConsumeMaterials objects are required.

The ForceConsumption and UpdateQuantity parameters can be used to change InTrack's validation procedure for consume or assemble command:

- **ForceConsumption**
  This parameter enables consumption of material that is not modeled as an input material or a BOM material. You also can force consumption above the modeled upper tolerance (ForceConsumption set to TRUE). If the lower tolerance is set to 100% and ForceConsumption is set to TRUE, InTrack will not track consumption in the database. This manual procedure overrides the validation process because the parameter implies that the given is valid. Forcing consumption can increase performance by reducing the amount of record keeping required.
- **UpdateQuantity**
  This flag is used to specify if inventory is reduced upon completion of the consume method. The property is used only for consuming material that is not modeled. When set to TRUE, inventory lots are decreased and tracked for the material. If inventory tracking is performed by an external system, such as ERP, setting the UpdateQuantity flag to FALSE can improve InTrack performance. InTrack simply records the consumption to the log system without validating or decrementing inventory. However, the lot must exist in inventory for InTrack to log the material.

The following table provides guidelines on using the %ConsumeItem parameters:

	Inventory in InTrack Validate Against Inventory	**Inventory Not in InTrack Do Not Validate**
Modeled Input Materials	Update Quantity = TRUE Force Consumption = FALSE	Update Quantity = FALSE Force Consumption = FALSE
No Modeled Input Materials	Update Quantity = TRUE Force Consumption = TRUE	Update Quantity = FALSE Force Consumption = TRUE

The SerialNumber and %SerialNumbers collection allow consumption of material using only a serial number. A SerialNumber is a text string (not an OLE object) that contains a SerialNumber or a comma-separated list of SerialNumbers. The %SerialNumbers collection object contains an array of serial numbers.

The serial number identifies the LotID and Material, so these values are implied. The quantity for each serial number is fixed at 1.0 unit. ForceConsumption (FALSE) and UpdateQuantity (TRUE) are assigned as the default and cannot be changed. To consume serialized inventory with different parameters, use the %ConsumeItem object and add the SerialNumber to the %PrimaryAmt parameter.

Data for InTrack consumption is stored in the LotConsumption table (InTrack 7.0) or in the SublotConsumption table (InTrack 7.1) when the sublot has a started amount at a RouteStep. The SublotCoinsumption table tracks the amount of material consumed and completed at the RouteStep.

If the Flow Lots parameter is activated, tracking can be complicated and transaction processing performance will be affected. The LotConsumption table will grow in size considerably if the total quantity started at the RouteStep is not completed. To reduce the burden on InTRack processing, start only the quantity of material that will be completed.

InTrack 7.1 consumption tracking was enhanced to to reduce the number of records the SublotConsumption table stores when completing partial quantities. After a started quantity is reduced to zero by completing the total or by using the ForceCompletion parameter, records from the LotConsumption table are saved in a CompleteLogConsumption table. If a Close transaction reduces the started quantity to zero, records are written to the CloseLogConsumption table. These two additional tables reduce processing in the lot and sublot consumption tables, and also enable future undo transactions.

LotConsumption, SublotConsumption, CompleteLogConsumption, and CloseLogConsumption are internal tables and are not recommended for use in reports. Accurate historical information about consumption is retrieved from the LotBaseLog and ConsumeLog.

Starting with version 7.1, InTrack tracks machines used for WIP lot processing. The %ConsumeItem object includes a new property for the MachineName. If the WIP lot is being processed on a machine, the MachineName property must be set. If a sublot is being processed on multiple machines, multiple ConsumeItem objects must be used.

For all consume commands, a CONSUME transaction is recorded in the LotBaseLog and an entry is added to the ConsumeLog. The queued amount of the consumed sublot is updated in the Sublot table if the UpdateQuantity field is set, and a record is added to the internal LotConsumption table or updated in the SublotConsumption table (except for the case mentioned previously). The SerialNumber table may be updated to remove serial numbers consumed by the call and an entry or entries in the SerialNumberLog will be made. The ConsumptionDone column of the Sublot table is not updated automatically to reduce processing time. The update the Sublot table, use the Sublot.UpdateConsumptionDone method.

Completing Lots

A complete command performed on WIP lot at a RouteStep will cause the modeled function to be executed. The assigned disposition code directs the next function available from the RouteStep. Four types of completion are possible at the RouteStep:

- **On Route (OR)**
- **Material Output to a Route (MOR)**
- **Material Output to Inventory (MOI)**
- **Scrap Material (SM)**

At the last step of the route's main path, a disposition code of NONE is used to indicate that normal production is closed for the route after this step (there is no exit from a main path). For the last step on the path, the Sublot.Close method is required for a Close to Inventory (CI) function. The Sublot.Complete method used to move material from one route step to another will not accept a disposition code of NONE.

The complete command is a method of the sublot object. It can be executed only for a started sublot in a start and complete operation, or for a queued sublot in a complete-only operation.

The standard %Sublot.Complete method has the following syntax.

```
ReturnCode = %Sublot.Complete(%PrimaryAmt, %SecondaryAmt,
   Disposition, ForceCompletion, CreatedLot);
```

If machine tracking is used:

```
ReturnCode = %Sublot.CompleteOnMachine(%PrimaryAmt, %SecondaryAmt,
   Disposition, ForceCompletion, CreatedLot, MachineName);
```

or

```
%Machine.Complete(%CompleteItems);
```

where

The %PrimaryAmt and %SecondaryAmt objects specify the amount of the WIP sublot to be completed.

Disposition is the modeled disposition code that determines the destination of the material from the route step.

ForceCompletion is a flag used to delete the remaining quantity of the sublot from the databse (TRUE). InTrack will log the complete transaction; however, but no further logging of unused quantity deleted by a ForceCompletion method will occur.

CreatedLot specifies the lot ID of the destination sublot when material is output to inventory or to another route. If set as a NULL string (“”), the system will generate a unique lot ID to be used for returning the value.

The Complete command is logged in the LotBaseLog as a COMPLETE transaction and an entry is made to the CompleteLog. The Sublot table is updated accordingly, and any associated records in the LotConsumption and LotCollection tables will be moved to the CompleteLogConsumption and CompleteLogCollection tables. If machine tracking is used, the ResourceBaseLog, MachineStatusLog, MachineStatus, MachineQuantity, and MachineTaskStatus tables may be updated. The SerialNumber and SerialNumberLog tables will be updated if serial numbers are used.

On Route Completion

An On Route completion will move the selected quantity to the next route step based on the assigned disposition code. This method always decrements the WIP quantity. No tolerances are accepted and you cannot specify an amount greater than the amount started or queued.

Material Output Completion

Material Output completion creates a new Lot/Sublot record in the database. The DEC QTY option in ModelMaker is used to indicate if the output material will decrement the WIP quantity.

Unit conversion and quantity scaling can also be performed with the Material Output methods.

Materials can be output to a Route/RouteStep or to a Location, only if the destinations were modeled.

The complete method will use the CreatedLot parameter to choose a LotID for the destination Lot/Sublot. If this parameter is NULL (“”), the system will generate a LotID with format WIPLOTID_*xx*, where *xx* is a sequential number beginning with “01”.

Material output also can be a different material than the WIP material. All output materials must be specified on the Material BOM and modeled as Outputs for the Route/RouteStep.

The database enforces tolerances on the material output. If a Lower Tolerance of 0% is assigned, the entire WIP amount must be output and no partial quantities will be allowed. If a Lower Tolerance of 100% is assigned, any amount between 0 and the WIP quantity can be output. The tolerances can be used to downgrade partial WIP quantities to other products or to use Material Output to classify generic production into specific products.

Material Outputs provide the capability to execute complex material/route/lot relationships with simple coding in the client application. Material Outputs can be used to model a Master Route and SubRoute relationship, which can simply the model in complex manufacturing processes. Material WIP Lots also can be queued at another Route/RouteStep synchronized with the current WIP Lot to ensure simultaneous completion.

Scrap Completion

A disposition code with a destination of “Scrap” causes the selected quantity to be removed from the database. A log record of the scrap transaction is made (unlike the ForceCompletion parameter described in the examples below).

Closing Lots

The Close command is the most flexible Sublot method to remove lots of material from a Route. A Sublot.Close method allows you to specify the destination SublotKey (LotID, RouteName, RouteVersion, RouteStepName, and Location), so that any route step in the route can be closed.

Unlike the Complete method, the Close method is not modeled. The DispositionCode assigned in a Close method is provided for LotBaseLog reporting only and does not affect the function

The Close method can be used only if all material consumption and data collection are finished. The sublot can be InProcess or Queued.

The syntax for the Close method is:

```
ReturnCode = %Sublot.Close(%PrimaryAmt, %SecondaryAmt, Disposition,
   CreateLot, ToInvLot);
```

where

Setting the CreateLot parameter to TRUE will create a new lot. Setting it to FALSE will add amounts to an existing lot. The method will generate an error if the sublot exists and CreateLot is TRUE, or if the sublot does not exist and CreateLot is FALSE.

The ToInvLot can be used to close a sublot amount to inventory, or to a WIP lot. To close the amount to a WIP lot, specify a RouteName[RouteVersion]:RouteStep in the SublotKey passed to the ToInvLot parameter.

The close command does not include a ForceCompletion parameter to remove unused quantities from the database. If closing a partial amount, the remaining fractional amounts should not be left unprocessed in the database.

A simpler version of the Close method is Sublot.CloseAllQty. It simply accepts a disposition code as a parameter. This will then close the entire amount at the operation to an inventory LotID and Location the same as the WIP lot. This command can not be used for more than one combination of LotID and Location. If the WIP lot is flowed across multiple operations then subsequent calls of Sublot.CloseAllQty will fail since the inventory lot already exists.

Sublot.CloseOnMachine and Machine.Close allow the same functionality as Sublot.Close only operate on lots already started on a machine. These close methods are not interchangeable and must be paired appropriately.

The Close command is logged in the LotBaseLog as a CLOSE transaction and an entry to the CloseLog. The Sublot table will be updated accordingly, and any associated records in the LotConsumption and LotCollection tables will be moved to the CloseLogConsumption and CloseLogCollection tables. If machine tracking is used, then the ResourceBaseLog, MachineStatusLog, MachineStatus, MachineQuantity, and MachineTaskStatus tables may be updated. The SerialNumber and SerialNumberLog tables will be updated if serial numbers are used.

Moving Lots

The Sublot.Move method enables runtime movement of material lots that have not been modeled. You can move material from one RouteName, RouteVersion, RouteStep or Location to another. The Move method also can be used to transfer material from Inventory to WIP, WIP to Inventory, Inventory to Inventory, or WIP to WIP.

When moving lots to another route, the material must be associated with the destination route. WIP lots must be in the QUEUED state. The LotID of the Sublot cannot be changed.

The syntax is:

```
ReturnCode = %Sublot.Move(%PrimaryAmt, %SecondaryAmt, NewRoute,
    NewRouteStep, NewLocation, disposition);
```

Partial quantities can be moved. The disposition code in the parameter list is for reference only and will not affect the move.

A variation on the method introduced with InTrack 7.1, allows you to move a WIP lot that has been started:

```
%Sublot.InProcessMove(%PrimaryAmt, %SecondaryAmt, NewRoute,
    NewRouteStep, NewLocation, disposition, ForceMove, Machine)
```

The machine on which the lot was started must be specified, and a ForceMove flag is used to control the movement.

The Move command is recorded in the LotBaseLog as a MOVE transaction. The Sublot table is updated to reflect the new information, and the previous information is stored in the MoveLog table. If serialization is used, then the SerialNumber and SerialNumberLog will be updated. The InProcessMove command also updates the machine tables if machine tracking is used.

Merging and Splitting Lots

Use the Split method of the Sublot object to divide a single lot into two lots.

The Merge method merges two lots into a single lot.

The amounts must be QUEUED and the material in all lots must be the same. The status of two lots to merge must be the same. Partial quantities of a lot can be merged.

The LotBaseLog will record a SPLIT or MERGE transaction and create an entry in the associated SplitLog or MergeLog. Sublot and Lot tables will be updated. If serial numbers are used in any of the quantities, the SerialNumber and SerialNumberLog tables will be updated.

Shipping Lots

The ship method can be used to reduce a quantity in inventory.

```
%Sublot.Ship(%PrimaryAmt, %SecondaryAmt, Customer);
```

This method logs the quantities and the customer to which a lot is shipped.

The customer field is an open text field that is not modeled in the database.

Only queued lots can be shipped. Lots with on hold or on quarantine status cannot be.

The SHIP transaction is logged in the LotBaseLog and the associated ShipLog. If serial numbers are used in any of the quantities, the SerialNumber and SerialNumberLog tables will be updated.

Lot Status

The status of a lot can be changed to HOLD or QUARANTINE as needed.

A hold or quarantine status limits the activities that can be performed on a lot. Assigning a hold or quarantine on a lot will affect all sublots in the lot.

Lot status is stored in the HoldStatus and QuarantineStatus columns of the Lot table, so the same lot can be both quarantined and on hold.

To place an individual sublot record on hold, specify a location or create a user-defined attribute. InTrack does not automatically restrict the processing of individual sublots assigned a hold status, but customized restrictions can be built into the application. The location field is effective in excluding a lot from material consumption.

Data collection can be modeled to place a lot on hold automatically if a collected value exceeds assigned limits (numerical data collection items only). If the collected value exceeds a limit value, the entire lot is placed on hold and notification is returned through the method call. Subsequent entry of a "good" collection value will not remove the hold status. Removal of the hold status must be programmed in the application using the %Lot.SetHoldStatus(Status).

In the LotBaseLog, the transactions are identified as HOLD, QUARANTINE, RELEASE_HOLD, or RELEASE_QUAR. Transactions are entered in the HoldLog or QuarantineLog. The Lot table is updated with the new status.

Data Collection Using Dataset Templates, User Variables, and User Tables

Several objects in the OLE automation server enable data collection on a sublot and/or a machine:

Dataset Template Collection

Any existing dataset template can be used to collect data on a WIP lot. The template can be modeled or selected at runtime.

Two basic data collection merhods are used:

- **Save**
  The save method inserts a new record in the collection table. If less than one second has elapsed between collections on the same key fields (SampleName, CollectTime, LotID, OperationName, OperationVersion, Machine, and Location), the previous sample will be updated by the new one. Data collection more frequent than one second cannot be achieved unless a key field is changed. The Database.Backdate property will update a collected sample if the backdate is set to the sample collect time and all key fields are identical.
- **Supersede**
  The Supersede method replaces a previously collected record. The Superseded column is changed to TRUE and a new timestamp is inserted in the TimeSuperseded column. It is not possible to backdate a record that has been superseded.

Save and Supersede methods can be used for Lot objects, Machine objects, or for generic data collection. Six method variations can be used to add data to a Data Collection table.

- DataSetSample.SaveLotSample
- DataSetSample.SupersedeLotSample
- DataSetSample.SaveMachineSample
- DataSetSample.SupersedeMachineSample
- DataSetSample.SaveSample
- DataSetSample.SupersedeSample

Each collection type includes a Load method used to retrieve a previously saved value from the DataSetSample object in the database. It can be helpful for retrieveing information before performing a supersede method call.

Save and Supersede methods write or update the runtime table name modeled in the dataset.

The SaveSample method requires only the SampleName and CollectTime fields of the key.

The SaveMachineSample method requires the Machine and Location properties, which are written to the database. The Route and Step also can be used to insert the appropriate values in the OperationName and OperationVersion columns.

The SaveLotSample method can be used to write to all key fields if desired. Data collection by SaveLotSample is logged in the LotBaseLog as a COLLECT_DATA transaction with an entry in the DataCollectLog. If a route step includes an operation with a data collection assignment, InTrack maintains a list of samples in the LotCollection table. This internal table monitors modeled data collection. It is used when the Sublot.UpdateDataCollectFlag method is called to set the Sublot.CollectionDone property, which is not updated for a SaveLotSample.

If SaveMachineSample can be used if a the dataset is modeled for a machine. A record will be inserted into the ResourceBaseLog (MACHINE_COLLECT_DATA) and MachineDataCollectLog.

User Variables

InTrack enables the storage of user variables in the database. The %Database.GetUserVariable method reads variable information from the database; %Database.SaveUserVariable writes user variable to the database.

User variables are stored in the Configuration table. They are not modeled can be accessed only through the runtime interface.

A user variable is identified by name and description. Up to three values can be stored for a string, integer, or real data type.

The SaveUserVariable method overwrites the existing values in the database. InTrack performs no historical logging for user variables.

User-Defined Tables

The User-Defined Tables (UDT) object was introduced in InTrack 7.1. It also can be accessed in ModelMaker.

The table is created with only the key Name column. The %Table, %TableColumn, and %TableColumns objects can be used to save data to the table.

One or more %TableColumn objects sets the values for the user-created columns. They are added to the %TableColumns collection object and saved to the database using the %Table.SaveRow("NameValue", %TableColumns, 1) method. The first parameter must be a unique value for the primary key, Name. The row value is specified at Runtime using the the SaveRow method.

The last parameter is used to commit transactions to the database (TRUE). If not TRUE, a lock will be placed on the user table.

Not that, if InTouch is the operator interface, all floating-point columns must be defined as double.

Adding Comments

Comments can be added to the database in the CommentText field of the LotBaseLog or ResourceBaseLog tables.

There are two methods for adding comments to the database:

- A comment of up to 255 characters can be assigned to an activity transaction using the Comment property of the Database object. When set, the comment will be associated with subsequent transactions until the field is cleared.
- A Comment transaction also can be added to the database with the %Sublot.Comment() method. A COMMENT entry will be created in the LotBaseLog and the CommentLog tables for the transaction.

Monitoring Machine Downtime

Machine downtime can be scheduled (modeled as a machine task) or unscheduled based on failures (modeled or unmodeled). All machine activities are maintained and in the same tables.

A planned maintenance task is modeled as a MachineType task, based on an assigned wear measure (processing time, units produced or calendar time). The tasks apply to all machines assigned to the machine type.

As lots are processed on a machine, the MachineTaskStatus table is updated to reflect the current status of all assigned tasks. The task table is updated only during a transaction: time-based tasks are not monitored continuously. When a task is started on a machine, the MachineStatus table will be updated with the name of the task.

If unscheduled tasks are modeled, use the %Machine.StartTask() and %Machine.CompleteTask() methods.

If unscheduled tasks are not modeled, use the %Machine.OpenRepair(), %Machine.StartRepair(), %Machine.StopRepair()and %Machine.CloseRepair() methods.

Multiple tasks can run on a machine at the same time. Each task may have assigned work instructions.

Only one repair can run on a machine at a time. After a repair is opened on a machine, many methods are available to update failure reasons and symptoms, diagnosis, work status and repair person assignment. Comments, update parameters and repair person assignments will be written in the Description column of the MachineMaintActivityLog, with the corresponding MaintenanceType, which matches the TransactionName in the ResourceBaseLog.

No validation of the machine's FailureReason or FailureSymptom occurs when the Machine.UpdateFailureReason() and Machine.UpdateFailureSymptom() methods are used. Repairs are free form, so for effective identification, use the failure symptom name for the repairID (the ID may be reused) or comment. The structural OLE object can be used to create a new failure symptom or reason, if needed. After the repair is closed, use the failure reason in the comment field. Using the same symptom and reason names stored in the database as comments will provide a consistent set of values that can be used for future retrieval.

If more than one reason or symptom is required, use the update methods repeatedly as needed.

Multiple tasks and a repair can run on a machine at the same time. Both repairs and tasks can be used in combination to generate an automatic task with an assigned repair person, failure reasons, and failure symptoms.

In the database, all maintenance activities are written to the ResourceBaseLog with various transaction names. As tasks or repairs are initiated, the MachineStatus and MachineTaskStatus tables are updated with the current condition of the machine. The history of all commands will be stored in the MachineStatusLog (changes in operational or maintenance status) and the MachineMaintActivityLog (maintenance commands, details of tasks and repairs, and comments). The time to complete a repair or task can be found in the MachineStatusLog.

Database Locking

A returned Error 2038 indicates a database lock contention. This occurs if two nodes attempt to access the same row (Oracle and SQL Server 7.0) or data page (SQL Server 6.x) in the table at the same time. Two clients cannot lock a row or page at the same time. The database permits one connection to be locked and sends an error to the second.

If you receive the error, resubmit the command. In most cases, the lock will have been released and the command will be successful.

Database lock occurs commonly when two nodes attempt to execute a transaction on the same LotID. The error is more frequent when user transaction control is implemented, since the table is locked for a longer period of time.

InTrack will not reissue the command. The object that generated the error (typically the Sublot object) must be recreated and the command reissued. If the object is not recreated, the object persistence layer of InTrack will recognize that the command was just issued and generated an error. InTrack will simply return the same error without issuing any calls to the database.

To reduce locking conditions, limit user transaction control. Single InTrack methods should never be enclosed within transaction control. InTrack code uses transaction control internally when multiple tables are affected by a method call. If the user takes over transaction control, InTrack will not attempt to begin a transaction set. Consequently, the lock will be on the table for a longer period. If user transaction control is necessary, make all database queries before starting the transaction set and include only method calls that make changes to the database within the transaction set. This will minimize the amount of time the lock is maintained.

If database locks occur in an application, rewrite the code to trap the error and retry the commands. To implement retry logic, the entire transaction set must be reissued. The code should recreate any Lot or Sublot objects involved in the transaction set to clear object persistence. It is not necessary to recreate the Database object.

➢ Example: Unlocking the Database with Transaction Control

An InTouch example using transaction control follows. The error code for the commit transaction must be monitored since it may return a 2038 error as well as any of the method calls, and the Sublot object is recreated during each loop of the command. It is not necessary to have a long delay between executions. The example uses 200 ms which is sufficient in most cases.

While True condition script on Condition every 200 ms
(counter is set to zero in the On True condition)

```
IF counter < 3 then
   Failure = 0;
   RetryError = 0;
   OLE_CreateObject(%Sublot, "InTrack.Sublot");
   OLE_CreateObject(%Amount, "InTrack.Amount");
   %Sublot.SublotKey = Lot@Route[ver]:Step@Location;
   %Amount = StartAmount;
   %Database.BeginTrans();
   ReturnCode = %Sublot.StartOnMachine(%Amount, "MachineName");

   IF ReturnCode > 0 then
      IF ReturnCode == 2038 then
            RetryError = 1;
      Endif;
      Failure = 1;
   Endif;

   IF Failure == 0 then
      ReturnCode = %Database.CommitTrans();
      IF ReturnCode == 2038 then
            RetryError = 1;
      Endif;
   Else
      %Database.RollbackTrans();
   Endif;

   IF RetryError then
      counter = counter + 1;
   Else
      Condition = 0;
   Endif;

Else
   LogMessage("Three failed retries.  Exiting script");
   Condition = 0;

Endif;
```

Query Objects

Pre-built Queries

InTrack includes a large number of pre-configured query objects to save time when making standard queries of the database.

Each InTrack query object uses the same procedure and methods.

1. An object is created.
2. The query method is called with the necessary parameters (which set the RowCount property to the number of rows in the result set).
3. The SetRow method is used to retrieve the column information into the remaining properties for a given row.

ModelMaker uses many of the same standard queries available as OLE objects. When using the OLE queries, follow ModelMaker's structural guidelines to determine the interdependence between two structural objects. For example, when given the name of a route, the RouteMaterial query will supply a list of materials assigned to the route.

A few standard activity-based queries can be used to return information about active lots. For example, the LotMachineQuantity query object returns a list of machines and the quantity running for a supplied Sublot.

General Query

The general SQL query object (InTrack.Query.SQL) allows you to retrieve data from the InTrack database if a pre-built query is not available. Used like other pre-built queries, the Query method requires three parameters for the three main parts of a select statement:

1. A list of table names, followed by …
2. A list of column names, finishing with
3. The where expression.

Parameters for generic queries are verified against the InTrack database. The InTrack model must include all tables, including data collection tables and user defined tables. The column list must include valid column names in the defined tables. This validation process does present limitation for using the general query object: tables created externally from InTrack cannot be queried, aggregate functions cannot be used with InTrack version 7.0 (or earlier). For example, SQL.Query("Lot", "count(LotID)", "") is not valid in InTrack 7.0. Starting with InTrack 7.1, aggregate functions with group by function clauses are permitted. A stored procedure cannot be called from the generic query object, since the query always builds a select statement.

The where expression is passed through the system. If no where expression is needed, leave THE parameter blank. The word "where" must be included. It is possible to use other expressions such as order by. For example, SQL.Query("Sublot","LotID, MaterialName," where StartedQty > 0 order by LotID").

If the generic Query object is used with InTouch, the 131-character limit applies to each parameter. To keep the query compact use aliases, IN clauses and LIKE clauses. For example:

```
select Sublot.LotID, Machine, Sublot.RouteStepName, Sublot.StartedQty

from Sublot, MachineQuantity

where Sublot.LotID = MachineQuantity.LotID AND (Sublot.Location =
    'Raw Materials' OR Sublot.Location = 'Production Floor')

order by LotID
```

A generic SQL call would look like:

```
%SQL.Query("Sublot, MachineQuantity", "Sublot.LotID, Machine,
    Sublot.RouteStepName, Sublot.StartedQty",

"where Sublot.LotID = MachineQuantity.LotID AND (Sublot.Location =
    'Raw Materials' OR Sublot.Location = 'Production Floor')

order by LotID");
```

The SQL call will not work in InTouch because the where expression is 140 characters, exceeding the 131 limit.

The where expression can be reduced to 111 characters by aliasing the table names:

```
%SQL.Query("Sublot s, MachineQuantity m", "s.LotID, Machine,
    s.RouteStepName, s.StartedQty",

"where s.LotID = m.LotID AND (s.Location = 'Raw Materials' OR
    s.Location = 'Production Floor')

order by LotID");
```

Reduce the expression to 95 characters with the IN clause:

```
%SQL.Query("Sublot s, MachineQuantity m", "s.LotID, Machine,
    s.RouteStepName, StartedQty",

"where s.LotID = m.LotID AND s.Location IN ('Raw Materials',
    'Production Floor')

order by LotID");
```

or with the LIKE clause (assuming no locations begin with Raw or Pro).

```
%SQL.Query("Sublot s, MachineQuantity m", "s.LotID, Machine,
    s.RouteStepName, StartedQty",

"where s.LotID = m.LotID AND (s.Location LIKE 'Raw%' OR s.Location
    LIKE 'Pro%')

order by LotID");
```

If the 131-character limit cannot be achieved with these short-cuts, InTouch's SQL Access using ODBC is an option.

CHAPTER 9

Reporting

Reporting factory information is InTrack's primary function. Because so much information on the production process can be collected, defining the scope of the reporting needs and identifying the type of information required is required to use InTrack successfully.

Reporting requirements should be identified before you begin to model objects. The objects created and the relationships between the objects will determine the viability of any reports generated from the data tables.

The way information is detailed in the model can greatly improve the quality and effectiveness of reports. Consistent object naming conventions, proper grouping of objects, differentiating between essential and nonessential data to be tracked, and coordinating the InTrack system with reporting methods of other systems are important concerns.

This chapter provides basic reporting issues to address before beginning the InTrack modeling process.

Contents

- Table Relationships, 2
- InTrack Views, 2
- Using a Report Server, 3

Table Relationships

The names and schema of all InTrack database tables can be found in the appendix of the Runtime Development Guide. A copy of the database schema (dbschema.doc) also is available from the Wonderware website (www.wonderware.com). An Adobe Acrobat Reader file (itr71dbs.pdf) is available to show the relationships between the various database tables.

Two groupings of tables exist in the database:

- **Structural tables** contain information on all modeled objects such as materials, routes, and machines, plus Lot and Machine status.
- **History tables** record all database transactions for an activity object. History tables that show the flow, quantity and output of the production process are of primary importance for management reporting.

History tables are designed with a main log table and a collection of secondary log tables.

- For lot transactions, the main table is the LotBaseLog with a primary key on BaseLogID.
- For resource transactions (machines and users), the main table is the ResourceBaseLog with a primary key on BaseLogID.

The two main tables never contain a matching BaseLogID. The main tables have secondary tables with a primary key on TransLogID (except SerialNumberLog with a combined key on the BaseLogID and SerialIndex columns).

A common mistake in report set-up is attempting to link a secondary table's TransLogID to the main table's TransLogID instead of the main table's BaseLogID. The TransLogID in the main table should never be used.

The LotBaseLog has an additional index on the LotID column to assist in queries based on LotID. If performing a significant number of queries based on time, add an index to the TimeLogged or ActualTime column. Since LotBaseLog is the most frequently used table in the database, avoid adding too many indexes.

InTrack Views

A number of views are created in the InTrack database upon initialization. They make it easier to report some transactions and can simplify yield reporting, machine utilization, and consumption. Use the views as beginning step when structuring queries.

Using a Report Server

When using a reporting package, it is important to consider its demand on system resources and its impact on performance.

If data collection for reporting will require heavy usage of the database, large or frequent reporting activities could slow the production server significantly.

If your system will store a large amount of data or if you generate frequent or complex reports, the use of a second server for reporting is often recommended. Data from the production server can be replicated to a reporting server. All offline reporting activity can be performed on the reporting server.

Replication is a good idea if some InTrack data is not needed for long-term storage. Only data that has long-term value is replicated and stored on the reporting server. For example, the Production Server may keep data for two weeks, yet the Reporting Server can keep pertinent information for six months.

To use a reporting server, one option is to denormalize the database during the replication process. A larger table can be used to combine the LotBaseLog and its associated Log tables. Any unused tables or columns would not be replicated. Note that if the tables are altered before replication, the same reports designed for the combined or truncated table in the reporting database would not work for the production database.

A reporting server also is recommended when multiple production servers exist. The single reporting server can query information from all production servers. Replicating data from multiple production servers to a single reporting server also can reduce the complexity of reporting.

Each production server can be assigned a two-character identifier in the BaseLogID. All production servers would have unique transaction identifiers, making it possible to determine which production server handled a transaction.

CHAPTER 10

Purging and Archiving Data

With ongoing InTrack operation, database tables will continue to increase in size. The tables may be storing historical data on activity much longer than needed for practical usage. Excessive storage of large amounts of data will require considerable system space.

Information on shipped lots, for which all quantity is reduced to zero, is generally not needed beyond a certain timeframe, depending on the type of historical data storage needed for your product or industry. Historical activity data usually can be removed safely from the system after a period of time.

Two options are available to remove obsolete or unneeded history of transactions:

- **Purge**
  Purge removes information from database tables, as specified by timeframe and the type of data to be deleted.
- **Archive**
  Archive writes data, specified by timeframe or type, to a condensed text file that can be retrieved for future storage if needed. After writing the specified data to the text file, the purge function is performed.

These procedures can be scheduled routinely as a server task or can be performed manually as needed.

Both are available using a specialized utility program or OLE automation methods.

This chapter provides basic information on purging obsolete data. Refer to the InTrack Runtime Development Guide or the OLE Reference Guide for full information on using the purge utility program or OLE methods.

Contents

- Purging, 2
- Archiving, 3
- Database Concerns, 4

Purging

Purging of InTrack data can be performed with the interactive Purge.exe program or through the OLE automation interface. Both use the same options.

You can schedule a specified node to purge data automatically at a set time interval, or maintain the system with manual purges as needed. The method used depends on the timeliness of data, quantity of data collected, and the need to maintain extended historical records.

InTrack tracks material from receiving to shipping. After a Lot completely ships, the primary quantity will be zero. When the lot quantity reaches zero, all entries in the Sublot table are removed, the PrimaryQty field in the Lot table will be set to zero, and the ZeroDate field in the Lot table will be set to the ship date of the final quantity.

Purge Options

The standard method for purging shipped lots, %Database.PurgeLot(%TimeInterval, PurgeLots, PurgeLogs, PurgeData), allows you to specify a time period so that only lots shipped before a specified past date are purged. The purge lots parameter can be set to delete the Lot and Sublot table entries. The purge logs parameter can be set to delete associated entries in the log tables. The purge data parameter can be set to remove data collection tables associated with the lots.

To delete a single lot with a primary quantity of zero, use the %Lot.Delete() method.

To delete only machine information, use the %Database.PurgeMachine() method.

To remove data from dataset templates that are not associated with a machine or lot, use the %Database.PurgeDataSet(%TimeInterval) or %Database.PurgeDataTable(dataSetTemplate, %TimeInterval) methods.

Purging Unshipped Lots

A purge method enables you to selectively delete lots that do not have a zero quantity.

The %Database.PurgeLotByUserQuery() method accepts a comparison statement that is appended to the “SELECT DISTINCT Lot.LotID FROM Lot” SQL statement.

You can query any column in the Lot table, including UDAs. Refer to the following section on Database concerns when using this option.

Archiving

Archiving is an enhanced purge method that writes data from any InTrack table to a text file. Archiving can be used to delete database records after they are written to the file.

Archiving handles a set of tables as a group. The group may contain a single table or multiple tables. When archiving, a directory location is assigned to indicate where archived files (one per table) will reside. Additional options allow you to query and retrieve a subset of the data.

If the Purge property is set, the data will be deleted after archiving. The OLE automation command for archiving is %Database.Archive(%ArchiveItem(s)).

The Archive item method also allows you to set a percent idle time to execute all the necessary commands in series with no delay. If the idle time is increased, InTrack will pause between command sets to achieve the targeted percent idle time. For example, if a target idle time of 50% is entered and it takes two seconds to delete a lot, then InTrack will wait two seconds before deleting the second lot. The actual idle time is monitored so that adjustments can be made as more lots are deleted.

One drawback to the ArchiveItem object is that there is a separate class for Lots, Lot logs, and Runtime Data collection. Consequently, three commands are needed to delete all information associated with an expired lot instead of a single purge command.

Database Concerns

Performance

The amount of time it takes to delete records from the database can be substantial. Use the following suggestions to reduce the purge time.

If the purging process is lengthy, you can identify the problem using SQL Trace for MS SQL Server. Capture a trace file of a client connection during the purge and review it. Look for queries that require the most time to deletion a single lot. The following recommendations will apply to most situations.

Using the standard query on the ZeroDate will take many seconds if there are thousands of records in the Lot table. Reduce the time by adding an index on the ZeroDate column of the Lot table. This will increase the insertion time slightly, but will reduce the time needed for purging significantly.

If there are many UDAs on the Lot or Sublot tables being set, a large number of entries will be made in the AdjustLog. The AdjustLog entries are created when a property of the Lot changes outside the model using one of the Set methods, or when a UDA value is saved to the database. Logging of UDA entries in the AdjustLog can be deactivated. If you are certain you will not need to undo setting a UDA value and will not longer need the history of all UDA values for a Lot or Sublot, turn off UDA logging by setting the %Database.UDALogFlag = 1.

If using the PurgeByUserQuery method, examine the where expression used. If the column(s) in the where expression is not indexed, adding an index to the column in the Lot table will improve retrieval speed. This will help for a Lot table with many records.

If log tables require a long time to delete, add an index to the LotBaseLog for the combination of LotID and TransLogIDType. However, the index should be added only if all other attempts to shorten the purging time are not successful.

To ensure that indexes are as efficient as possible, you can create a scheduled task to update the statistics on the Lot, Sublot, and LotBaseLog tables. The syntax for this is UPDATE STATISTICS LotBaseLog. Because this may take a while when there are many records in the table, so it is advisable to run this only before or after a purging cycle.

With SQL Server 7.0 and later, it is preferable to enable automatic updating of statistics on the entire database.

Database Size

A sizing document (DBSize.xls) is available that assists one in predicting the amount of space that the database will require. This spreadsheet is useful in generating an estimate of storage requirements. However, it is only an estimate and actual tests should be used for verification. Large amounts of data collection can increase database size up to 25%.

If testing reveals a significant deviation from the estimated size, an application review is needed.

Review the Performance Considerations document on the Knowledge Base CD to determine if the model is optimized. Look for an implementation that queues a large quantity of a sublot at an operation but completes only single quantities to the next step. If the step at which the sublot is queued also requires consumption or data collection, there may be excessive data storage.

In InTrack 7.0, every consume transaction was stored in the SublotConsumption table. If consumption is performed one sublot unit at a time, many records are created in this table. The number of records in the SublotConsumption table associated with the sublot continues to increase as more consumption is done against the sublot. When a complete command is issued, all these records are copied to the CompleteLogConsumption table. In this arrangement, the CompleteLogConsumption table will become the largest table in the database. Records in the SublotConsumption table are not created when material consumption is modeled with a 100% lower tolerance. No consumption of the material is required, so InTrack will not track it.

For InTrack 7.1 or greater, the consumption bookkeeping model has been updated to store fewer records in the renamed LotConsumption table. InTrack updates existing records in this table for each consume and complete transaction instead of inserting new records, so fewer records are added to the CompleteLogConsumption table.

Two methods can help keep table size to a minimum.

A staging operation can be added before the consumption step. If possible, create the large sublot at the staging operation, defined as “Complete only” and “allow flow lots”. Complete a quantity of 1 to the second step, start the unit (or use complete only if machine tracking or user certification is not required), consume the necessary items, and complete the single unit. In this method, only one record will be copied to the CompleteLogConsumption table when the single unit is completed instead of an increasing number of records.

If a staging step is not possible, you can create a task on SQL Server to truncate the CompleteLogConsumption table once or twice a day as needed. Use the truncate command and not the delete command for faster performance. Truncating the table is recommended only if undoing the complete transaction will not be necessary. After records are removed, it is not possible to undo the complete transaction. There is no loss of reporting information by deleting the CompleteLogConsumption table. The data in the CompleteLogConsumption table is only used internally for the undo transaction, and all necessary historical information can still be found in the LotBaseLog and ConsumeLog tables.

The same procedures also apply to the CloseLogConsumption, CompleteLogCollection and CloseLogCollection tables.

CHAPTER 11

Terminal Server Architecture

With the introduction of InTrack 7.11, it is possible to deploy a solution using Microsoft Windows Terminal Server.

Terminal Server service allows relatively low-end computers (clients) to access centrally configured software installed on a server and use it as if it was installed on the client computers.

While the Terminal Server must be a high-end computer, the clients can be a wide variety of computers and operating systems. The Terminal Server client must be installed on any computer using the software on the server. Or, a client can access software on the Terminal Server using the Advanced Client (TSAC) control that is accessible via the web.

The supported operating systems for InTrack clients using Terminal Server are listed below:

- Windows 2000
- Windows NT 3.51 or later
- Windows 98 (via TSAC)
- Windows 95 (via TSAC)
- Windows for Workgroups 3.11 (via TSAC)
- Windows CE-based Terminals (not recommended for InTrack)
- Unix, Linux, etc. (Supplied by Citrix systems. (www.citrix.com) These clients have not been tested by Wonderware.)

NOTE: InTrack's Model Maker program is not supported using a Terminal Server configuration. In order to use Model Maker you MUST either run Model Maker on a computer that has InTrack installed or on the Terminal Server itself. You must use the Windows Install/Uninstall option to install InTrack

---

The advantages of using Terminal Server for a plant wide solution are listed below:

- It is maintained more easily. Because the software used is deployed on a central server, upgrades to the software involve only a handful of servers. The clients will not need upgrading.
- It is more scaleable. Because clients do not need software installed, they can be added and removed easily.
- It is more cost effective for large solutions. Because clients can run on a large number of machines, old computers can be used and new computers do not have to be high-end machines.

The disadvantages of using Terminal Server for a plant wide solution are listed below:

- Clients may run more slowly. Because the client software is running on a server machine and many other clients may be running as well, client performance can be slower.
- It can be costly for small solutions. Because the terminal server must be a very high-end machine and because every machine not running Microsoft Windows 2000 must buy a separate Terminal Server license, small installations can be more costly.
- Solutions must be tested. The proposed solution using Terminal Services must be configured and more thoroughly tested than a normal solution because of server loading and client responsiveness.

Special Notes on Terminal Service Environments

Archiving must be performed only on the TSE server.

In a terminal services environment, archiving should be performed only on the server.

If archiving is performed on TSE clients independently, an archiving operation on one client could potentially append data to an archive performed by another client at the same time. Avoid this risk by performing archiving operations exclusively from the server.

Installations must be performed using Add/Remove Programs.

When running InTrack on the Terminal Services Environment, the Windows Add/Remove Programs function must be used when installing or uninstalling InTrack.

Different log file names are required when running the trace utility on multiple nodes simultaneously.

The ITRTrace utility can operate on multiple nodes by different users. However, the same user should not attempt to run ITRTrace on multiple nodes simultaneously.

When the ITRTrace is performed, data is written to the log file (init.log), which is unique for each user ID. The same log file will be used and data for trace activities on one node will be overwritten by data from another node.

To run multiple trace sessions in a TSE environment, the same user ID can be used only if the log file is renamed individually for each session.

Sizing the Terminal Server

Benchmark tests have been performed to help guide selection of processor speed and memory.

Test Hardware and Software

The test environment consisted of a single TSE server, 35 clients, an SQL database server, and a 10Mbps Ethernet hub

Hardware

Client

Dell OptiPlex GX110 667Mhz 256MB

Intel 810 graphics controller video card

3Com 3C920 Integrated Fast Ethernet Controller

TSE Server

Dell PowerEdge 866Mhz 2GB

ATI 3D Rage IIC PCI

Intel 8255x-based Integrated Fast Ethernet Controller

Database Server

Dell PowerEdge 550Mhz 512MB

Intel PRO/100+ Integrated Fast Ethernet Controller PCI

ATI 3D Rage IIC PCI

Network

NetGear EN104TP 10Mbps Ethernet hub

Software

InTrack 7.11 (7,11,0,129)

InTouch 7.11 (Build 2001-08-30-f-TSE)

Visual Basic 6.0 (8176)

WinNT 4.0 SP5 – Database Server

Win2000 SP2 – TSE Server

Custom Logging Tools

InTrack Logger (1.01)

Logger (1.00)

Timer Log (1.00)

Client Startup

Clients were started sequentially at 1-minute intervals until the total number of clients was started or a performance threshold was exceeded. Each client, upon startup was logged by name and start time to allow for distinction of transaction data. The process was as follows:

Start application

Configure logging parameters (Name, Logging Server/Database, Time Interval between transactions)

Connect to logging database to establish baseline

Connect to InTrack

Select sequential test parameters

Start the test (includes the following transactions)

Sublot Create

Sublot Start

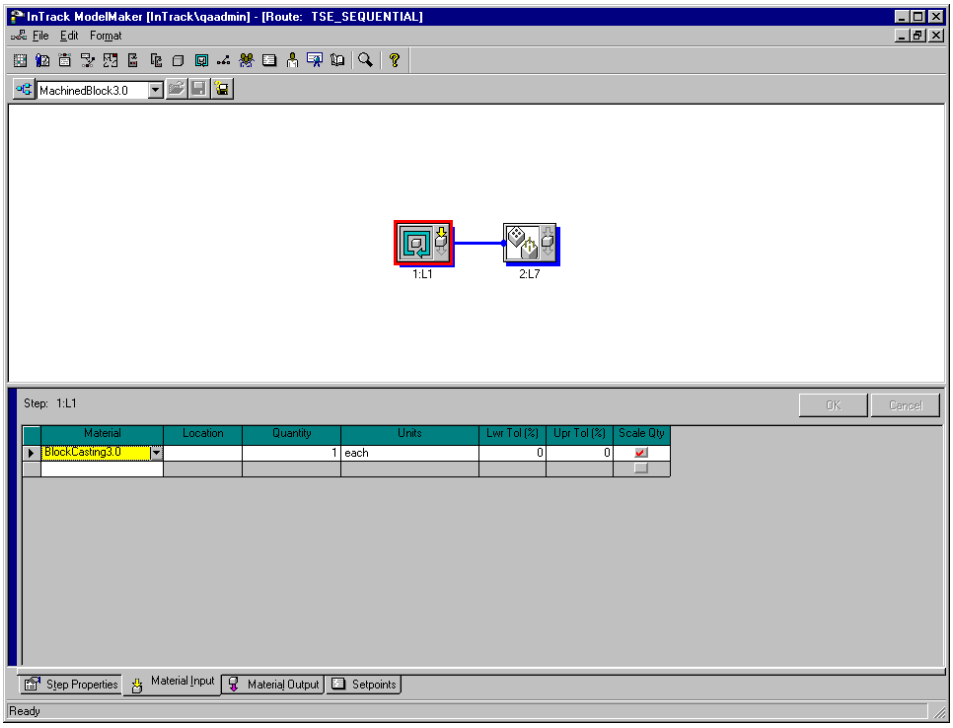
Sublot Data Collection

Sublot Consumption

Sublot Complete

Sublot Close

These transactions were processed on a 2-step route with a single input material defined at step #1. Between transactions, in addition to the logging, a 1 second delay was allowed.



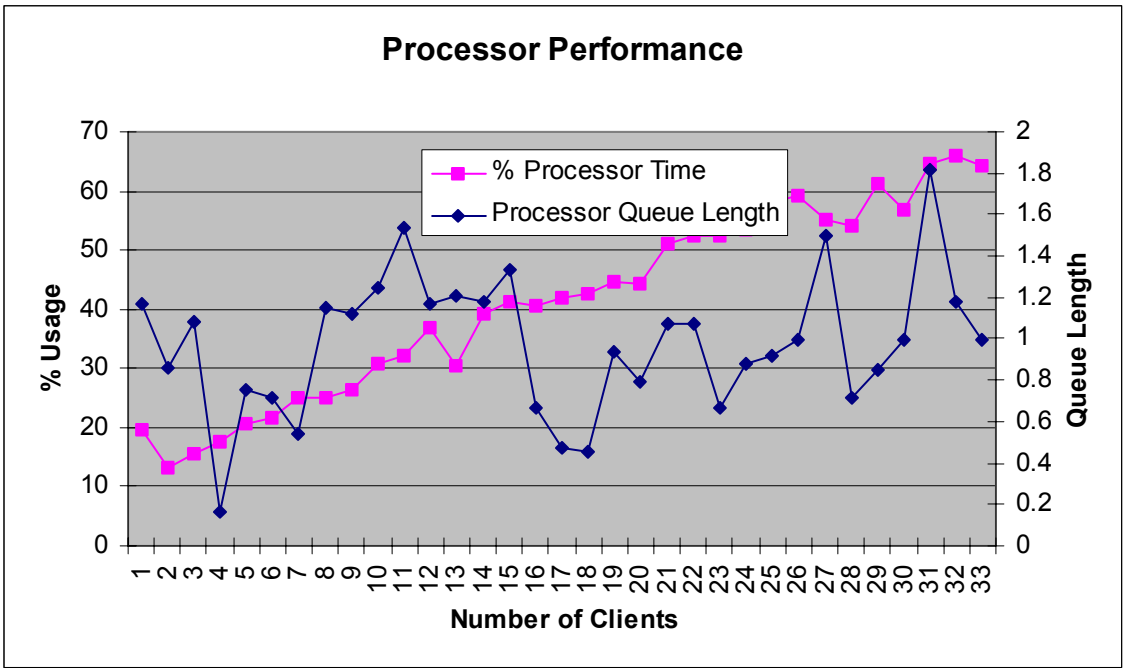
Performance Measurements

Processor Performance

The % processor usage is a primary indicator of processor activity measuring the percentage of time the processor spends executing useful work. The queue length provides an indication of the number of threads in the processor queue waiting to be processed. A queue length greater than 2 is considered to be excessive.

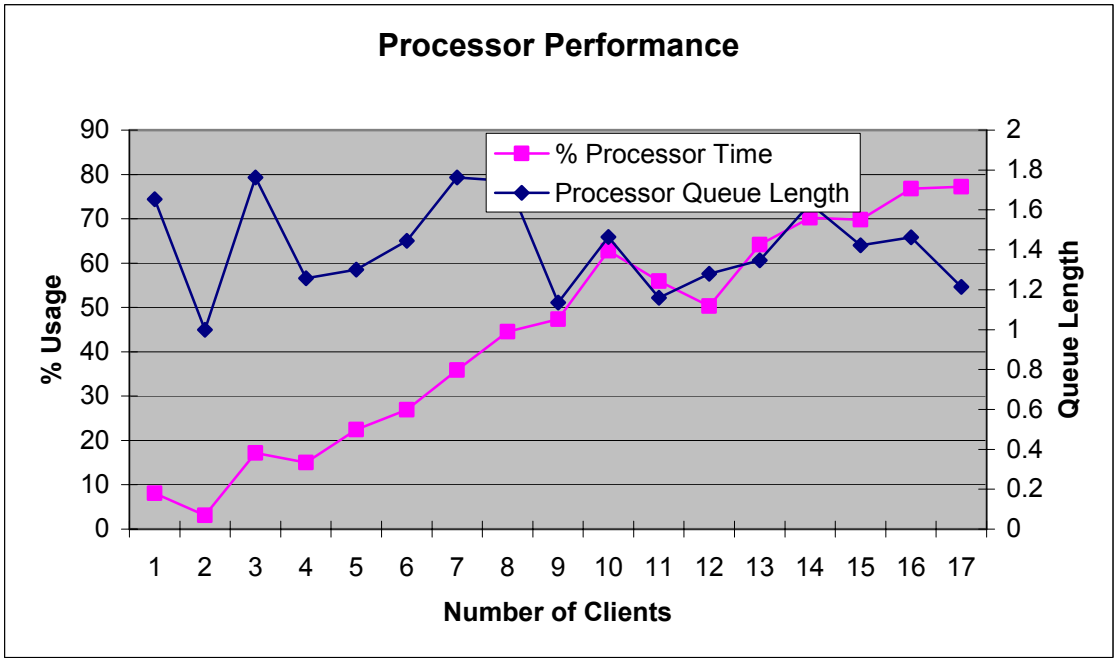
Processor threshold (> 60%) was reached at approximately 31 **Visual Basic** clients however the queue length did not exceed 2. The average InTrack database transaction rate at this threshold was 25/sec.

Visual Basic Client



Processor threshold (> 60%) was reached at approximately 15 InTouch clients and again, the queue length did not exceed 2. The average InTrack database transaction rate at this threshold was 15/sec.

InTouch Client

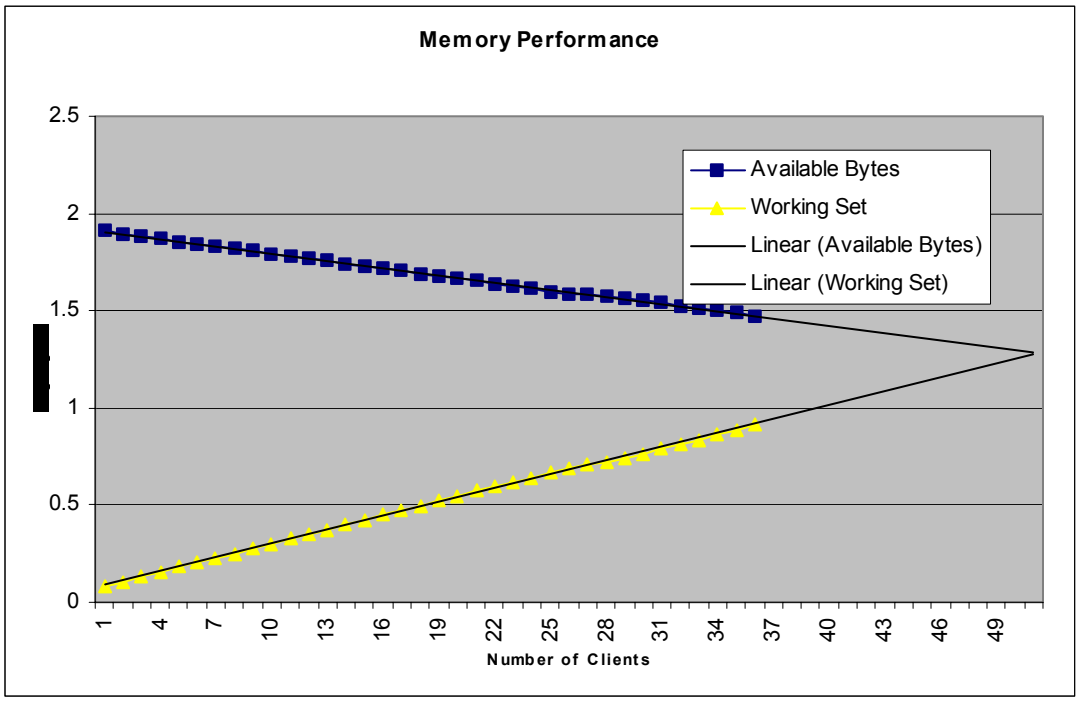


Memory Performance

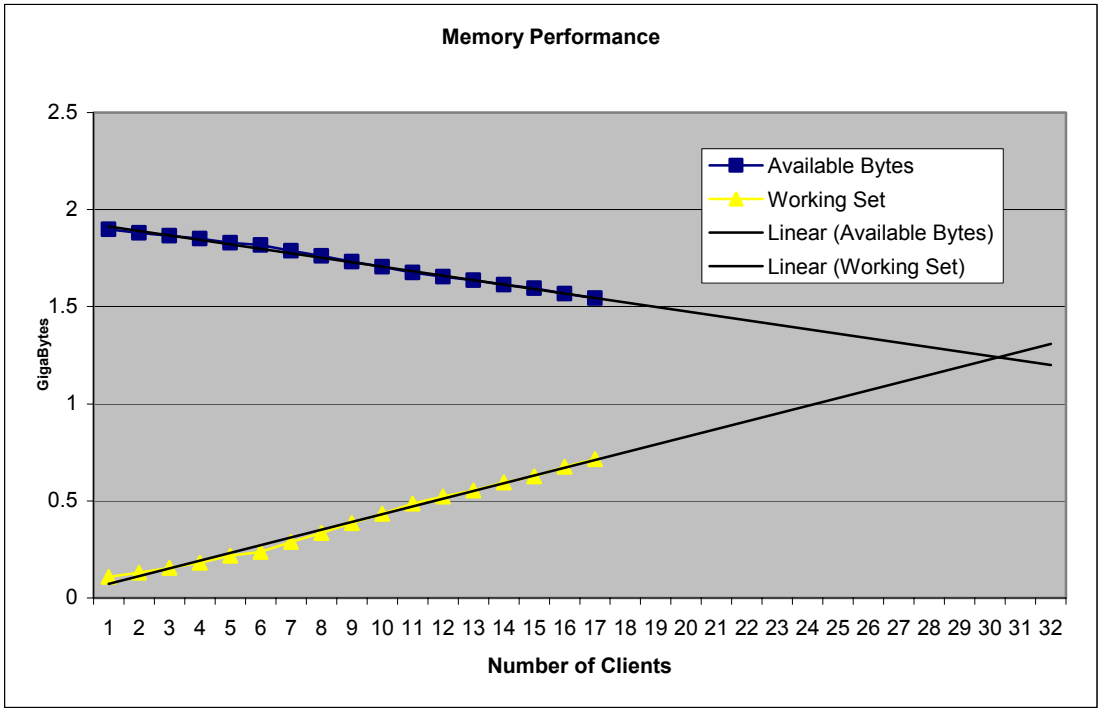
In a single-user operating system, each InTrack application consumed memory for the executable image and the dynamic stack and heap. Code page sharing, which is supported in TSE, reduces total memory usage by backing only a single copy of the code text and mapping it read-only into the address space of each application instance. Therefore, memory is consumed by applications in TSE in two ways. First, each new instance of an application consumes at least as much memory as the size of its stack and heap. Second, the greater the number of distinct applications used, the greater the combined memory usage for all of the applications' shared code pages. The available bytes counter displays the amount of physical memory (in bytes) available to processes running on the server. The most prevalent indication of memory limitation, to the point of performance degradation is the level at which available bytes crosses the working set slope.

A linear estimation was calculated based on actual samples that were collected. Based on this estimate, a total of 51 ***Visual Basic*** clients could operate given the test hardware configuration. A total of 30 ***InTouch*** clients could operate given the test hardware configuration.

Visual Basic Client



InTouch Client

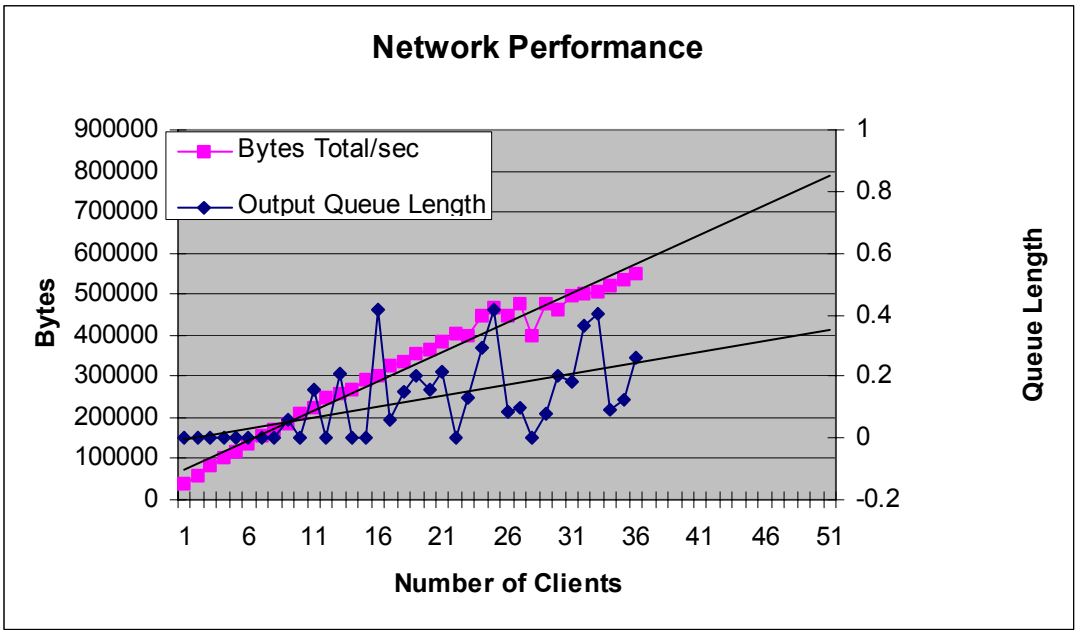


Network Performance

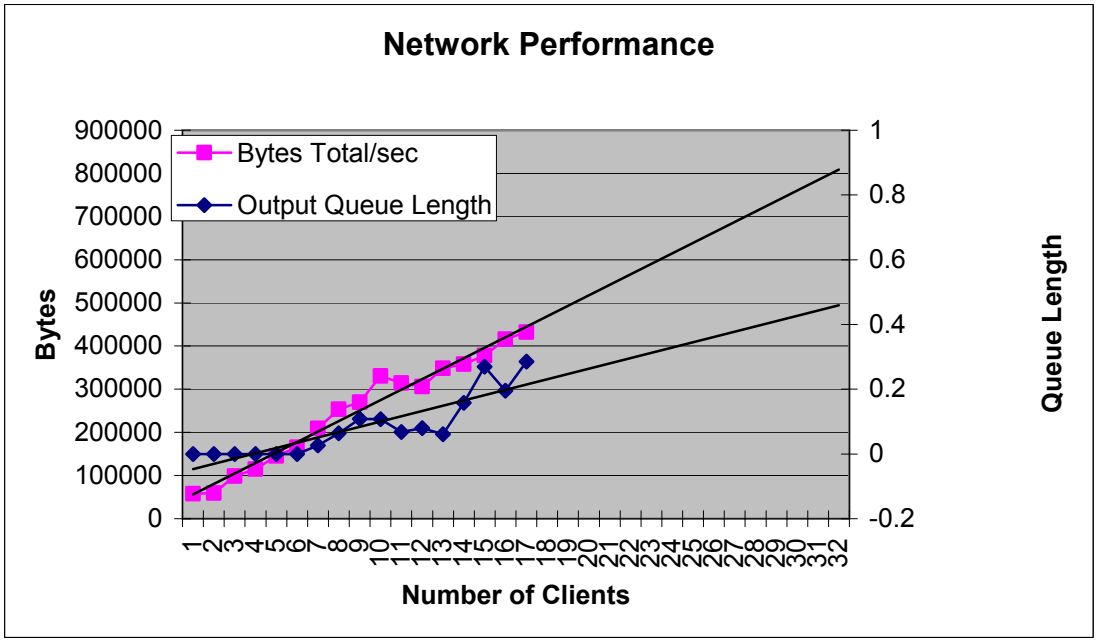
Unacceptable propagation delays or long network queue lengths indicate that the environment will suffer from network bottlenecks. Network utilization for the test scenario is shown below. This includes all traffic into and out of the terminal server for this test. As indicated in the result data, network utilization tends to be quite low on Terminal Services. This is due to Protocol efficiency and because the default setting of the Terminal Services Client (mstsc.exe) is to use data compression for all connections. Output queue length is the length of the output packet queue. If the length is greater than 2, this indicates that delays will be experienced. Therefore the bottleneck should be identified and eliminated. Current bandwidth is the estimate of the network interface's current bandwidth. The value is the nominal bandwidth and for this test was measured at 100000000.

The queue length in the ***Visual Basic*** and ***InTouch*** clients never exceeded a length of 2.

Visual Basic Client



InTouch Client



Transactional Measurements

User Response Times and Record Locking

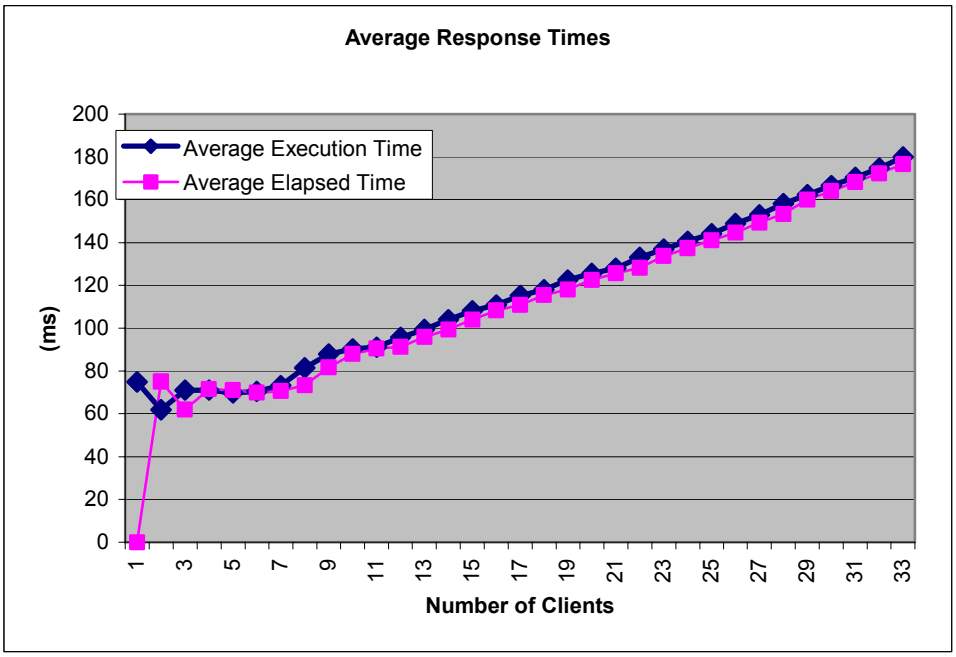
Response times were measured to estimate the amount of delay that can be expected for each transaction when running in a TSE environment. The times represent the amount of time required to make a successful database transaction. In addition, the elapsed time spent in each of the method areas provides the basis for beginning the tuning of an application. The findings indicate the SQL transaction rates were conditional not on the TSE configuration, but specific measures used to optimize InTrack performance.

The total elapsed time was performed using Visual Basic only.

Average Response Times

The average response time for the Visual Basic and InTouch clients, running a sequential process at a max transaction rate of 25/sec is shown in the graphs below. As indicated, the average time never exceeded 200ms.

Visual Basic Client

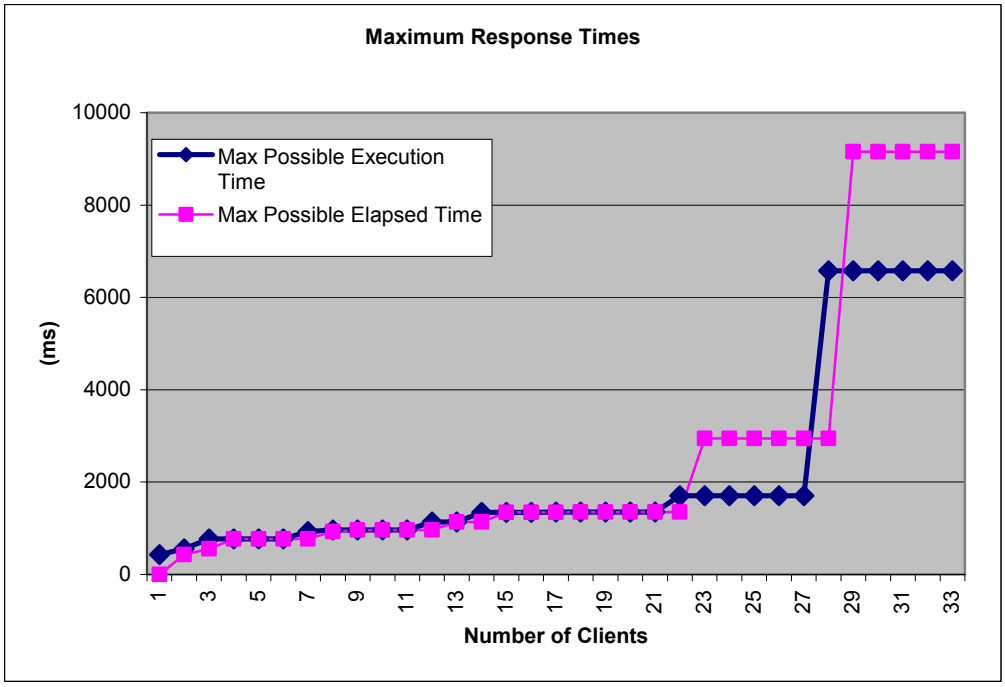


InTouch Client

Maximum Response Times

The maximum response and total execution time for a Visual Basic client, running a sequential process at a max transaction rate of 25/sec is shown in the graphs below. This shows that there is a possibility that operating more than 22 clients has the potential to cause a suspension time from when the operator clicks a button until a response is transmitted back, of greater than 2 seconds.

Visual Basic Client



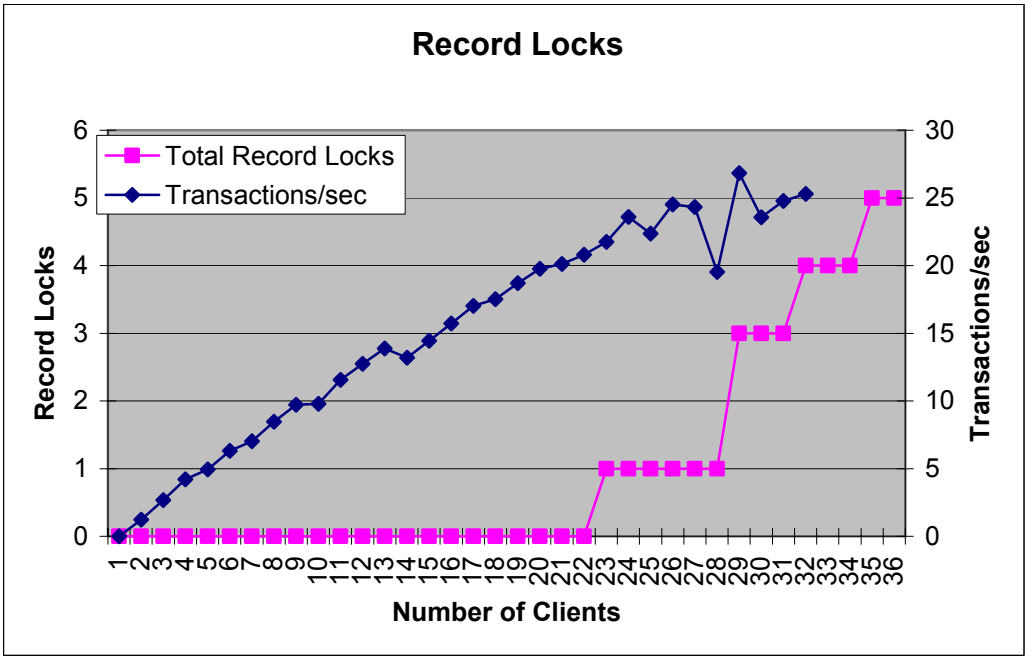
Record Locking

Deadlocking occurs when two user processes have locks on separate objects and each process is trying to acquire a lock on the object that the other process has. When this happens, SQL Server ends the deadlock by automatically choosing one and aborting the process, allowing the other process to continue. The aborted transaction is rolled back and an error message is sent to the user of the aborted process. This 1205 error is then sent to the client application causing the client to retry the method.

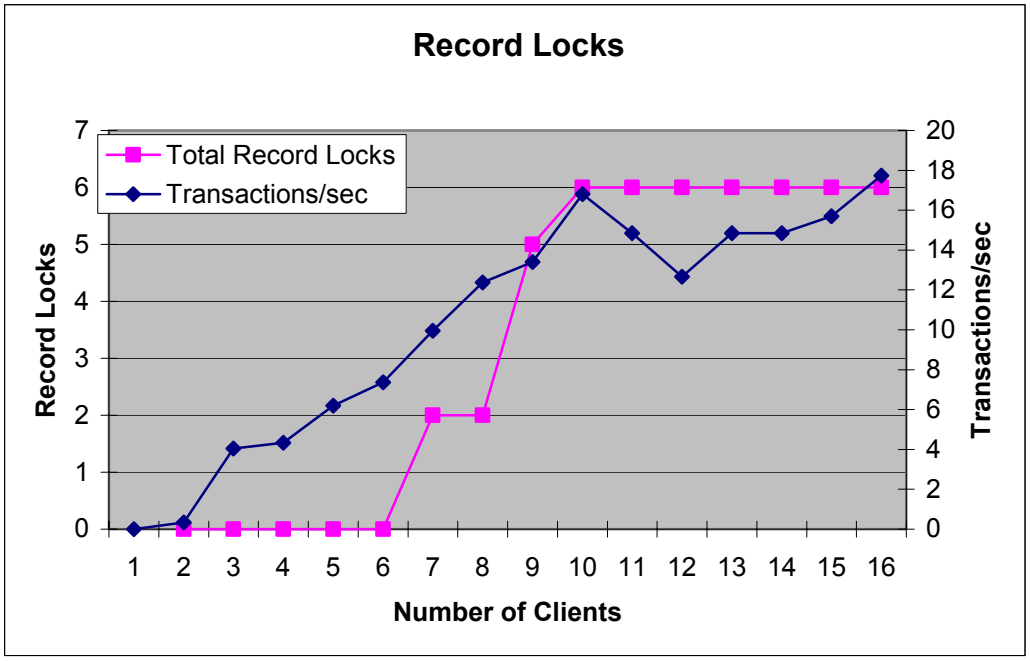
As indicated by the data, total record lock count, for 35 **Visual Basic** clients was 5. This count is a sum of the total record locks beginning from the time the first client was started until the last was started and process was subsequently shut down.

The total record lock count, for 16 **InTouch** clients was 6. This count is a sum of the total record locks beginning from the time the first client was started until the last was started and process was subsequently shut down.

Visual Basic Client



InTouch Client



Conclusion

Sizing a TSE Server InTrack solution presents complex challenges. Due to the variety of InTrack application mixes, measurements represented here may differ from solution to solution. The test model selected was to establish a benchmark and is not meant to cover all possible applications of InTrack. After sizing a solution, performance can be optimized by tuning the operating system, the registry, and most importantly the InTrack scripting.

Given the specifics of this test, the following generalizations can be made:

- A simple InTrack application using InTouch requires 50% more memory and utilizes twice as much processor time than using Visual Basic. This results in half as many InTouch clients able to connect to the same hardware configuration.
- Additional memory and additional processors (or faster processors) on the Terminal Server will increase the number of clients that will be able to connect.
- The terminal server and the database server must reside on separate systems.
- Although the processor time of the database server is not shown here, a balance must be maintained between the terminal server and the database server to ensure both are equally loaded. Otherwise, one of them will become the bottleneck and will require additional processors or memory. As the total number of transactions being executed continues to increase with a fixed number of clients, the database server will become the bottleneck and not the terminal server.

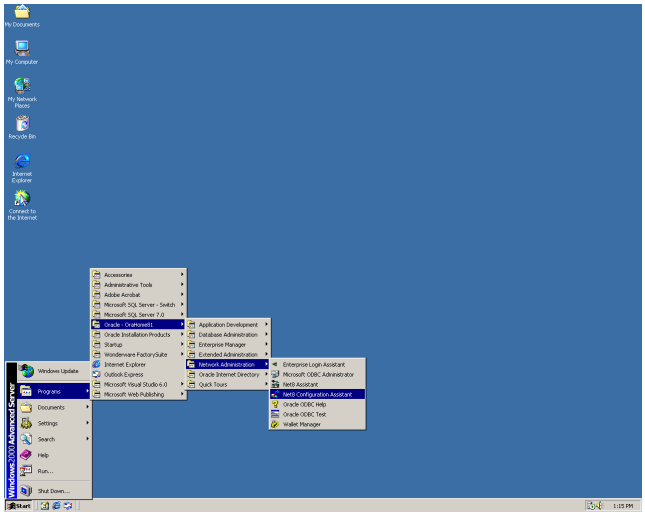
In summary, this test made some assumptions to assist one in determining the sizing requirements of a Terminal Server environment. The client transaction rates were artificially set at one transaction per second giving the system a very steady (and somewhat high) transaction load. In a production environment, the load will not be so steady and may be dramatically lower or higher. Do not make the assumption that only 30 VB clients can be used on a single terminal server. The exact number is dependent on the transaction rates of the clients, the complexity of the application (memory footprint), and the hardware configuration of the server. In a manual environment where the transaction rates are much less than one per second, one hundred clients could be maintained on a single terminal server with the addition of enough memory to load the specific application.

CHAPTER 12

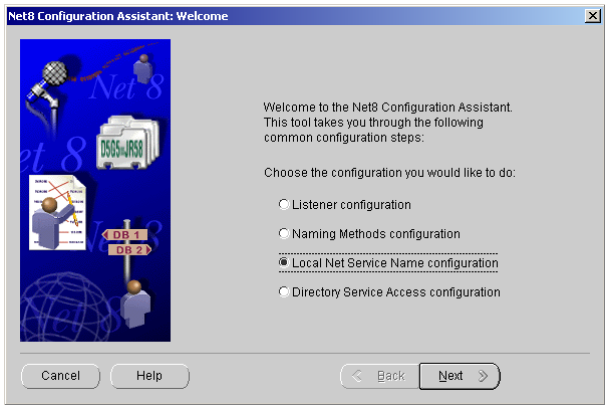
Connecting to an Oracle Database

Special procedures are required to connect to an Oracle Database:

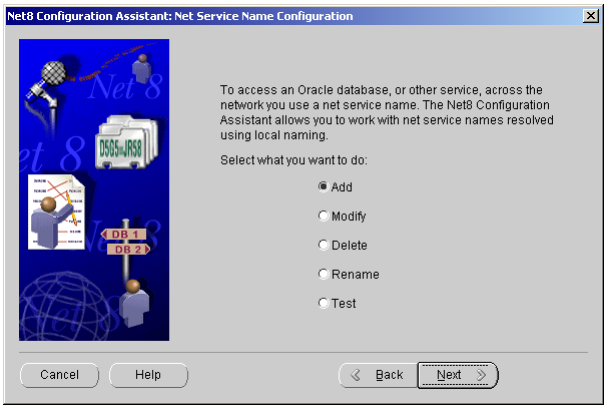
1. After installing Oracle, from the Start menu, select Programs/Oracle (OraHome81)/Network Administration/Net Configuration Assistant.



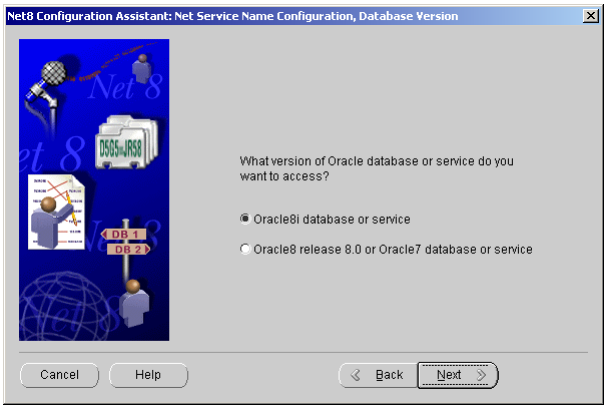
2. Select Local Net Service Name Configuration and click Next.



3. Select the Add option to add a local net service name or the Modify option to modify an existing configuration. Click Next.

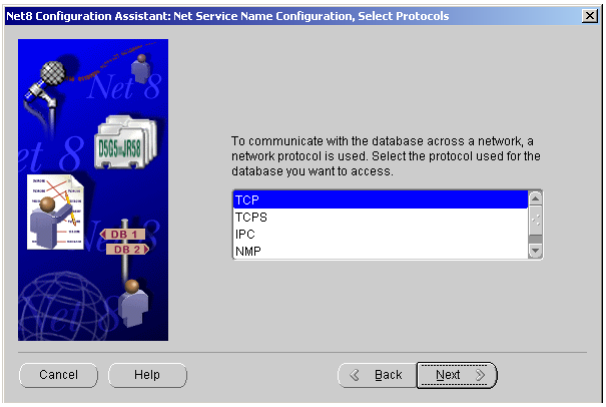


4. Select the Oracle8i database option to connect to an Oracle 8i database, or the Oracle8/Oracle7 option to connect to Oracle release 8.05. Click Next.

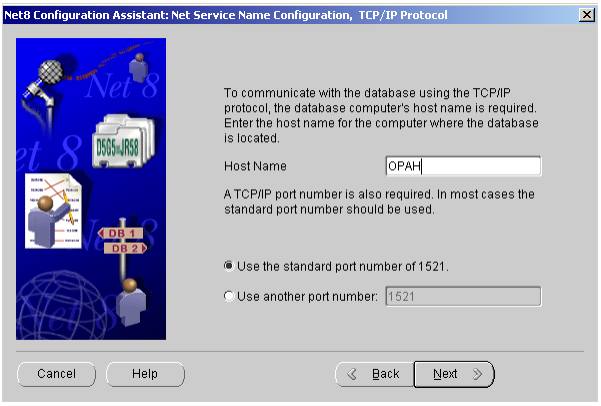


5. Enter the service name of the target Oracle database. This name is usually the same as the SID identifier assigned when the database was created. Click Next.

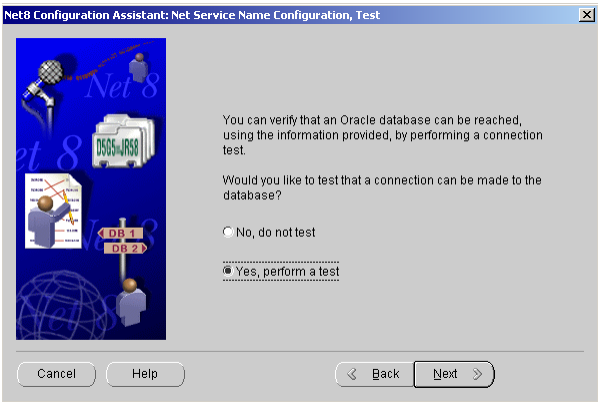
6. Select the network protocol type. For additional help contact your network administrator. Click Next.



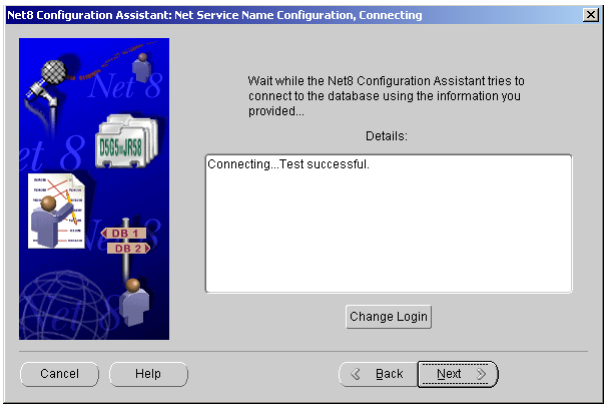
7. If TCP was selected on the previous screen, enter the host name of the computer where the database is located. The standard port number is usually 1521. Click Next.



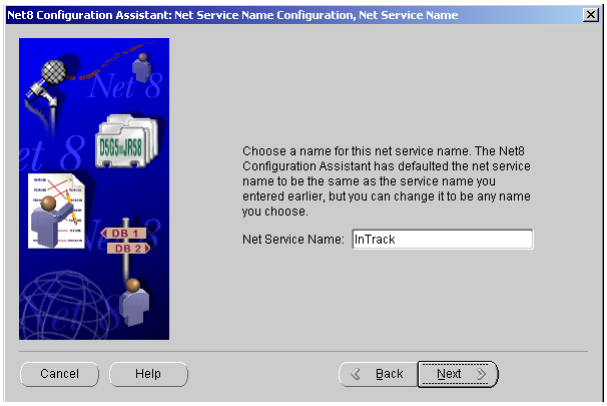
8. Even though the next step is not necessary, it is generally a good practice to perform a test to ensure that the service name connection was configured correctly. Click the next button to proceed.



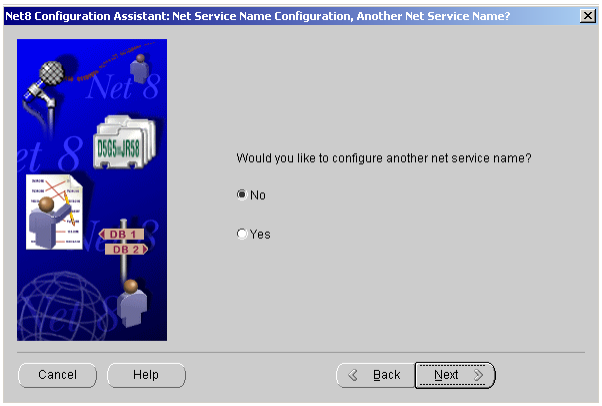
9. Results are displayed indicating whether or not the test was successful. Depending on the database security configuration, a different user ID and password may be required. Click Next.



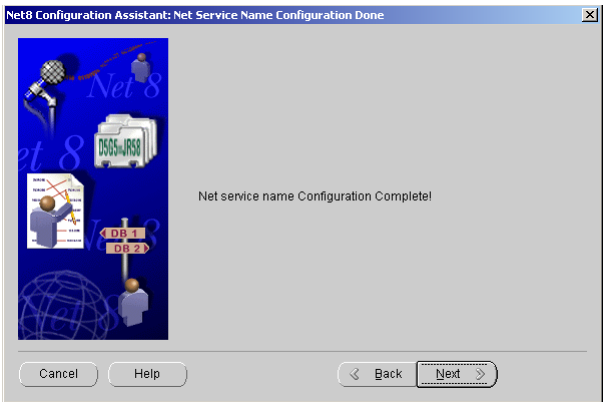
10. Enter the net service name for this configuration. This name will be used for access to the database using Oracle tools as well as making a connection via InTrack database setup. Click the next button to proceed.



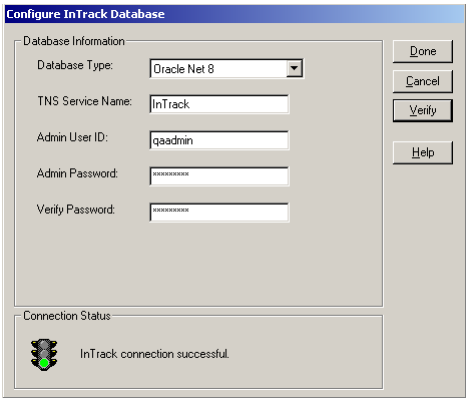
11. Unless multiple names are required, click No to proceed to the final screen. Click Next to proceed.



12. The system will indicate configuration is complete. Click Next to proceed.



13. When using the InTrack database setup utility to connect to an Oracle database, select Oracle Net 8 as the database type. This can be used to connect to Oracle database versions 7.x as well as 8.x. Enter the net service name provided in Step #10 in the TNS Service Name textbox. Enter the user ID and password associated with the Oracle tablespace within the target Oracle database.



Glossary of Terms

Active — The status of a WIP lot, inventory lot, or bulk inventory that releases it from quarantine, enabling it to be shipped or consumed. *See also* quarantined.

Activity object — Objects created and manipulated at runtime via transactions, or OLE scripts. Transactions can be issued either manually by an operator or automatically by the InTrack system. Activity objects are dynamic components of the manufacturing model in that they can move between the different locations or operations of a manufacturing route. *See also* structural objects.

ActiveX — Microsoft's brand name for the technologies that enable interoperability using the Component Object Model (COM). ActiveX technology includes, but is not limited to, OLE.

Adjust — A WIP lot, inventory lot, or bulk inventory transaction that changes any attribute. Adjust is typically used to change the quantity, priority, or due date.

Alternate path — A path that does not follow along the main exits of a route toward the ensuing destination steps, but along side exits. A route can have more than one alternate path.

Archive — The process of extracting data from the InTrack RDBMS and saving it in operating system files for long term storage.

Backup — A set of copied data, reserved in case of a system failure.

Bill of material (BOM) — A list of all parts, components, and quantities of material needed to produce the target product (input), and any coproducts or by-products produced during the manufacture of the target product (output).

Bitmap — A graphic image, such as an icon, used to represent a route operation in the Graphical view. Bitmaps for a particular operation can be changed by selecting a different bitmap or by modifying the existing one.

BOM — *See* bill of material.

Boolean — A data type with logical (TRUE, FALSE) values.

Bulk inventory — A quantity of material that is not assigned or tracked by a lot number. Bulk inventory is tracked by storage location.

Bulk Inventory Selector A runtime display object into the InTrack database tables used by the operator for selecting bulk inventories to work on. *See also* WIP Selector, Inventory Lot Selector.

Button wizard A unique, preprogrammed combination of actions that can create or manipulate InTrack objects. Button wizards issue OLE script for the currently selected item in the appropriate selector.

By-product An output material of little or no value that is an inevitable result in manufacturing; i.e., waste. A by-product is not included in the primary quantity of the final product produced on a route. Since production of by-products reduces the WIP lot quantity, they are typically minimized in manufacturing and are tracked separately. *See also* coproduct.

Cached objects A list of objects stored locally in each workstation (node) on a multiple-user database. Cached objects are continually updated from the database when new objects are defined or when an object definition is saved or restored.

Calendar An object name that defines when one or more resources are available.

Client A user workstation containing the InTrack user interface and programs.

Client/server An architecture of hardware and software where the client (a user or program) makes requests (to the server) for resources or information.

Close A WIP lot transaction that completes the last operation in a route and moves (closes) the WIP lot to inventory.

Collect A WIP lot transaction that collects engineering data along a route to provide in process quality checks for tracking, analysis, and reporting. Information collected during runtime is entered in dataset templates that were defined in ModelMaker.

Column *See* field.

Comma Separated Variable (.CSV) A format used by the clipboard for transfer of text columns and numerical data between applications. A .CSV data item is similar to text, with each variable separated by a comma. Although Microsoft Excel is probably the principal creator of .CSV clipboard data, many DOS applications support this format.

Comment A WIP lot, inventory lot, or bulk inventory transaction that allows the operator to log comments. The comments are stored in the InTrack database.

Complete A WIP lot transaction that moves a WIP lot to the next operation. Issuing this transaction changes a WIP lot's status to "queued."

Configure A WIP lot, inventory lot, or bulk inventory transaction that allows only lots or inventory with certain parameters to be displayed in the appropriate selector.

Configure to order Deviation from a standard way of manufacturing a product to meet specific customer requirements.

Connect A system transaction that establishes a connection to the database.

Connection The graphical line connecting a route step with the next route step (the destination route step). Each connection has a direction indicator to indicate the path of the route. A main exit connection is centered on any side of a route step, and a side exit connection is offset from the center on any side. *See also* main exit, side exit.

Consume A WIP lot transaction that controls and tracks the quantities of input material consumed at an operation.

Consumption *See* material consumption.

Coproduct A valuable output material, but not the final product produced on a route. As coproducts are produced, they are moved off the route, given a different lot number, and are tracked separately on another route. A coproduct does not reduce the quantity of the WIP lot from which it originates. *See also* by-product.

Create A WIP lot transaction that creates a new WIP lot and queues it at the initial step on a manufacturing route.

Customer specifications Special customer requirements for producing a product. Customer specifications are object names referencing a specific set of changes to an operation's attributes, called overrides. *See also* override.

Customization The modifying of a system to meet the specific needs of a user.

Cycle time The processing time necessary for completing an operation.

Data Information associated with a manufacturing system.

Data collection The process of acquiring meaningful data for work in process, usually used for analysis and record keeping. Data is typically collected when an operation is executed and can be alphanumeric, analog, or Boolean. Data is collected in groups controlled by dataset templates, which are defined in ModelMaker.

Dataset template The definition of a group of data items to be collected together at runtime; a form defined in ModelMaker that the operator uses to enter runtime data. When a dataset template is defined and saved, a new runtime table is created to store collected datasets. *See also* dataset.

Dataset Data collected in a dataset template. *See also* dataset template.

Database A system depository of common types of data, sorted by unique identifiers, organized into tables. The InTrack database tables contain both the manufacturing model objects defined in ModelMaker and the lot transaction data created during runtime.

Detail (1) A WIP lot, inventory lot, or bulk inventory transaction that returns attribute information about the lot or inventory. (2) Attributes for a particular activity object. Object details include the defined characteristics of an object, such as the name of the object, the quantity, its quarantine status, any user-defined attributes, etc.

Destination Route Step From a route step, the next step on the route according to a specific disposition code.

Disassemble A WIP lot transaction that controls and tracks the quantities of input material previously consumed (assembled) at an operation that are removed from the assembly.

Disconnect A system transaction that terminates the current connection to the database.

Dispatching The assigning of priority ratings to lots queued at an operation. WIP lots with higher priority ratings usually are worked on before WIP lots with lower priority ratings. Priority ordering is typically based on the due dates or slack times of the WIP lots.

Disposition code A value representing the disposition of a WIP lot once it has completed an operation. The disposition code represents the condition under which quantities of the WIP lot move out of a route operation or are scrapped at a route step.

Due date The date the WIP lot is supposed to be completed at the last operation on a route. The WIP lot may be created using the default due date or a date entered manually by the operator. The default due date is calculated by adding the cumulative cycle times for the route operations to the current date.

Dynamic link library (DLL) A software module loaded into memory at execution time in order to access functions of that model.

Enterprise resource planning (ERP) Software that has expanded the functionality of a manufacturing resource planning (MRP II) system.

ERP *See* enterprise resource planning.

Event The activity in a system initiated by the user in order for processing to take place.

Field A column in a database table that describes one characteristic of the entity, such as a WIP lot's primary quantity or current operation, an inventory's location or area, or the date or time a transaction occurred. *See also* record, table.

Finished goods inventory (FGI) A collection of material representing the final product manufactured on a route.

Flow lot A WIP lot queued or in process at more than one operation simultaneously. The individual WIP lot quantities of a flow WIP lot are called sublots.

Function A procedure in programming language.

Graphical route editor (GRE) The visual display in ModelMaker; also known as the Graphical view.

Graphical user interface (GUI) The visual and functional operation of an application that provides the user with the means of interacting with the application at runtime.

Graphical view One of five views of the route document window that visually displays a route and is used for defining route steps. *See also* route views.

GUI *See* Graphical User Interface.

Hold *See* on hold.

Index An ordered list of data records (rows) that are returned as a result of a database query.

Initial step Typically the first step on the route. In runtime, new WIP lots are automatically queued at the initial route step to begin processing.

Input Data entered into the system by the user or another system, device, or program.

Input material The elements, constituents, or substances used for manufacturing a target product; defined as part of the bill of material for that product.

Input parameter Defines the values needed to complete a transaction (e.g., quantity to start, attribute to adjust).

Instruct A WIP lot feature that displays work instructions for a particular operation. *See also* work instructions.

Integer A whole number, such as 0, 50, or 764. Integers can be signed (positive or negative) or unsigned (positive) and are without specified value limits for data collected.

Integrator An application builder, or an application programmer who customizes a program by enhancing and building user interfaces so programs can communicate and share data.

Intermediate material Material that has exited an operation and waits in queue to enter the subsequent operation on a route.

Inventory A collection of material tracked by lot number or storage location. *See also* inventory lot, bulk inventory.

Inventory Lot A collection of material of the same type (product) stored in inventory and tracked by a lot number. *See also* bulk inventory.

Inventory Lot Selector A runtime display object into the InTrack database tables used by the operator for selecting lot inventories to work on. *See also* WIP Selector, Bulk Inventory Selector.

Key A field in the database, such as LotID, or MatlTypeName (material type name), used to locate activity objects. One or more keys constitute the object ID for the activity object. WIP lots, inventory lots, and bulk inventory all require a different set of keys in order to be identified by the InTrack system. *See also* object ID.

Laboratory information management system (LIMS) A system used for data collection in a laboratory environment.

LIMS *See* laboratory information management system.

Location A department or site (physical or logical) responsible for processing lots of work in process or inventory. Examples of locations are physical positions of machines, raw material inventory, and finished goods inventory. A location provides lot ID uniqueness; that is, the combination of location and lot ID must be unique.

Log Tables of data, categorized by common types of information, that reside in a database and use a time stamp as a primary key.

Lot A collection of one or more items of the same material type that are manufactured, stored, and tracked together. There are two types of lots - WIP and inventory.

Machine A device that performs some process on a lot at an operation. Once a machine is defined, you can associate it with one or more operations in the database.

Machine type An object used to group machines that have similar maintenance, failure, and repair characteristics. A machine type lets you update the failure symptoms and maintenance and repair tasks for all the machines in the group just once rather than individually for each machine.

Main exit An exit out of a route step that is associated with the default disposition code. The main exit connection is centered on any side of a route step.

Main path The primary sequence of operations a WIP lot will follow along a route. WIP lots that complete an operation and are assigned the default disposition code progress through the route along the main path. The main path is used in cumulative yield and cycle time calculations.

Maintenance The process of keeping system data current and usable.

Manufacturing Execution System (MES) A set of programs that performs the functionality of plant management, supervisory control, quality management, and plant engineering. MES provides a link between process control systems on the factory floor and the MRP II business system.

Manufacturing Resource Planning (MRP II) A set of programs that plans production to meet demand. MRP II, referred to as a "business system," is the outgrowth of material requirements planning (MRP).

Material The elements, constituents, or substances used for manufacturing a target product (input material), and the coproducts and by-products of the target product (output material).

Material consumption The process of using up input material during an operation.

Material input view One of five types of views in the route document window; used to assign consumable materials to route steps. *See also* route views.

Material output view One of five types of views in the route document window; used to assign by-products and coproducts to route steps. *See also* route views.

Material Requirements Planning (MRP) A system that collects data from other available manufacturing systems (inventory, BOM, WIP, scheduling) to compute material requirements based on sales forecasts.

Material type Identifies and describes items in raw inventory, intermediate material, WIP material, or finished goods; the type of material(s) used in manufacturing a WIP lot.

Merge A WIP or inventory lot transaction that combines two or more lots into one lot along a route. To merge, lots must be (1) queued at the same operation, (2) producing the same product, and (3) have the same status. The system maintains the genealogy of merged lots.

Model The functionality engine of InTrack. The model is a representation of a plant's actual manufacturing processes. The model includes static components, such as operations, and dynamic objects, such as work in process (WIP) lots. The model also includes: materials, the bill of material, work instructions, specifications for data collection, disposition codes for output materials, and locations.

Modeling The process of creating a computer model of plant floor operations. During the modeling process, the paths (routes) that move products through successive stages in manufacturing and the relationship between these routes are defined.

ModelMaker The development environment used to create the manufacturing model. *See also* runtime environment.

Move An inventory lot or bulk inventory transaction that transfers some or all of an inventory to an existing inventory in the same location, or moves the entire inventory lot into another location, storage location, or both. *See* Nonstandard move.

Network A communications infrastructure connecting a group of physically connected computers.

Nonstandard move A WIP lot transaction that changes the WIP lot position to a route step other than the next step on the main path, as defined by the disposition code. A nonstandard move can only be performed if the WIP lot is in queue at an operation.

Object An item or record, usually stored in a database, representing a uniquely definable entity in an application, such as a route, operation, or disposition code.

Object ID Identifier used by the InTrack system to locate objects in the relational database. An object ID points to a specific field(s) in the tables. *See also* keys.

Object model foundation (OMF) The collection of object types representing a manufacturing process; the basic design concept of ModelMaker.

Object report A printout of any or all of the static information displayed in the object viewer area. The object list should always be updated before an object report is generated.

Object selector button A button that appears at the right of every field requiring an object name. When selected, it causes a Select... dialog box to display.

Object type A particular class or category of objects.

OLE .Microsoft's object-based technology for sharing information and services across process and machine boundaries (object linking and embedding).

On hold The status of a WIP lot, inventory lot, or bulk inventory quantity when it is in a unprocessable state for an unspecified length of time, pending release. When a quantity is on hold, it cannot be moved. *See also* released.

Operation An element of work performed during the manufacturing process. An operation can occur at one or more steps on a route and can be associated with one or more routes. An operation can have a work instructions set and one or more dataset templates associated with it.

Operation bitmap In a route step graphic, a bitmap that is an operation attribute and displays once an operation is assigned to a route step.

Output Data processed and formatted for use by an end user, system, device, or program; graded material coming out of an operation.

Output material Material created as the result of an operation. Output materials include coproducts and by-products. *See also* coproduct, by-product.

Output parameter Defines a value to be returned from the database once a runtime transaction is complete (e.g., number of total records found, value of an object attribute).

Overlapped operation An operation that completes the processing of WIP lots in partial quantities, and advances the partial quantities to the next operation on the route. *See also* flow lot.

Override A changed attribute of an operation on a route that manufactures a target product. The operation attribute change is specific to the product or a customer specification associated with that product.

Pane One of the two main sections of the Route window. The Graphical view is displayed in the top pane, and the Step Properties, Material Input, and Material Output views are displayed in the bottom pane.

Parameter An informational element that has a value. Parameters define the values to be written to or returned from the database. *See also* input parameter, output parameter.

Parameter type Specifies how a parameter value is stored in the InTrack relational database (message, integer, date).

Parent lot The original WIP lot that has been divided into sublots.

Parent DataSet Template The definition of a group of data items for which multiple other dataset templates can depend upon for all or part of their definition.

Primary quantity The quantity for a particular WIP lot on a route. The primary quantity defaults to the quantity specified as the standard quantity on the bill of material for the product.

Print A system command that prints a database report.

Product Generally, a type of material produced by a manufacturer. A product's characteristics are defined by the product's material type. During manufacture, a product can be called a part, a component, a product, or a lot. A final product is the primary finished goods produced on a route. *See also* by-product, coproduct, *and* target product.

Product genealogy The complete manufacturing history of a product.

Purge A system transaction that purges from the database WIP lots, inventory lots, and bulk inventory that have a quantity of zero and have exceeded a specified time limit from the last activity date.

Quarantined The status of a WIP lot, inventory lot, or bulk inventory where the lot or inventory is allowed to continue processing, but is prevented from being shipped or consumed. *See also* active.

Query A script statement issued to the database by a client that searches for objects in a database table. Queries can be performed for both activity objects and structural objects.

Query result A script statement that returns the characteristics, or object attributes, of any object located by a database query. A query result can only be issued for one object at a time.

Queue (1) The state in which a WIP lot waits to enter an operation on a manufacturing route. The arrangement of the WIP lots in queue determines the processing priority. (2) The order in which transactions are processed by the InTrack system.

Queue time The amount of time a WIP lot waits to start an operation.

Raw material Input material that has not been processed on a manufacturing route.

Raw material inventory Inventory of unprocessed material to be consumed during the manufacturing process.

Real A floating point number represented by digits with a fixed base, such as the decimal system. A real number can be made up of either a finite or infinite set of digits.

Receive An inventory lot or bulk inventory transaction that creates new inventory or brings material into existing inventory.

Record In a database table, a row describing a single occurrence of a specific entity, such as a WIP lot, an inventory, or a transaction.

Reinitialize Returning a database to its initial state. Reinitializing erases all rows from InTrack tables (all model objects created using InTrack ModelMaker and all runtime data are removed).

Relational database A database structure that organizes data according to the relationships between the data. In a relational database, relationships between data items are expressed by means of tables.

Relational database management system (RDBMS) A system that manages access, integrity, recovery, and security of data stored in a database. All clients must be able to communicate with the RDBMS before InTrack is installed.

Released The status of a WIP lot, inventory lot, or bulk inventory that releases it from being on hold, enabling it to be processed. *See also* on hold.

Remaining cycle time The time left in the cycle to complete the WIP lot on the route. The remaining cycle time is calculated by adding up the cycle times for the remaining operations on the route.

Restore The process of writing previously archived data back to the RDBMS. See "archive".

Resynchronize Updating local cache memory with current, saved definitions in the database.

Rework A disposition code for a route operation that will move all or part of the WIP lot to another route step or to another route for additional processing. The system does not enforce a predefined meaning for the rework-type disposition code.

Route A network of possible paths containing operations that a WIP lot can follow during manufacturing. A route is made up of route steps, with an operation at each step. The route specifies the sequence of operations that are performed on a WIP lot as it moves along the route.

Route graphic A graphic that depicts a route step and shows details about the properties of the route step. For example, the presence of shading or a border on a route step provide visual representations of the route and route step characteristics.

Route path The logical movement of a WIP lot between two route steps.

Route step On a route, a step associated with one operation.

Route views Multiple types of route displays, used in InTrack to show multiple objects and their attributes and relational dependencies to each other; graphical, tabular, step properties, material input, and material output.

Row *See* record.

Runtime environment Activities performed by an end user (operator) to execute the application once it has been built; the time during which the control unit fetches data and the arithmetic-logic unit performs actual processing. *See also* ModelMaker.

Scale To define input or output materials so that their quantities will always match the ratio of the target product quantity.

Schema The structure of the tables in a database. The schema of the database tables for a manufacturing process can only be changed by a system administrator or by adding UDAs and dataset templates to the model.

Scrap To remove a quantity from the route. Scrapped quantities are no longer tracked.

Script A set of instructions executed within the InTrack runtime environment to automatically perform transactions.

Script function A set of logical commands or functions used at runtime to process WIP lot, inventory lot, or bulk inventory items.

Secondary quantity A trackable quantity for a particular WIP lot on a route in addition to the primary quantity. A secondary quantity is tracked at the WIP lot level; a WIP sublot cannot have a secondary quantity.

Security group An object in ModelMaker that establishes access privileges for an assigned group.

Selector wizard A spreadsheet type display object used for viewing and selecting work items at runtime.

Server The hardware system containing InTrack database(s) and data.

Setpoint The value, such as a target temperature or other measurement setting, that is downloaded to a process controller to define the normal running parameter of a machine.

Ship A lot or bulk inventory transaction that tracks finished goods being shipped to customers and removes the shipped quantity from the database.

Side exit An exit out of a route step that is associated with a disposition code other than the default. A different side exit is used for each disposition code associated with the route step. A side exit connection is offset from the center on any side of a route step.

Slack time The difference between the due date and the standard cycle time for the route. Slack time is created when a manually assigned due date is an overestimate, or when a WIP lot progresses through operations on the route faster than the cycle times specified for the operations.

Split A WIP lot or inventory lot transaction that divides one lot into two lots, called sublots. The system maintains the individual genealogy of a split lot.

Split lot A quantity that has been split off from a WIP lot. The system maintains the individual genealogy of a split lot.

Standard yield *See* yield.

Start A WIP lot transaction that changes the WIP lot status from "in-queue" to "in process."

Statistical process control The method of monitoring and controlling a process by gathering data about the characteristics of the output, analyzing the data, and drawing conclusions based on that data.

Statistical quality control General management of quality control, cause and effect analysis, and compliance to industry standards (ISO 9000).

Status A WIP lot, inventory lot, or bulk inventory transaction used to place a lot or inventory quantity on hold or in quarantine. This transaction is also used to release a lot or inventory quantity on hold or in quarantine.

Status bar A line at the bottom of ModelMaker window that displays information about the current status of the application. The status bar can be displayed or hidden from view.

Step ID The identifier for a single step in a route. The default step ID is a sequential number assigned as the route step is built. The default step ID can be changed to a different number or to a character string.

Step name The identifier for a single step in a route. The step name is a combination of the step ID and the operation name, separated by a colon.

Step properties view — One of five types of views in the route document window; used to assign disposition codes between two route steps. *See also* route views.

Stock room — An inventory location, where material is received into inventory or issued for consumption.

Storage location — A place where an inventory lot or bulk inventory physically resides.

String — A text expression treated as a single data item. A string does not require a special format or syntax.

String selection — A discrete list of possible text choices (such as Red, Green, Blue).

Structural object — Objects created at development in ModelMaker. Structural objects are, for the most part, static components of the manufacturing model. When used together, structural objects provide the "structure," or rules, that support the flow of WIP lots, inventory lot, and bulk inventory and their related data in a manufacturing enterprise. *See also* activity object.

Structured Query Language (SQL) — A language used in relational database systems for defining, searching for, and manipulating data.

Sublots — Quantities of a WIP lot that are spread out among consecutive operations. Sublots share the same lot ID as the "parent," or original, WIP lot.

Substitutes — Materials that can be used in place of previously designated materials in a product's bill of material. Substitutes can be assigned in varying ratios of UOM to the material it is replacing, and in various percentages of the originally designated material.

Table — Group of related data entities and their characteristics. *See also* row, column.

Tabular view — One of five types of views in the route document window; used to assign the operations for each route step. *See also* route views.

TagList — A collection of tagnames referenced by one or more labels or names. TagLists can be used by one or more InTrack commands that will link the tagnames in the list to command parameters so that the system can automatically perform validation, data acquisition, and processing. There are five types of TagLists: Command TagLists, Data Collection TagLists, Object Details (UDAs) TagLists, Diagnostics TagLists, and Query TagLists.

TagList template — A pre-defined set of input and/or output parameters upon which a TagList is based. Each type of TagList has one or more available templates. *See also* TagList, input parameter, *and* output parameter.

Target product The primary finished goods produced on a route. The target product of one manufacturing route can be an input material for another manufacturing route.

Template *See* TagList template.

Toolbar A line of short-cut buttons located directly below the menu bar. ModelMaker toolbar includes the object editor buttons. The toolbar can be hidden from view, or displayed with labeled or unlabeled buttons.

Tracking Monitoring and recording the changing status of material quantities traversing a route, or residing in inventory.

Transaction A collection of one or more OLE script statements that read and/or write to the relational database. A transaction is a request to the InTrack system to find, enter, change, or return information about an activity or structural object the relational database. All transactions are processed at runtime and are performed as a single unit of work for an object.

UDA *See* user-defined attribute.

Unassemble A WIP lot transaction to remove previously assembled components. See "Disassemble".

Unarchive The process of restoring to the InTrack RDBMS previously archived data. See "Archive".

Undo A WIP lot, inventory lot, or bulk inventory transaction that returns the selected lot or inventory to the status it had prior to the most recent transaction .

User A person performing work in InTrack. User access is controlled in InTrack ModelMaker.

User certification Qualifies a worker to perform specific operations and allows the qualification to be tracked in InTrack.

User-defined attribute (UDA) A characteristic of an object that a user defines uniquely (as opposed to built-in attributes). UDAs are contained in fields added to the database using the InTrack ModelMaker.

Version The subdivision of a class of structural objects according to criteria established by the end user; the naming convention used for edited revisions of an object.

Views Types of displays in InTrack that allows a user to view, define, and manipulate objects in a tracking model.

WindowMaker The development environment used to create the display and control components for viewing and manipulating the manufacturing model at runtime.

WindowViewer The runtime environment used to view and manipulate the database for the manufacturing model.

WIP *See* work in process.

WIP lot A collection of the same material type processed and tracked together through manufacturing processes.

WIP Selector A runtime display object into the InTrack database tables used by the operator for selecting WIP lots to work on. *See also* Bulk Inventory Selector, Inventory Lot Selector.

Work in process (WIP) Material undergoing a value-added transformation by completing operations along a route.

Work instructions A set of instructions that indicate to an operator task(s) to perform at an operation, safety procedures, etc.

Yield The percentage of good material coming out of an operation. Yield is used to determine how much material is needed to start a route to produce the target product.