



Best Practices in Testing Contact Center Voice Applications



Table of Contents

INTRODUCTION	1
WHAT IS A BOTTLENECK?	2
WHY DO BOTTLENECKS EXIST?	4
ISOLATING AND IDENTIFYING CONTACT CENTER PERFORMANCE BOTTLENECKS	5
It's Not Only What You Test But How You Test	5
Step 1 – The Testing Approach -- Set Up and Methodology	8
Step 2 - Know Your Architecture	8
Step 3 – Component Monitoring	9
Step 4 – Running Tests and Finding the Bottlenecks	10
Case Study: The Value of Testing	11
SUMMARY	12
ABOUT EMPIRIX INC.	13



Introduction

Most customer contact with a company happens through the contact center. Traditionally, call center managers have focused cost and customer Quality of Experience initiatives on agent performance. As the contact center infrastructure has become increasingly complex, leading companies have increasingly recognized that their voice system and applications must perform efficiently in order for their agents to perform effectively.

For a customer call to be successful, many voice system components – switch/ACD, IVR, database, network, and CTI – must work together seamlessly. The entire system also must be able to handle the expected call volume and traffic, as well as grow with the business. To ensure seamless performance, most companies test their system in the pre-production and production environment prior to wide-scale deployment. Yet, for a variety of reasons, including time or expertise constraints to poor planning or execution, these tests fail to capture all the problems and truly prove that the voice system can meet the business objectives it was designed to support. As a result, costs go up, and customer Quality of Experience suffers. Problems are left for the customers to detect and for technical teams to scramble to resolve.

To help companies identify problems before their customers do and to ensure that their voice systems will scale, Empirix consultants have created this easy-to-follow guide for spotting and resolving bottlenecks within a Contact Center system. The paper describes the most common Contact Center bottlenecks that occur and the methods our consultants have successfully used to identify them. The advice this paper presents is based on knowledge accrued through engagements on countless system configurations at hundreds of Contact Centers.

Performance bottlenecks exist in all Contact Center systems. The application development teams that create Contact Center applications typically are responsible for locating, identifying, and correcting them before an application launches. Proper testing techniques and execution can help improve the performance of any step in the call flow, as shown in Figure 1. Note, however, that ensuring quality does not end with successful pre-production and production testing. Because a contact system will undoubtedly undergo software and hardware changes during production, ongoing monitoring is also very important. It will immediately alert contact center operations managers when problems occur, providing them with real-time performance data they can use to resolve these problems quickly.

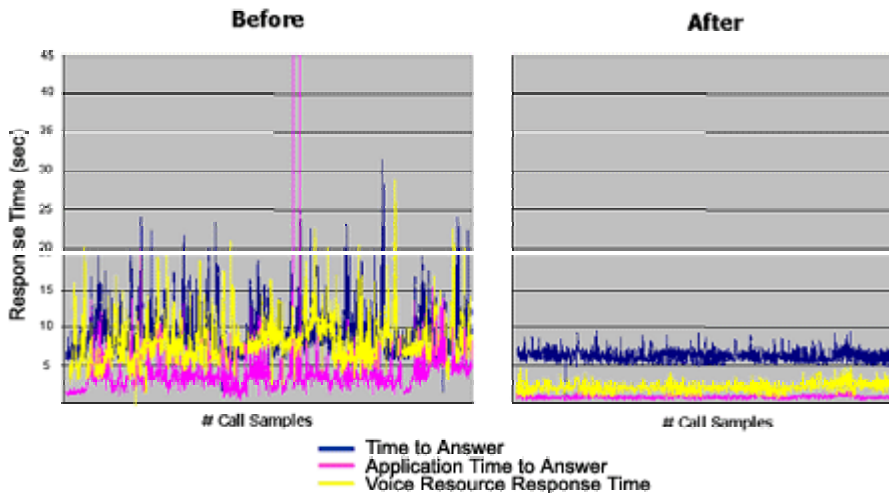


Figure 1 – These graphs represent dramatic improvements for three performance metrics.

What is a Bottleneck?

Any Contact Center system that is subjected to testing will exhibit performance bottlenecks. This paper defines performance bottlenecks and suggests how to recognize and resolve them. Below is a list of terms used throughout the paper.

Bottleneck – This is a term used for a resource in the system under test that limits either performance or scalability. Bottlenecks are typically identified as the cause of slow or unacceptable performance. At any point in time, only one bottleneck causes the system to slow for a specific transaction.

Performance – In system testing, this generally refers to the ability of a system to meet the requirements defined prior to testing. Systems are usually said to exhibit good or poor performance based on how well they meet their performance criteria. Performance can be expressed in terms of concurrent callers, transactions per day, average response time, or any other measure of speed.

Scalability – This refers to the ability of a system to maintain performance as the number of concurrent callers or requests increases. A system that can handle a specified number of concurrent users or request rate as additional load is applied is said to provide scalability.

Symptom or Problem – These are terms for how a bottleneck shows itself in the system under test. Symptoms or problems are typically the result of a bottleneck, not the cause. Knowing the symptoms or problems, and combining that knowledge with other data, can help find the causes of bottlenecks. Examples of symptoms include slow response times, routing errors, and over-utilized servers.



Bottleneck Cause or Indicator – A single point that is causing a system under test to exhibit slow performance is the cause, or indicator, of the bottleneck.

Solution – The corrective action taken to repair the cause of a bottleneck is the solution.

Measurement Point – This is a specific metric used to determine when and where a bottleneck exists. Examples of these metrics include connection rates, response times, sound quality, and prompt correctness.

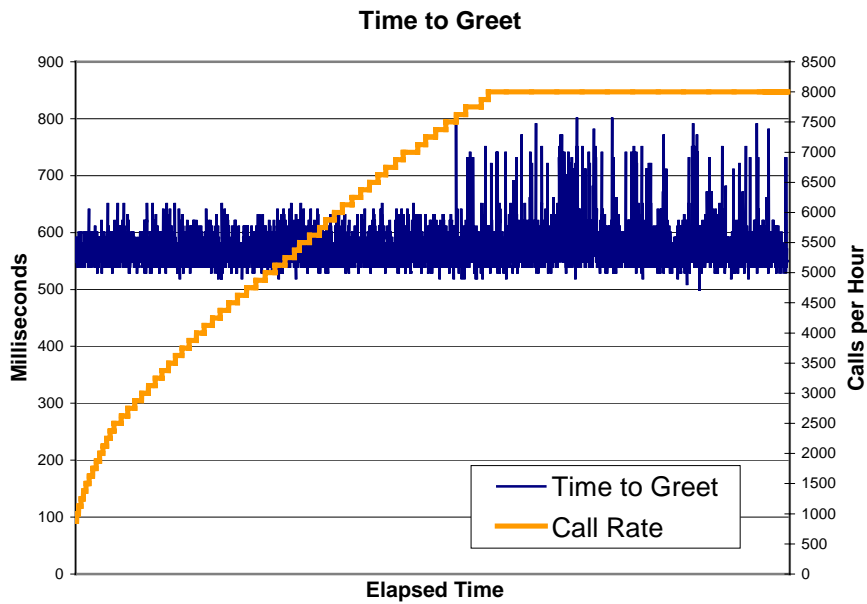


Figure 2 – This graph provides an example of a measurement point and how it would be used to identify a bottleneck. It plots response-time measurements for an application’s initial greeting prompt. The first half of the graph shows consistent response times, between 450 and 550 milliseconds. As the call rate increases, the response times increase. The second half of the graph, shows that starting at about the 7,500-call-per-hour rate, many response times are more than 550 milliseconds, with many as high as 800 milliseconds. This behavior is a result of the additional resources required as the load increases. If response times exceed acceptable limits, a bottleneck may exist that requires more CPU capacity for the IVR system or some other corrective action.



Why Do Bottlenecks Exist?

Bottlenecks exist for a number of reasons that include:

- Inadequate server hardware to support the projected system load
- Inadequate internal, external, telephony or data network capacity
- Poor system or architecture design
- Poor database implementation or tuning
- Insufficient attention given to performance considerations during development
- Lack of, or scattered, expertise on system performance
- Tight deadlines and changing requirements

During system development, these shortcomings and challenges can easily be overlooked. Component testing and system testing can miss performance problems in even the best-designed systems. Performance testing is the only way to isolate, identify, and resolve all bottlenecks before a system is launched.

Because *no* system is infinitely scalable, testers must define the criteria that will allow them to evaluate system performance. By using feedback from business users and owners of the system, they can develop meaningful criteria that match business needs. The criteria can be used in testing to determine whether the system under test has met performance expectations.

The team responsible for system performance should understand the immediate performance requirements and contrast them with growth performance requirements. Systems that meet minimal performance standards can be launched, and improvements can be made during subsequent releases, effectively allowing the system to become more capable as business needs change.

Knowing the maximum performance level the current system will handle is critical. It may be defined in terms of number of concurrent callers, transactions per second, sales per day, or any other criteria relevant to the business the application supports. Load testing will help in planning for future needs so that the system can keep pace with business requirements.



Isolating and Identifying Contact Center Performance Bottlenecks

It's Not Only What You Test But How You Test

Bottlenecks can occur in any part of the system architecture. Our experience shows that they are more likely to occur in some places than in others. Most of the time, problems found within Contact Center systems are related to the integration of hardware and CTI software.

The most common problem areas are:

- Capacity
- Switch setup/programming
- Backend database
- Voice applications and middleware

There often is more than one bottleneck in systems that have not been tuned or tested. Because only ONE bottleneck can be found at a time, it becomes necessary to iteratively test and fix the system throughout the development process. Bearing in mind that each test can show just one problem that is causing the system to slow down help prevent you from chasing after other, unrelated problems. Once a test has found the bottleneck causing the problem, and it has been fixed, testing can be performed to find another one. This type of iterative testing process is the most effective way to eliminate performance and scalability bottlenecks. The chart below shows where the most common Contact Center performance problems occur. These findings are based on data gathered by Empirix consultants and engineers in successful customer engagements.

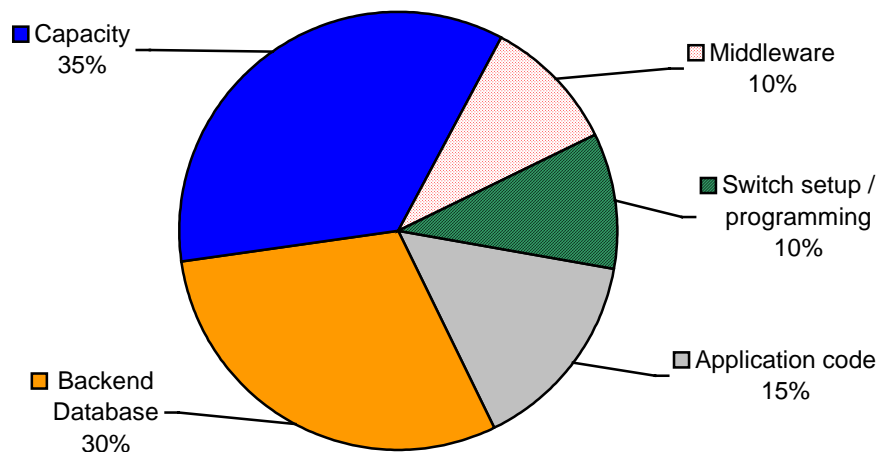


Figure 3 – Note that more than half of the most common performance problems plaguing Contact Centers occur in the backend database or CPU.

The following sections present a methodology you can follow to quickly and efficiently find, isolate, and resolve performance bottlenecks.



Step 1 – The Testing Approach – Setup and Methodology

The Typical Approach

For most Contact Center systems, the application team has an idea of the size of the load the system will need to handle. This load is often defined in terms of calls per minute. For example, a system might need to support, at peak, 100 calls per minute. (If these calls last three minutes on average, this would be equivalent to 300 concurrent callers performing typical transactions). The Empirix load testing systems or services simulate real callers dialing in to the system in order to determine whether that performance goal can be met.

With the target number of calls per minute in mind, a series of scripts is created. Each script emulates some type of activity that will occur in the application. The scripts, using different data and transactions, test various levels of simultaneous callers. The results of those tests indicate how well the Contact Center will handle the anticipated load.

After testing begins, symptoms of a performance bottleneck typically arise, especially in systems that are early in the iterative testing process. The symptoms could be high utilization of the database server, for example, or slow response times for a prompt. The next step – isolating the specific bottleneck that is causing this symptom – can be difficult because Contact Center systems involve many subsystems that can be decentralized or operated by different organizations. In addition, these subsystems often interact in ways that may not be clearly perceived through discrete analysis. Empirix has found a way to isolate and identify performance problems despite these challenges using a unique approach.

The Empirix Approach

The Empirix approach to testing includes feature, performance, and monitoring stages. Feature testing includes pre-production testing of application parameters and functions during development. Performance testing is done both in the pre-production environment and production environment immediately before deployment. Many companies schedule it to occur throughout the year. Performance testing simulates multiple simultaneous callers to test total system performance under load.

Ongoing monitoring ensures system stability in a production environment over time by placing an automated test call at a predefined interval (usually every 15 minutes) to ensure that if a problem occurs, technical staff will immediately be alerted and informed of where the error occurred. Ongoing monitoring not only cuts down on problem detection and resolution time, but also provides objective performance metrics to identify systemic performance issues and drive continuous improvements. After feature testing the various components of the system, a larger, broader set of tests are run to validate the performance criteria that have been set for the system under test. Once the performance tests have isolated bottlenecks and those bottlenecks have been corrected, the system is ready for production and the monitoring stage.

Planning a Performance Test

Planning your performance testing, before any work begins, is critical. The following steps can help you streamline the testing process and ensure its success.



- Lay out the strategy that will be used for testing your Contact Center system.
- Document the objectives of the load test and get buy-in from all relevant parties.
- Clearly define the scope by determining what will and will not be tested during the load test phase.
- Document the system performance requirements that are used to determine whether the system passes or fails.
- Document the test cases and scripts that will be used.
- Document the system and personnel resources that will be needed for testing. Work with their owners to ensure that all the resources will be available for testing.
- Gather and document any data that will be needed for testing, including both back-end database information and front-end data entered by the customer.
- Document all tools that will be used during the testing and the role each will play.
- Have a plan in place to address any serious errors or performance problems found during testing.

Build the Testing Team

Performance testing is a highly technical activity that requires a broad array of skill sets. These people should be included on the testing team:

- **Senior Testers** – There is no substitute for expertise. These people will coordinate, drive, and ultimately be held responsible for the performance testing activities. They should have an excellent understanding of performance testing and be able to adeptly use the tools needed to carry out their testing duties.
- **Database Administrators** – Database experts will be needed when the tests are run to help optimize testing and spot database performance problems.
- **Telephony Architects** – Telephony experts can help spot configuration problems at the network level, including problems with the switches, interactive voice response (IVR) units, automatic call distributors (ACD), intelligent call managers (ICM), load balancers, networks, firewalls, routers, hubs, and other infrastructure components.
- **Senior Developers/Architects** – The designers of the system under test should be available to answer any questions about functionality and help resolve code-related issues that arise.



Environment and Load Test Tool Setup

The final step is setting up the hardware and software for the tool that will be used. If the testing is being done by a remote testing service, this will require setting up system monitors on the server components and providing access to the system, where necessary. In addition, you will need to verify that the system under test is ready for performance testing. All systems should be configured and set up as they would be in production, with one exception. Test accounts, not real customer data, should be used during testing. Databases should be loaded with production data or at least data that is sized appropriately for production. If replicating the production environment is not feasible or possible, you should take great care in making any assumptions about the scalability of hardware components as they are added. A lab environment rarely contains the mirror-image equipment, therefore, items that were component tested in the lab should be tested in the real production environment before the system under test goes live.

Step 2 - Know Your Architecture

The second step in isolating performance bottlenecks in any system under test is to know your architecture. You must have a firm understanding of the physical components (such as switches, voice response systems, CTI servers, as well as internal and external networks) and logical components that make up your system.

Scripting test scripts should be well thought-out and have a specific purpose in mind. They should apply to specific functionality and follow the guidelines listed below:

- Test the transactional call flows that get the most use (or that you estimate will get the most use).
- Test the most commonly used business transactions in your system.
- Test the most critical business functions on your system.
- Test the transactional call flows and areas that developers suspect may cause bottlenecks.

Most scripts fall into one of three categories:

- End-to-end – Long transactions that cover a wide range of functionality
- Modular – transaction-specific scripts
- Single – short, transaction-specific scripts

Modular tests or a single/low number of test calls work best for isolating bottlenecks. These are the transactions you will want to script for testing. Using a tool like Empirix's CallMaster, model your Call Center system, then tag short tests that focus narrowly on problem components or functions. This model can also be used in feature (regression) testing, performance testing (as this document discusses), and production monitoring.



Step 3 – Component Monitoring

Monitoring the servers and other components of your system during a performance test provides important data that supplements the results collected by the Empirix testing tools and services.

Develop a list of the critical metrics that should be monitored on each system component. Test, at minimum, the metrics listed below. More metrics can be added as testing progresses.

Switches

- Total current connections – The system should be able to sustain a pre-determined number of concurrent calls
- Load balancing split – Is the switch working as intended?
- System utilization (CPU) – High utilization rates, greater than 70%, may indicate a bottleneck.

CTI Servers

- System utilization (CPU) – Again, utilization of greater than 70% may signal a potential problem
- Memory utilization – Verify that the available memory on the Web systems is sufficient. Pay close attention to this metric if the system requires any SSL traffic.
- Throughput – Verify that the CTI servers can keep up with their rated network capacity.

Voice Response Systems

- Memory utilization – Too little memory can result in degraded application performance under load.
- CPU – High CPU utilization can be an indicator of a well-designed system working very hard or a poorly designed system trying to work, but being bottlenecked by other problems.
- Connection pool – There should be enough available connections to the database to satisfy the requests being made. An insufficient number of threads will reduce performance.
- Queue – Monitor the number of calls that go into the queue. When a disproportionate number of requests start to queue, you may have hit a bottleneck.
- Queue wait time – Predefine the acceptable queue wait time and monitor for excessive waits.
- Disk IO – Verify that disk accesses do not degrade application performance.

Database Servers

- Memory utilization – Verify that the database has ample memory for cache. The more memory, the better performance you can expect.



- CPU – High CPU utilization can be an indicator of a well-designed system working very hard or a poorly designed system trying to work, but being bottlenecked by other problems.
- Table scans – Verify that there are low table scans on all critical business transactions.
- Table locks – Make sure that processes are not locking critical tables because that will slow access to them.
- Parse ratio – If parsing is high, convert to stored procedures and update the query plan cache.
- Cache hit ratio – Make sure there is ample cache to store the data.

External Systems

- Current connections – Make sure there are enough connections to handle the requests to external systems.
- Bandwidth consumption – Verify that the bandwidth to the external system providers is sufficient.
- Queue wait time – A queue is acceptable, as long as the wait time it causes is not excessive.

Step 4 – Running Tests and Finding the Bottlenecks

During this step, you will execute the tests you have created, assuming that you have created your scripts, and your environment is ready for testing. Start by running one type of transaction at a time. Remember, while running individual performance tests, the key is to run each critical transaction on its own to find any problems, then fix, retest, fix, and retest. When the transaction performs acceptably, move on to the next transaction.

After each transaction is performing acceptably on its own, you can move on to a larger scenario test that will verify that all the transactions are working together. The idea is to resolve the issues in each piece of the system *first*, then to move on to the big picture. Follow these guidelines:

- 1) Execute a low volume load test to validate system stability and gather baseline data.
- 2) Ramp the calls per minute up to a rate that allows the system to stabilize before adding more callers. The time spent at each prompt should be set at a fixed amount in order to allow you to understand exactly where the delays are occurring.
- 3) Keep the ramp constant until you notice a slowdown in response times or increased call failures (as illustrated in Figure 4).
- 4) Check the servers you are monitoring to see whether any are exceeding CPU or memory capacity.



- 5) If the servers are exceeding capacity, note which machine is exhibiting a problem, refer to your network diagram, and determine what piece of code is running the problem transaction. Your database administrator or programmer can identify potential inefficiencies. Fix, and retest. If resources are not being exceeded, you may have a problem with network bandwidth or another resource constraint. For example, queuing requests to a specific resource or to an outside system may be the problem. Identify, fix, and retest.

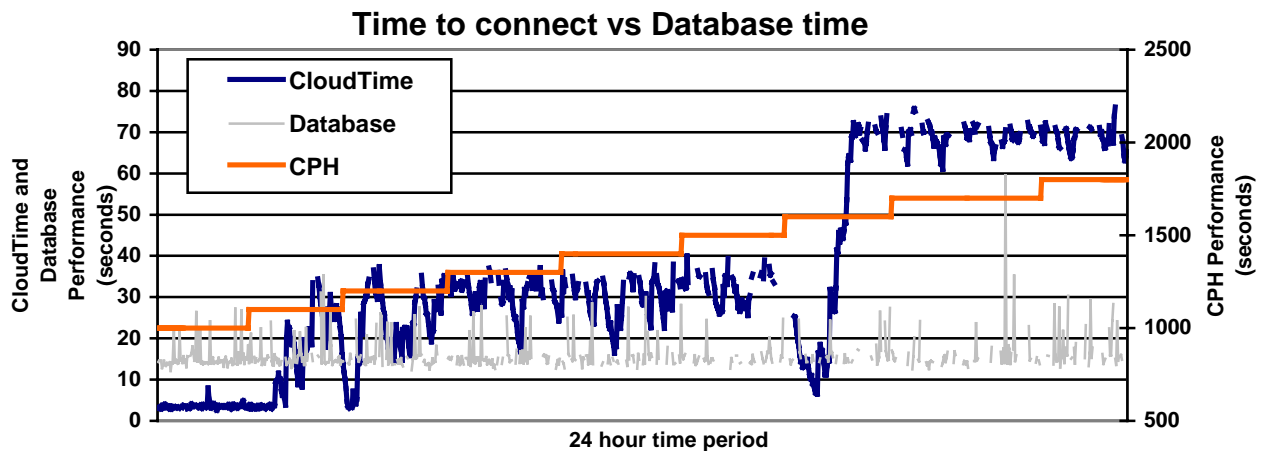


Figure 4 – This graph shows how ramping call volume affects how a system performs under load.

Case Study: The Value of Testing

This case study summarizes how Empirix consultants implemented best practices to help a large cable TV company ensure the quality of its Contact Center voice applications before they were deployed.

Problem

A well-known cable TV company was in the final stage of implementing a new IVR-based customer service application that included CTI-based call routing. Low-volume, manual load tests on the systems had not found any problems. Ordinarily, this would have been the only testing performed prior to deployment of the new systems. The company's IT development group decided to perform additional testing that would let them know – with complete certainty – that the new systems would perform properly under real-world calling conditions.

Solution

As a result, the group decided to use Hammer technology from Empirix to perform automated testing to supplement the earlier manual tests. Its goals were accurately measure how system performance, including time to connect, call length, and screen pop success, was affected under increasing call load levels.



Result

Almost as soon as the company's QA test group started testing, the consultants found serious problems with both the IVR and CTI systems – problems that had not surfaced during in manual testing. The IVR system was able to handle as many as three simultaneous calls without problems; however, under higher load conditions, 10-15 percent of incoming calls would fail. Incorrect prompts would start to play for a couple of seconds, then the correct prompt would break in.

In addition, testing of the CTI call routing found that calls were being routed to the wrong type of agent. It also found screen pop errors and discovered that calls to some service queues were failing after the queue reached a certain size.

Thanks to Hammer testing, these problems were fixed quickly, before the new systems were deployed. Moreover, the company was able to benchmark the performance of the new systems under different call load levels and use the data they collected to gauge and plan for future capacity requirements.

Summary

Voice system performance is critical to contact center success. After all, if your systems don't perform, your agents can't perform. A thorough testing process will help ensure that your voice applications are performing smoothly and will scale to meet future business needs. Testing will not only reduce your application development time, but can also help you get the most out of your existing equipment and avoid unnecessary capital expenditures.

Empirix offers a broad set of testing products and services including the Hammer IT test system and Hammer Voice Testing services that can provide you with the confidence that your voice systems will meet your and your customer's expectations. Additionally, Voice WatchSM hosted monitoring services and OneSight for contact centers can provide you with the tools you need to ensure that your voice systems remain up and running on an ongoing basis.



About Empirix Inc.

Empirix is the only test and monitoring company to provide enterprise and telecom network customers with a comprehensive range of performance-enhancing solutions for Web, voice and data applications; contact centers; and network infrastructure. Empirix products and services enable customers to accelerate development cycles, reduce operating costs, strengthen revenue streams and improve Quality of Experience for their end-users. The company serves over 2,500 customers worldwide, including Global 2000 enterprises, leading telephone companies, switch manufacturers and e-business leaders. Headquartered in Waltham, Massachusetts, Empirix has offices throughout the United States, and in Europe and Asia. For more information, visit Empirix on the Web at <http://www.empirix.com>.

Hammer IT and OneSight are trademarks of Empirix in the United States and other countries.

WP218.1.03/02

© 2002 EMPIRIX. ALL RIGHTS RESERVED.
THIS DOCUMENT MAY NOT BE REPRODUCED IN ANY FORM OR
BY ANY MEANS WITHOUT THE EXPRESS WRITTEN PERMISSION
OF EMPIRIX INC.