SUN SEEBEYOND

JAVA™ COMPOSITE APPLICATION PLATFORM SUITE DEPLOYMENT GUIDE

Release 5.1.1

Sun microsystems

Version 20060620163711

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an overview of this Sun Java™ Composite Application Platform Suite document.

What's in This Chapter

1.1 What's New in This Release

This document includes the following new features and changes:

- The **Max Concurrent Instances** property has moved from the eInsight Engine configuration to the **General** tab of the **Business Process Properties** dialog box. This change provides added flexibility. If a Project has multiple Business Processes, then you can configure each Business Process to have a different number of maximum concurrent instances.

1.2 About This Document

1.2.1 What's in This Document

This document includes the following information:

- **Chapter 1 "Introduction"** introduces you to this guide, its general purpose and scope, its organization, and writing conventions. It also provides sources of related documentation and information.
- **Chapter 2 "Overview of the Sun Java Composite Application Platform Suite"** provides a general overview of Java CAPS.

- **Chapter 3** **“Requirements Analysis and Deployment Planning”** describes the first two phases of deploying Java CAPS: requirements analysis and deployment planning.
- **Chapter 4** **“System Design and Development”** describes the third phase of deploying Java CAPS: system design and development.
- **Chapter 5** **“Testing, Transition to Production, and Maintenance”** describes the final three phases of deploying Java CAPS: pre-transition testing, transition to production, and post-transition maintenance.
- **Chapter 6** **“Determining System Requirements”** offers guidelines to help you determine the system requirements for the deployment of Java CAPS.
- **Chapter 7** **“Configuring Failover Support in a Sun Clustering Environment”** describes how to configure failover support for the Repository and the Logical Host using Sun™ Cluster 3.1.
- **Chapter 8** **“Configuring Failover Support in a Windows Clustering Environment”** describes how to configure failover support for the Repository and the Logical Host using Windows Server 2003 clustering technologies.
- **Chapter 9** **“eInsight Load Balancing and Failover”** describes how to configure Sun SeeBeyond eInsight™ Business Process Manager for load balancing and failover.

1.2.2 Scope

This guide provides deployment planning guidelines and deployment strategies for Java CAPS.

1.2.3 Intended Audience

This guide is designed for management, system administrators, and others who are tasked with deployment of Java CAPS.

1.2.4 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

Text Convention	Used For	Examples
Bold	Names of buttons, files, icons, parameters, variables, methods, menus, and objects	▪ Click **OK**. ▪ On the **File** menu, click **Exit**. ▪ Select the **eGate.sar** file.
`Monospaced`	Command line arguments, code samples; variables are shown in ***bold italic***	`java -jar` ***`filename`***`.jar`
Blue bold	Hypertext links within document	See **Text Conventions** on page 11

Table 1 Text Conventions (Continued)

Text Convention	Used For	Examples
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.2.5 Screenshots

Depending on what products you have installed, and how they are configured, the screenshots in this document may differ from what you see on your system.

1.3 Related Documents

For more information about Java CAPS, see the following documents:

- *Java Composite Application Platform Suite Installation Guide*
- *Java Composite Application Platform Suite Primer*
- *Sun SeeBeyond eBAM™ Studio User's Guide*
- *Sun SeeBeyond eGate™ Integrator JMS Reference Guide*
- *Sun SeeBeyond eGate Integrator System Administration Guide*
- *Sun SeeBeyond eGate Integrator User's Guide*
- *Sun SeeBeyond eGate Tutorial*
- *Sun SeeBeyond eIndex™ Single Patient Identifier User's Guide*
- *Sun SeeBeyond eInsight Business Process Manager User's Guide*
- *Sun SeeBeyond eTL™ Integrator User's Guide*
- *Sun SeeBeyond eView™ Studio User's Guide*
- *Sun SeeBeyond eVision™ Studio User's Guide*
- *Sun SeeBeyond eXchange™ Integrator User's Guide*

In addition, each eWay™ Adapter has a user's guide.

1.4 Sun Microsystems, Inc. Web Site

The Sun Microsystems web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.sun.com

1.5 Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

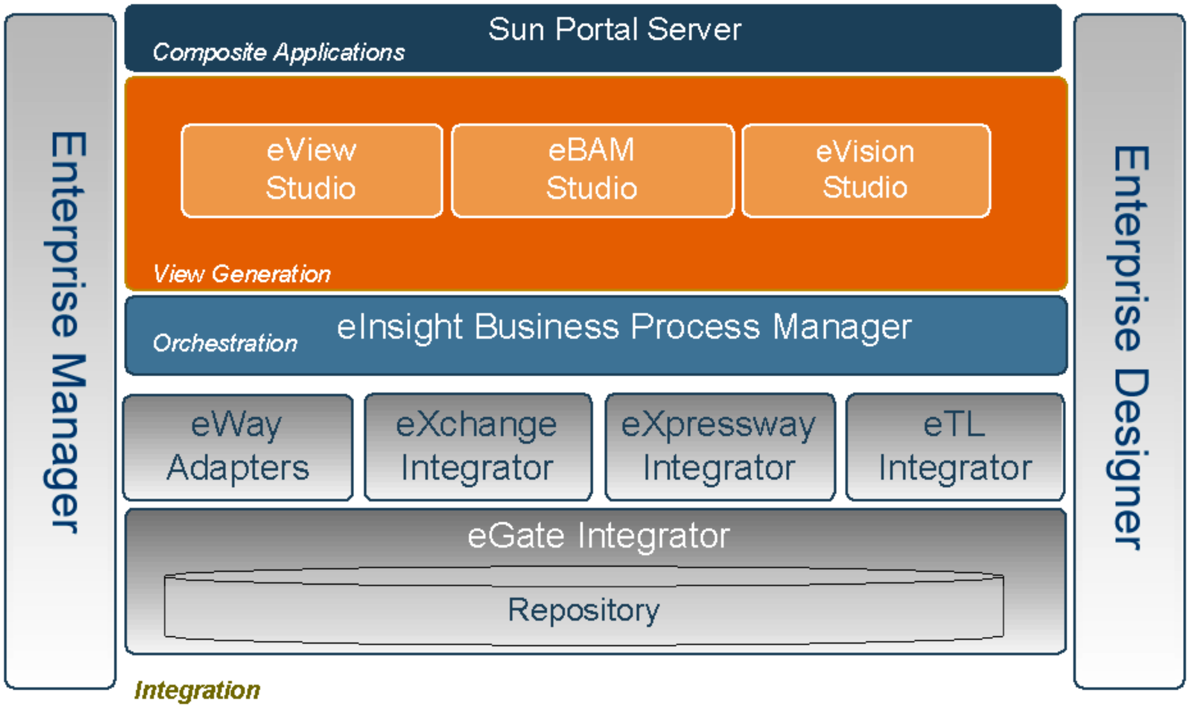
CAPS_docsfeedback@sun.com

Chapter 2

Overview of the Sun Java Composite Application Platform Suite

This chapter provides a general overview of Java CAPS.

Figure 1 Java CAPS



What’s in This Chapter

- "Sun SeeBeyond eIndex™ Single Patient View" on page 18
- "Sun SeeBeyond eBAM™ Studio" on page 19

2.1 Sun SeeBeyond eGate™ Integrator

Sun SeeBeyond eGate Integrator is a fully Java™ 2 Platform, Enterprise Edition (J2EE™ platform) certified and web services-based, distributed integration platform that serves as the foundation of Java CAPS.

eGate Integrator provides the core integration platform, comprehensive systems connectivity, guaranteed messaging, and robust transformation capabilities. eGate Integrator also provides a unified, single sign-on environment for integration development, deployment, monitoring, and management.

2.1.1 Integration Model

eGate Integrator addresses application integration by means of a Project, which contains the business logic required to solve the specific problem. The Project contains the various logical components and supporting information required to perform the routing, processing, and caching of messages containing the relevant data from one application to another. Project information is stored in the Repository.

2.1.2 User Interfaces

The main user interfaces are Enterprise Designer and Enterprise Manager.

Enterprise Designer

Enterprise Designer is used to create and configure the logical components and physical resources of a Project. Through this interface, you can develop Projects to process and route data through an eGate Integrator system. Enterprise Designer is also used by other components of Java CAPS.

Enterprise Manager

Enterprise Manager is a web-based application that is used to deploy and monitor runtime components.

2.1.3 System Architecture

eGate Integrator employs a versatile architecture that is ideally suited to distributed computing environments. As a result, the various components of an eGate Integrator system can reside on the same hardware platform (assuming adequate system resources), or be distributed across several different hardware platforms in the enterprise network.

2.2 Sun SeeBeyond eInsight™ Business Process Manager

Sun SeeBeyond eInsight Business Process Manager delivers business process management features and functions to Java CAPS.

Business process management is a strategic orchestration of the movement of information and the flow of complex processes between participants (systems, users, and organizations) to accomplish larger business objectives.

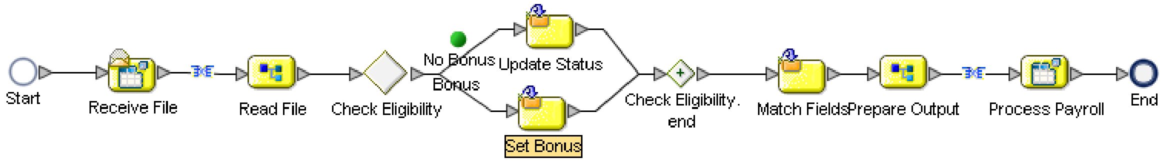
eInsight provides you with a clear view into the internal and external processes of an organization. These processes may be executed by computer systems or employees.

Business process management consists of two phases:

- The design phase begins before you start using eInsight and ends once the business process is deployed.
- The runtime phase refers to the tasks that you perform after the business process is deployed.

Figure 2 shows an example of a business process model created with eInsight Business Process Manager.

Figure 2 Business Process Model Example



2.3 Sun SeeBeyond eVision™ Studio

Sun SeeBeyond eVision Studio is a graphical environment that enables business analysts and web developers to rapidly create interfaces for composite applications, without the need for advanced programming abilities.

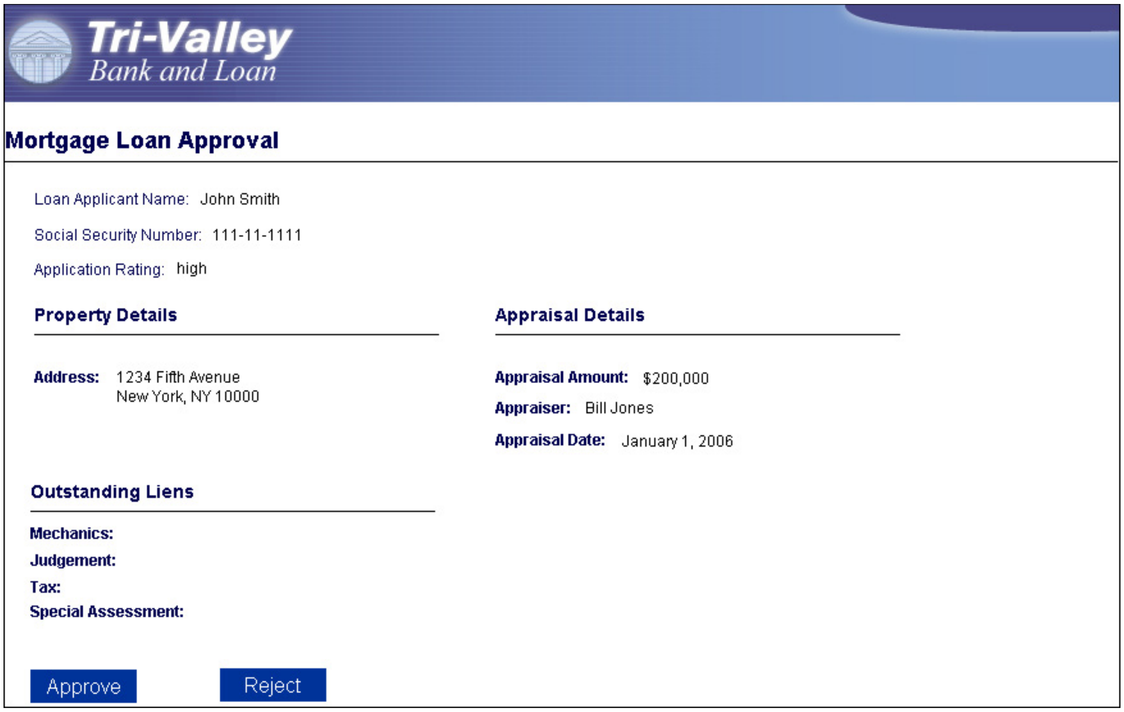
eVision Studio provides two main tools: the Page Layout Designer and the Page Flow Designer. Both tools are integrated within Enterprise Designer.

eVision Studio can be used in a variety of ways as a front end to Java CAPS solutions. For example:

- In conjunction with eInsight Business Process Manager, eVision Studio enables users to participate in business processes. The users perform workflow tasks with eVision Studio web pages that are tailored for specific organizational roles.
- eVision Studio can provide the login pages in which users enter their user name and password, which are then authenticated.

Figure 3 shows an example of a Page Layout created with eVision Studio. The runtime data is retrieved dynamically from back-end systems.

Figure 3 Page Layout Example



Mortgage Loan Approval

Loan Applicant Name: John Smith

Social Security Number: 111-11-1111

Application Rating: high

Property Details

Address: 1234 Fifth Avenue
New York, NY 10000

Appraisal Details

Appraisal Amount: $200,000

Appraiser: Bill Jones

Appraisal Date: January 1, 2006

Outstanding Liens

Mechanics:

Judgement:

Tax:

Special Assessment:

eVision Studio enables you to create a JSR 168-compliant portlet for use in JSR 168-compliant portals, such as Sun Java™ System Portal Server. The Portal Server provides a new level of enterprise productivity, enabling users and groups to work together easily and securely within the requirements of a dynamic organizational structure.

2.4 Sun SeeBeyond eTL™ Integrator

Extraction, transform, and load (ETL) is a data integration technology that extracts data from several heterogeneous data sources, transforms the data, and then loads the data in a uniform format into a target data source.

Sun SeeBeyond eTL Integrator is optimized for extracting, transforming and loading bulk data between files and databases, and provides an ETL development and runtime environment that is both fully integrated into the Java CAPS platform and specially optimized for handling very large record sets.

2.5 Sun SeeBeyond eXchange™ Integrator

Sun SeeBeyond eXchange Integrator is an application that runs on the Sun SeeBeyond Integration Server in distributed computing environments. eXchange Integrator manages interactions with trading partners by facilitating the reception, validation, transmission, and tracking of messages in supported formats.

2.6 Sun SeeBeyond eView™ Studio

Sun SeeBeyond eView Studio provides a flexible framework to allow you to create matching and indexing applications called enterprise-wide master indexes.

eView Studio helps you to design, configure, and create a master index that uniquely identifies and cross-references the business objects stored in your system databases. Business objects can be any type of entity for which you store information, such as customers, members, vendors, businesses, and hardware parts. In eView Studio, you define the data structure of the business objects to be stored and cross-referenced. In addition, you define the logic that determines how data is updated, standardized, weighted, and matched in the master index database.

2.7 Sun SeeBeyond eIndex™ Single Patient View

Sun SeeBeyond eIndex Single Patient View is a health care-oriented, enterprise-wide master person index that creates a single view of person information by maintaining the most current information about the people who participate throughout your organization, and by linking information from different locations and computer systems.

2.8 Sun SeeBeyond eBAM™ Studio

Business activity monitoring involves the collection, aggregation, and presentation of business activity data according to specified Key Performance Indicators (KPIs).

Sun SeeBeyond eBAM Studio provides the tools for generating custom, cross-application digital dashboards for defining and monitoring KPIs, which summarize the aggregated business data collected through the eInsight or eGate application layers. The eBAM Studio web interface enables the business analyst to transform data that has been collected over time into meaningful, rich visual presentations.

eBAM Studio delivers a real-time view of business activities (for example, monitoring service-level agreements for enforcement), enabling the business analyst to identify problems and trends and to resolve them proactively.

Notifications are triggered according to preset threshold limits set by the designer. eBAM Studio generates alerts when KPIs fall below or exceed a specified threshold. The alerts can be used to send email messages and to launch other processes.

Chapter 3

Requirements Analysis and Deployment Planning

This chapter describes the first two phases of deploying Java CAPS: requirements analysis and deployment planning.

What's in This Chapter

3.1 Introduction

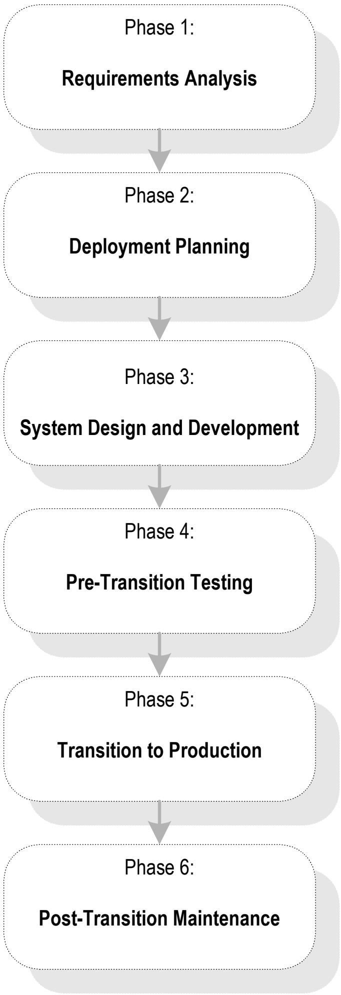
Deploying Java CAPS consists of the following phases:

1. Requirements analysis
2. Deployment planning
3. System design and development
4. Pre-transition testing
5. Transition to production
6. Post-transition maintenance

Note: *In a typical deployment, some of the phases overlap with each other. For example, unit and integration testing is usually performed during the system design and development phase.*

Figure 4 shows a diagram of these phases.

Figure 4 Deployment Phases



This chapter explains the first two phases:

- **Requirements analysis:** This guide seeks to give you a road map of how to deploy Java CAPS. First, to use a road map, you have to know where you are (analysis) and where you are going (planning). In other words, find out everything you can about your information system setup and business processes. Then, you can decide what information systems and business process needs you want Java CAPS to meet.
- **Deployment planning:** Deployment begins when you plan and schedule how, in view of your analysis and allocated resources, you want to implement your Java CAPS environment. During this phase, you set up the operation procedure and schedule for the entire deployment project.

The first two phases lay a foundation for the entire deployment project. Poor analysis and planning can cause major problems during the later phases. Thorough, comprehensive analysis and planning can make the deployment project easier, more efficient, and less costly.

3.2 Requirements Analysis

In gathering and analyzing information on your Java CAPS needs, you must first know what type of information you need. Java CAPS links your current networks, business systems, and applications together into a single, seamless information system. The purpose of this system is to facilitate your current and future business process needs. In as much detail as possible, find out where you are and what you need.

During the requirements analysis phase, you examine your needs and define the properties that the system must possess to meet those needs. Also, you identify system constraints and performance requirements. Define what functions you want the deployed system to perform, but not how the functions work (this task occurs during the system design and development phase).

This section describes the information that you need to gather to facilitate your deployment, by posing a series of relevant questions. The questions are divided into the following categories:

- System-specific
- Operation and performance
- Personnel and training
- Business planning

The examples in this section are general and are only meant to start you thinking in the right direction. You must begin by assembling general information on your needs, categorizing the information, and adding the necessary details.

3.2.1 System-Specific Needs

These needs are the basic information systems, network, and database-related requirements that you want the Java CAPS system to meet. Determine your system-specific needs by asking the following questions:

What existing systems do we need to connect?

Create a complete picture of your current information system setup. Include applications, networks, systems, platforms, and outside information pathways.

How do we want to do the connecting?

Find out how you want your various systems to talk to each other (communication protocols), which systems must be linked, and the direction of communication.

Example: We have systems A, B, C, and D. Systems A, B, and C need to communicate with all of the other systems. System D needs to communicate only with system A. All of the communication is two way.

What are our data requirements?

What types of data do you use, how much, and when?

Example: Our system uses HL7 and X12 data types. On average, our system needs to move about 100,000 messages per day at about 5 MB per message, with 90 percent of the data moving between 8 a.m. and 5 p.m. every Monday through Friday. Peak data loads are generally between 2 p.m. and 4 p.m. on weekdays (60 percent of volume).

What are our system/hardware limitations and constraints?

Installing Java CAPS requires that you have the necessary hardware and operating system software. What is your budget for additional hardware and software? Do you have any space limitations in the area where the hardware will reside?

Example: We use the Solaris™ Operating System (Solaris OS) for our large-scale systems. We already have Windows client PCs for each of our system developers who will be using Enterprise Designer.

Planning for hardware needs requires special considerations, such as how many systems you need, memory (RAM) required, the number of CPUs you need, and total disk space. Chapter 6 **"Determining System Requirements"** discusses how to analyze and plan for these additional system requirements.

3.2.2 Operation and Performance Needs

Do you have any specific system operation and performance issues? Now is the time to discover, organize, and itemize them by asking the following questions:

What are our performance requirements?

Performance is a trade-off between speed and maintainability. This statement is true overall, as well as being true for the operation of individual system component operations. You must prioritize these needs specifically.

Example: Customer databases must be totally accurate and detailed because the information is used often and is vital to the company. Detailed maintenance of this data is more important than speed of processing. However, our moment-by-moment stock quotations have to be fast and up-to-the-minute. Maintainability here is negligible because this data changes so fast that long-term retrieval is not an issue.

What are our internal security requirements?

Java CAPS has access security, which allows only certain people to log on to the system and different people to have specific privileges after the log on.

Example: The company allows only five people to log on to the system: one with system administrator privileges, two with operator privileges, and two with monitoring privileges.

What are our error-handling and data validation requirements?

How, when, and where in the system does the customer require data to be error checked and validated? Keep in mind that processing speed decreases as checking instances and the detail of error checking increases.

Example: All data passing through our system must be validated as thoroughly as possible. To facilitate this process, we compiled a complete list of all different data types that require validation.

3.2.3 Personnel and Training Needs

Deploying and maintaining the Java CAPS solution might require some expanded personnel needs. Consider the following questions:

Do we have personnel trained and able to deploy the system?

The number of resources required to deploy the Java CAPS solution depends on the size and complexity of the project. You might want to take advantage of Sun Services' in-depth product knowledge and experience. In addition, Sun offers a comprehensive set of courses.

Example: We need three resources to deploy the solution. The senior member of the team must attend Sun training.

Do we have personnel trained and able to maintain the system after deployment?

You must also examine the personnel and training needs of the post-transition maintenance phase.

Example: We need one resource for the maintenance phase.

3.2.4 Business Planning Needs

Java CAPS can help you facilitate and improve your overall business processes. Assess your needs in these areas by asking the following questions:

What are our record-keeping and documentation needs?

Make sure that you set up a system for documenting your Java CAPS operation.

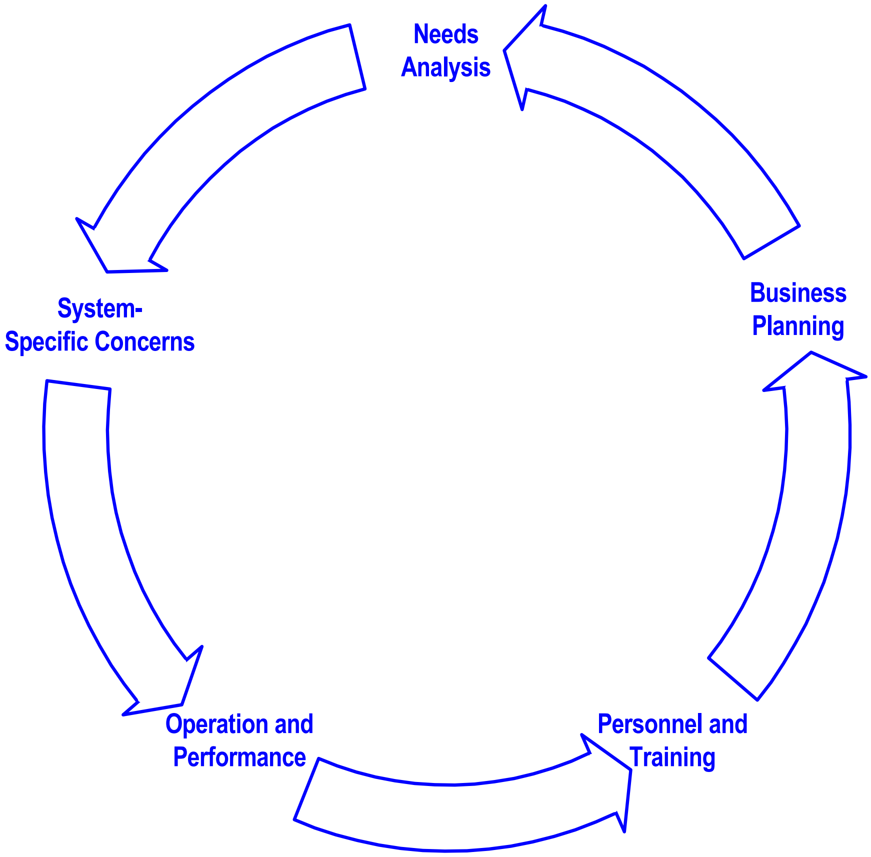
Example: We must put a new methodology in place to document and diagram the total operation of our Java CAPS operation. In addition, we must keep complete records on that operation.

How do we create a deployment road map?

Plan your deployment well. Choose a deployment project team, and be sure to document your plan in writing. For a small deployment, one person might be able to perform this task. Flowcharts and system diagrams can be helpful. See **"Deployment Planning" on page 26**.

Figure 5 shows a diagram of the information-gathering cycle in the deployment project's requirements analysis phase.

Figure 5 Information-Gathering Cycle in Requirements Analysis Phase



As you continue the analysis process, allow the results to feed back into your overall analysis. If necessary, repeat the process to fine-tune the information that you have gathered. This approach helps to ensure the accuracy and usability of the requirements that you collect.

Once we have the information, what do we do with it?

Complete the process of documenting and organizing your information as correctly and comprehensively as possible. When you finish the requirements analysis phase, you use this information to help you with the next phase: deployment planning. Much of the information will be added to the planning documents.

3.3 Deployment Planning

The deployment planning phase is the second phase in your deployment project. In this phase, you create a road map of what the deployment will look like. You include such criteria as resources, schedules, goals, and objectives. The main purpose of this phase is to initiate the project, define the integrated system to be developed, create top-level design documents, and create a formal project plan or road map.

In creating your road map, you provide a detailed description of the integrated system to be developed. This plan serves the following purposes:

- Designing what your future system looks like
- Showing you the resource allocation needed to implement the design

Planning enables you to find out where you want to go and how to get there. When necessary, you can obtain help from Sun Services and other Sun representatives. Thorough, comprehensive planning helps to ensure a successful deployment project.

The major steps in deployment planning are:

- Determining overall objectives
- Initiation steps
- Creating the planning documents

3.3.1 Determining Overall Objectives

This step of the deployment planning phase includes the following operations:

1. Achieve a consensus on the project’s overall functionality and scope.
    - Set up technical and functional teams or roles to handle individual phases and aspects of the deployment.

Note: *For a small deployment, one person might be able to handle the tasks of a team.*

- Ensure that the functionality is clearly stated and agreed on.
- Document the functionality and scope of the project based on analysis information, and match this information against the scope of the project as stated in the Approved Proposal.
- If necessary, resolve any differences between the Approved Proposal scope and your prepared analysis and requirements information (see **“Requirements Analysis” on page 22**).

2. Create a general model of what the system will do. This model:
    - Serves as the foundation architectural plan for all Java CAPS design.
    - Consists of diagrams and supporting documentation that represent the design strategy for any required Java CAPS interfaces.

3. Set up a design and development team and provide this team with an understanding of the application domain. In addition, provide the team with the top-level design documents.
4. Formulate a basis of validation of the final product during acceptance testing. This validation process includes the testing required to validate the functionality of the system and that it works as stated in the Approved Proposal.

3.3.2 Initiation Steps

The deployment project begins in earnest with the following tasks.

Hold a Project Kick-off Meeting

This meeting brings together all the members of the deployment project team. The analysis tasks and responsibilities assigned to each resource are also identified. The purpose of this task is to outline the reporting structure for the project and identify with whom the Project Manager communicates to ensure that other tasks in the project are being completed as planned. In addition, documentation standards are established.

Ensure That Hardware and Software Are Ready

Ensure that your hardware and software are in place and ready for the Java CAPS installation. This process includes ensuring that the Java CAPS software is fully supported on your hardware platform and operating system, and that the software has been shipped.

Complete Installation Test, Installation, and Checklist

The Java CAPS installation task is completed during this phase to ensure that there are no issues with your technical environment. You can use a deployment checklist to detail the exact hardware and operating systems where the installation will be performed.

This task includes the following steps:

- The total Java CAPS environment must be installed and tested. The deployment checklist is updated to identify what items were completed and to document any outstanding issues that prevented items from being completed.
- The hardware, software, and network requirements (current and planned) are identified and verified.
- The end-to-end communications with your other systems are also tested to ensure that communications are set up correctly and systems are exchanging messages correctly according to the communication protocol being invoked.
- Any communication with other businesses or trading partners are tested in the same way.

For detailed installation instructions, see the *Java Composite Application Platform Suite Installation Guide*.

Establish the Change Management Process

Change management is a critical factor during all phases of the project. Change management identifies and tracks all changes for a project that depart from the original deployment plan. All changes must be identified and tracked because many small changes can impact a deployment project in the same way as a more easily identifiable large-scale change. Tracking all changes enables the Project Manager to plan and control a project and keep track of all changes to the project's scope.

3.3.3 Creating the Planning Documents

In this step of the deployment planning phase, the following documents are created:

- Deployment project plan
- Functional requirements specification
- Technical requirements specification
- Initial test plan

Deployment Project Plan

The deployment project plan establishes a baseline reference plan. It is your road map for the deployment project. The roles and responsibilities of each organization, the schedule of tasks, and any estimates must be defined in this plan.

Be as detailed as possible. Any project risks must be assessed and documented. Your resources are budgeted using this plan (or validated if you have already created a budget).

The deployment project plan must be reviewed and agreed on by all of the organizations involved in the project. The plan must be communicated to all affected organizations.

Table 2 describes the contents of this document.

Table 2 Deployment Project Plan

Contents	Description
Scope of work	This item must be based on the purchase contract, Approved Proposal, or any equivalent document.
Project organization	Include the deployment project team, development organization, review organization, and any external organizations involved in the project. The roles and responsibilities must be clearly defined.
Delivery schedule	Include the schedules for all specified deliverables, including the final delivery date after all validation and verification tasks are complete.
Estimates	This section includes a work breakdown structure (WBS).

Table 2 Deployment Project Plan (Continued)

Contents	Description
Overall schedule	This section contains a schedule for all deployment project tasks including resource assignments; key phase completion milestones must be indicated. This schedule will be further elaborated by developing various phase work plans.
Resource requirements	Include the manpower, hardware, and software resources required for the pre-transition testing phase.
Issues and risks	Potential project issues and risks must be identified, and contingency plans must be drawn up for any risks.
Organizational interfaces	The dependencies on other projects and information needed from other organizations must be clearly identified, documented, and conveyed to any affected parties.

Functional Requirements Specification

In creating the functional requirements specification, you identify and analyze your specific system requirements. The behavior of the various application components (messages, process, and associated data) are carefully analyzed and documented.

The functional requirements specification and the technical requirements specification help form the basis for system design and final project acceptance.

The typical tasks in creating this document are:

- Study and identify system requirements to derive the business process functionality required and identify the system architecture needed to meet functional requirements.
- Create an architecture model to show the proposed integrated system.
- Create interface models to define interface requirements.

Table 3 describes the contents of this document.

Table 3 Functional Requirements Specification

Contents	Description
Statement of requirements	Define the objectives that you want the project to meet.
Proposed architecture	Show a summary design model of the sending and receiving systems, eWay Adapters to be used, and interfaces that take place.
Proposed directory structure and messages that trigger processing	Provide a map of the external sending/receiving systems, directory structures, and the business/processing messages, including the Java CAPS processing that will be initiated by the messages.
Exception processing	Define requirements for processing errors and exceptions.
Constraints	Define data volumes, performance, and any backup/archive requirements.
Interface diagrams	Produce a diagram for each proposed interface, showing the sending/receiving system, message processing, and any interdependencies.

Table 3 Functional Requirements Specification (Continued)

Contents	Description
Hardware/software diagrams	Show the hardware/software environment and high-level related schematics for development, test, and production systems.

The general design model provided by this document forms the basis of the system design and development phase.

Technical Requirements Specification

In creating the technical requirements specification, you identify and analyze your specific technical requirements. The behavior of the various application components (messages, process, and associated data) are carefully analyzed and documented.

The functional requirements specification and the technical requirements specification help form the basis for system design and final project acceptance.

Table 4 describes the contents of this document.

Table 4 Technical Requirements Specification

Contents	Description
Hardware/software model	Define the environment in which Java CAPS will process.
Security and system availability	Define the requirements for security, system availability, and the technology being used to meet these requirements.
Additional requirements	Any additional related requirements.

Initial Test Plan

A high-level test plan must be produced, highlighting the testing tasks to be performed during each phase. This document specifies the test approach, the types of tests to be carried out, and the organization responsible for carrying out the tests for each test phase.

Note: *The test plan is enhanced during the system design and development phase (see* **Chapter 4 "System Design and Development"***). The actual testing is performed during the pre-transition testing phase (see* **Chapter 5 "Testing, Transition to Production, and Maintenance"***).*

Table 5 describes the contents of this document.

Table 5 Initial Test Plan

Contents	Description
Test plan	Contains a general description of the testing phase. Ideally, this plan is created by an independent test team or role based on your requirements for the testing of applications and their process for promoting applications to production.
Test phases	Includes unit testing, integration testing, and acceptance testing.
Test approach	Details whether there will be manual or automated testing and the validation process for each test performed.
Organization	Includes the testing team or role (functional and technical).
Schedule	Defines the system availability for test data and system resources needed for the different test phases.
Resource requirements	Defines system, individual, and team resources needed for the test phases.

Going Forward

When the members of your management and deployment project team agree that these objectives have been met, you have finished the deployment planning phase. You have created a complete deployment road map (the deployment project plan), as well as some general designs for your completed system.

The next chapters in this guide describe the following topics:

- **System Design and Development:** For a discussion of the next deployment phase, see Chapter 4 **"System Design and Development"**.
- **Testing, Transition to Production, and Maintenance:** For a discussion of the pre-transition testing, transition to production, and post-transition maintenance phases, see Chapter 5 **"Testing, Transition to Production, and Maintenance"**.
- **Hardware Needs:** An important part of planning for your system and deployment is how to determine your hardware requirements. If you need additional information on planning for and determining your system's hardware requirements, see Chapter 6 **"Determining System Requirements"**.

Chapter 4

System Design and Development

This chapter describes the third phase of deploying Java CAPS: system design and development.

What's in This Chapter

4.1 Introduction

When the requirements analysis and deployment planning phases have been completed, your next major step is the system design and development phase. During this phase, you work on the essential system architecture that implements your business plans and processes.

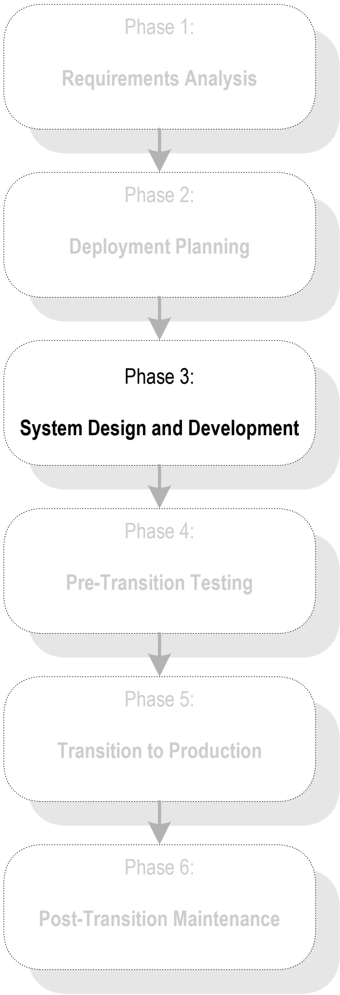
This phase requires a series of successive refinements applied to your initial summary plan (the functional requirements specification). Your design starts with the broadest view of the system and then proceeds to the details. This top-down approach results in the most effective application of the technology to the integration of your existing systems and applications.

The system design and development phase includes the following steps:

- Planning general hardware configuration
- System design methodology
- Software development
- System optimization

Figure 6 shows where the system design and development phase fits into the overall Java CAPS deployment.

Figure 6 System Design and Development Phase



4.2 Service-Oriented Architecture

Java CAPS enables you to develop composite applications on top of a *service-oriented architecture*.

4.2.1 Overview

In a service-oriented architecture, a group of software components offers functionality to other components. A service consumer makes a request, and a service provider sends a response. Some services are high level (for example, verify the insurance eligibility of a patient). Other services are more technical (for example, transform a message from an external format to an internal format).

The service provider publishes an interface that service consumers interact with. The implementation details are hidden from the consumers.

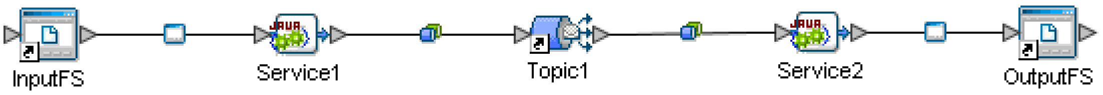
In Java CAPS, you can create various types of services, including:

- Java Collaborations
- Business Processes
- Page Flows
- eTL Collaborations

These services are created independently in Enterprise Designer and stored in the Repository. You then use a Connectivity Map to link the services and other component types (such as topics and external applications) together.

Figure 7 shows a simple Connectivity Map that contains two services, both of which are Java Collaborations.

Figure 7 Connectivity Map with Two Services



One of the key benefits of a service-oriented architecture is reusability. Services can be combined in different ways. For example, Service1 in the preceding Connectivity Map might also appear in a different Project.

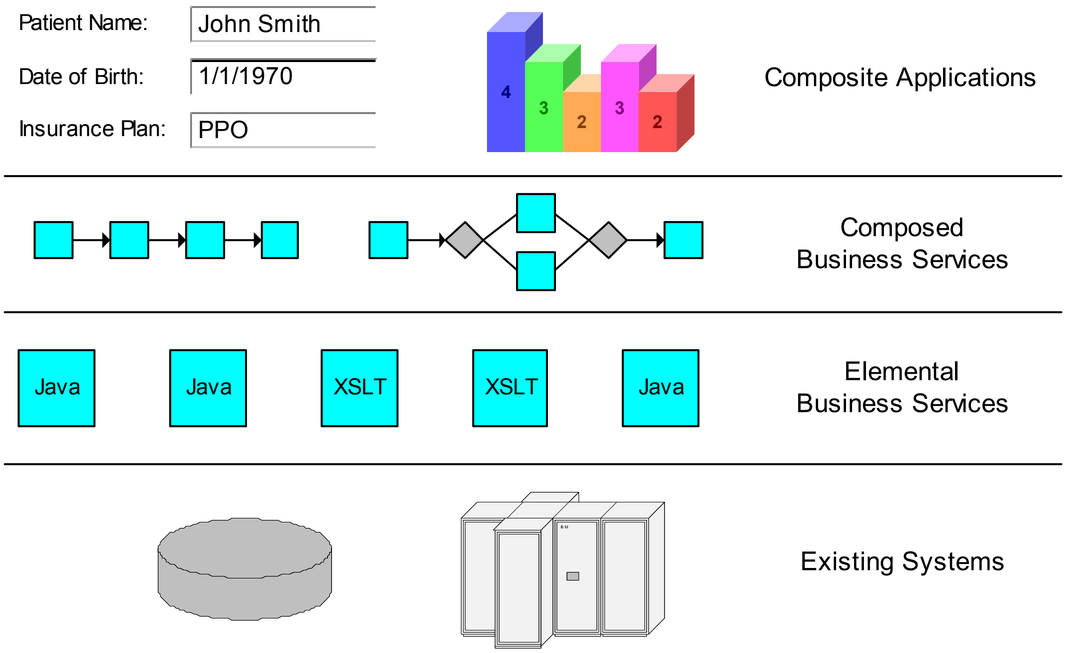
Another benefit is location transparency. The service consumer does not need to know where the service provider is located on the network, and vice versa.

A third benefit is that services in a service-oriented architecture are loosely coupled. This concept means that changing a service does not necessarily require changes to the components with which the service interacts.

4.2.2 Architecture Layers

Figure 8 shows the four layers of the service-oriented architecture in Java CAPS.

Figure 8 Architecture Layers



The following subsections describe each layer, starting with the lowest layer and moving upward.

Existing Systems

The Existing Systems layer consists of back-end programs, such as databases, packaged applications, external trading partners, and legacy systems.

The back-end programs might use technologies that are incompatible with each other. For example, information about a customer might be located in an Oracle database (used by one company division) and in an IBM DB2 database (used by another company division).

Elemental Business Services

The Elemental Business Services layer contains services that perform discrete, fine-grained tasks, such as:

- Verify customer credit
- Determine product availability
- Calculate shipping charges

eGate Integrator enables you to define these services, which can be based on Java or XSL Transformations (XSLT). Object Type Definitions (OTDs) define the various formats of messages that the services process.

eTL Integrator enables you to extract, transform, and load bulk data between files and databases.

eView Studio provides methods that you can use in Java Collaborations to specify how data is processed into the master index database.

eWay Adapters enable the services to interact with the different technologies in the Existing Systems layer. Thus, the adapters shield the incompatibilities between back-end programs from this layer.

Composed Business Services

The Composed Business Services layer combines services in the lower layer into larger-grained services that represent a specific business process. Examples of services in this layer are:

- Process customer order
- Execute revenue process

In many scenarios, conditional branching is required. An example of conditional branching is: if condition 1 is true, invoke service 1; otherwise, invoke service 2. The second Composed Business Service in Figure 8 includes a conditional branch.

You can incorporate human tasks into the flow of a business process. The flow waits for a person to complete the task before proceeding further.

Some business processes are long running, which means that under normal circumstances they can take days, weeks, or months to complete.

eInsight Business Process Manager contains a wide variety of design elements (including elements that support conditional branching) for creating business processes in this layer.

Note: *You can also use eInsight to create an Elemental Business Service.*

eView Studio enables eInsight to access the master index database.

eXchange Integrator includes the concept of a B2B protocol, which represents a business process in trading partner scenarios.

Composite Applications

The Composite Applications layer contains applications that are based on the services in the lower layers.

The information that appears in the user interface might be combined from multiple back-end systems. Without the composite application, the user would need to obtain the desired information by accessing each system independently.

eVision Studio enables you to create web-based user interfaces for composite applications.

eBAM Studio enables you to create dashboards that illustrate Key Performance Indicators (KPIs). These dashboards can help users to visualize the current state of the enterprise.

eIndex Single Patient View enables you to create a single view of person information across an enterprise.

4.2.3 Core Technologies

Table 6 describes various technologies that Java CAPS uses to implement a service-oriented architecture.

Table 6 Core Technologies

Standard	Description
Java 2 Platform, Enterprise Edition (J2EE)	An architecture for developing, deploying, integrating, and managing enterprise applications.
Extensible Markup Language (XML)	A form of Standard Generalized Markup Language (SGML) that is widely used for data exchange and configuration files.
XSL Transformations (XSLT)	A language for transforming XML documents into other XML documents.
Simple Object Access Protocol (SOAP)	An XML-based specification for exchanging messages in a distributed environment, typically over HTTP.
Web Services Description Language (WSDL)	An XML-based language for describing the functionality that a web service offers.
Universal Description, Discovery, and Integration (UDDI)	A specification that enables web services to be published in a registry. Service consumers can search the registry for service providers.
Business Process Execution Language for Web Services (BPEL4WS)	A notation for describing the behavior of business processes.

4.3 System Design

The three basic steps in designing a Java CAPS deployment are:

- Identify all of the external systems to be connected
- Define the configuration of Java CAPS components
- Define the configuration of hardware and network connections

4.3.1 Identifying External Systems

The first step is to identify all of the external systems to be connected to each other through the system. The resulting set of interconnected systems is called the *communication topology.* The communication topology exists without regard for the hardware hosts where components execute and without regard for the format of data exchanged between systems.

4.3.2 Configuring Java CAPS Components

The second step is to define a configuration of Java CAPS components (for example, eWay Adapters, Collaborations, and Message Servers) to run on the respective hosts in the hardware topology. The component arrangement is called the *component topology*.

Choose the simplest Object Type Definitions and Collaborations. Defining the most efficient component topology depends on the relationship of data formats. Therefore, defining object types is an integral part of designing the component topology.

4.3.3 Hardware and Network Connections

The third step is to define a configuration of hardware and network connections that enable the external systems to communicate as required by the communications topology. This hardware configuration is called the *hardware topology*.

Java CAPS is designed to run as a distributed system with central management. Only network performance and the demands on each host are relevant considerations in defining hardware topology. Because of the distributed architecture of the total system, the hardware topology is not rigidly defined. It can be adjusted as needed when system demands change. For example, increased demands on the Java CAPS environment can be met by distributing processing across more CPUs.

For more information on how to determine and meet your hardware requirements, see Chapter 6 **"Determining System Requirements"**.

4.4 System Development

This section describes a variety of system development topics, including user management, naming conventions, and project teams with multiple developers.

4.4.1 Creating Users

Java CAPS users are divided into the following categories:

- Repository
- Logical Host
- Enterprise Manager

Repository users can use Enterprise Designer to create the necessary Project and Environment components for a Java CAPS solution.

Java CAPS has one predefined Repository user: **Administrator**. The **Administrator** user has unlimited power to create other users. Each user has one or more *roles*, which specify the tasks that the user can perform. Every user has the predefined **all** role. The predefined **administration** role enables the user to perform uploads in the Java CAPS Installer and to create Repository branches.

When creating a user, the **Administrator** user should not give the user a more powerful role than necessary.

If your organization stores information about the project team members in an LDAP server, you can integrate with the LDAP server. The following LDAP servers are supported: Sun Java™ System Directory Server, Microsoft's Active Directory, and OpenLDAP.

Logical Host users can access Java CAPS applications that are running in a Logical Host.

Enterprise Manager users can monitor running Java CAPS applications.

For detailed information about managing users of the Repository, Logical Host, and Enterprise Manager, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

4.4.2 Naming Conventions

Create a set of naming conventions early in the system design and development phase. Naming conventions help to ensure readability and maintainability. Consider using the prefixes and suffixes described in this section. Additional standards might include:

- Start the main portion of a Service name with a verb
- Include the data format or external system in an Object Type Definition name.

Prefixes

Add a prefix to each component name to identify the component type. For example, you could use the prefix **prj** as an identifier for Projects.

Table 7 contains suggested prefixes for Project components.

Table 7 Suggested Prefixes for Project Components

Component	Prefix	Examples
Project	prj	prjFileTransfer prjWebOrder
Business Process	bp	bpProvideQuote bpOrderToInvoice
Collaboration Definition (Java)	jcd	jcdOrderIn jcdCheckInventory jcdOrderOut
Collaboration Definition (XSLT)	xcd	xcdOrderIn xcdCheckInventory xcdOrderOut
Connectivity Map	cm	cmProvideQuote cmWebOrder
Deployment Profile	dp	dpWebOrderDev dpWebOrderTest dpWebOrderProd
External Application	ea	eaFileIn eaFileOut
Object Type Definition	otd	otdEmployeeInfo otdOraOrders
Page Flow	pf	pfVacationRequest
Page Layout	pg	pgCredit
Page Link	pl	plMortgage
Queue	que	queOraOrders
Scheduler	sch	schNightRun
Service	svc	svcTransfer svcReadFromFile svcWriteToFile
Topic	tpc	tpcRequest

Table 7 Suggested Prefixes for Project Components

Component	Prefix	Examples
Web Service Definition	wsd	wsdStockQuote
XML Schema Definition	xsd	xsdWarehouseOrder

If you create subprojects as an organizational aid (for example, as the container for a large number of Collaboration Definitions or Object Type Definitions), then you can omit the **prj** prefix.

Figure 9 Naming Subprojects

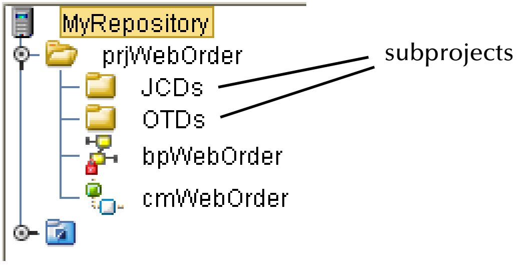


Table 8 contains suggested prefixes for Environment components.

Table 8 Suggested Prefixes for Environment Components

Component	Prefix	Examples
Environment	env	envWebOrderDev envWebOrderTest envWebOrderProd
Logical Host	lh	lhWebOrderDev lhWebOrderTest lhWebOrderProd
Sun SeeBeyond Integration Server	is	isWebOrderDev isWebOrderTest isWebOrderProd
Sun SeeBeyond JMS IQ Manager	ms	msClaimData
External System	es	esOraIn esOraOut

Suffixes

You might want to add a suffix to components that have development, test, and production versions. In Table 7, the examples for the Deployment Profile use this convention. In Table 8, the examples for the Environment, Logical Host, and Integration Server use this convention.

You can also use suffixes to indicate whether External Applications and External Systems are inbound or outbound.

4.4.3 Project Teams with Multiple Developers

This section contains guidelines for project teams with multiple developers.

Subprojects and Multiple Projects

If you have a large development team, you might find it useful to divide the work using one of the following approaches:

- Create multiple subprojects within a top-level Project. Choose this approach when the subprojects have common components.
- Create multiple Projects.

When using the first approach, the developers must consider the order in which components are created. For example, assume that a Collaboration Definition called **jcdCheckInventory** is used by two subprojects. The developers might want to work on this Collaboration Definition before working on Collaboration Definitions that are not used by multiple subprojects.

Version Control and Access Control Lists

The version control system and Access Control Lists (ACLs) enable a project team to limit access to components.

With the version control system, a developer can work on the new version of a component while other team members use the most recent stable version. For example, assume that version 1.4 of an Object Type Definition called **otdEmployeeInfo** is considered to be stable. While the developer works on version 1.5, the other team members can use version 1.4.

Important: *More than one person concurrently using the same user name will circumvent the version control system, and one person's work can be overwritten by another. Ensure that all personnel using Enterprise Designer use unique names.*

ACLs become more important as the size of a Project grows, or when you divide the work as described in **"Subprojects and Multiple Projects"**. They enable you to control which users have read only or read/write access to a component, and thus reduce the possibility of deliberate or accidental changes.

Ensure that the development team fully understands how the version control system and ACLs work. The *Sun SeeBeyond eGate Integrator User's Guide* describes how to use the version control system. The *Sun SeeBeyond eGate Integrator System Administration Guide* describes how to create and modify ACLs.

Logical Hosts

Ideally, each developer installs and uses a Logical Host on the same computer as Enterprise Designer. Thus, developers can run the Logical Host without interfering with the work of other developers.

If a development computer does not have enough resources to run a Logical Host, you can try other approaches.

Note: *Typically, the amount of system resources used by a developer's Logical Host is less than the amount of system resources used by a Logical Host in the test and production environments.* **Chapter 6 "Determining System Requirements"** *lists the minimum system requirements for the Logical Host on various operating systems.*

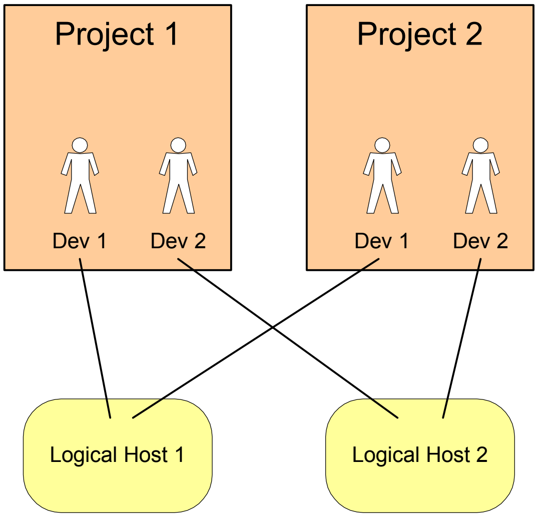
One approach is to share Logical Hosts. For example, assume that there are two Projects, each of which has two developers. Rather than creating four Logical Hosts (one for each developer), the teams could create two Logical Hosts:

- Logical Host 1 is shared by developer 1 from the first Project and developer 1 from the second Project.
- Logical Host 2 is shared by developer 2 from the first Project and developer 2 from the second Project.

Use this approach only when the two Projects are sufficiently independent from each other.

Figure 10 shows a diagram of this approach.

Figure 10 Sharing Logical Hosts Between Projects



4.4.4 Error Handling

In the requirements analysis and deployment planning phases, you define the requirements for processing errors and exceptions.

In the system design and development phase, you use various Java CAPS features to implement these requirements. For example:

- When creating a business process in eInsight, you can configure the process to catch all exceptions or certain exceptions that you specify. In addition, eInsight provides a Compensation Handler element, which enables you to add sophisticated rollback logic to Straight Through Processing (STP), fast processes, and long-running transactions.
- eVision enables you to return preconfigured web pages if a requested page cannot be found or if there is a serious problem with the business process. These preconfigured web pages correspond to the HTTP response codes 404 and 500, respectively.

4.4.5 Variables and Constants

Variables and constants enable you to avoid hard-coding environment-specific information such as file names and database names.

A *variable* functions as a placeholder. The value is determined when you create a Deployment Profile. You can define variables at the Project level.

Assume that you create a Connectivity Map that includes an inbound File eWay Adapter. The eWay Adapter has an environment property called **Directory**, which specifies the folder that the eWay Adapter polls for input files. Instead of entering a value such as **C:/input**, you can select a variable that you previously defined (for example, **VARIABLE_FILE_INPUT**). When you create the Deployment Profile, you assign a value to the variable.

A *constant* is a name/value pair that is visible across a Project or an Environment.

For more information about variables and constants, see the *Sun SeeBeyond eGate Integrator User's Guide*.

4.4.6 Multiple Environments

Create separate Environments for the development, test, and production phases.

Each Environment should have a corresponding Deployment Profile:

- The development Deployment Profile maps Project components to the development Environment.
- The test Deployment Profile maps Project components to the test Environment.
- The production Deployment Profile maps Project components to the production Environment.

4.4.7 Multiple Runtime Components

Under normal circumstances, you should not need to create multiple runtime components. In other words, you can run a single Logical Host that contains a single Sun SeeBeyond Integration Server and/or a single Sun SeeBeyond JMS IQ Manager.

The Integration Server is designed to scale up to meet higher demands. Note that you can modify Integration Server properties associated with scalability from the Integration Server Administration tool.

However, there are situations when you may want to create multiple runtime components.

Assume that you are running a Logical Host on a computer that has multiple CPUs. If you discover that some of the CPUs are being underutilized, check to see whether creating an additional Logical Host improves performance.

Multiple Logical Hosts can be useful in isolating deployed Projects from each other. For example, if you are running two Projects on the same Logical Host and Project 1 requires the Logical Host to be restarted, then Project 2 is also restarted. Using multiple Logical Hosts would prevent Project 2 from being restarted.

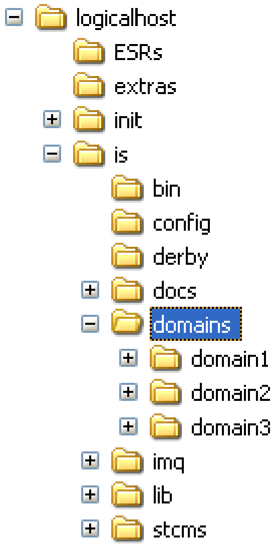
In general, be sure to test and monitor your deployed Projects so you can determine how effectively the runtime components are handling the load.

4.4.8 Multiple Logical Host Domains on a Computer

You only need to install the Logical Host once on a computer. You can create multiple domains within the Logical Host.

For each domain, a directory is automatically created with the same name as the domain. Figure 11 shows an example of this behavior on a Windows computer. The Logical Host contains three domains: **domain1**, **domain2**, and **domain3**.

Figure 11 Multiple Domains on a Single Computer (Windows)



Here is an example for a UNIX computer:

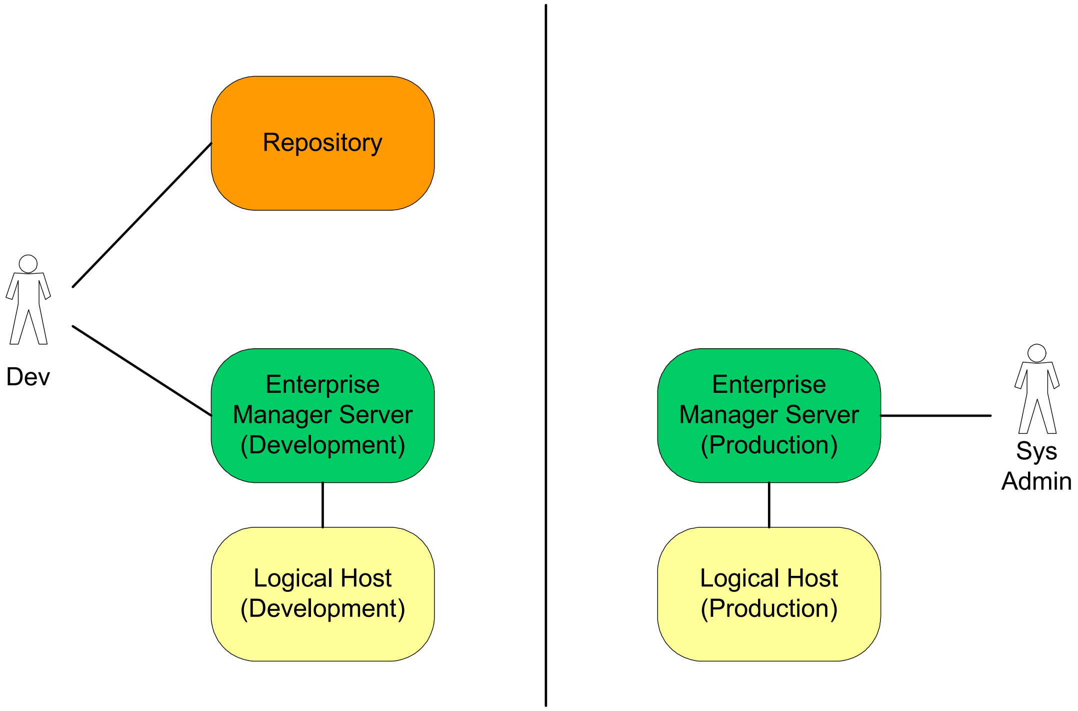
```
/home/JavaCAPS51/logicalhost/is/domains/domain1
/home/JavaCAPS51/logicalhost/is/domains/domain2
/home/JavaCAPS51/logicalhost/is/domains/domain3
```

If you plan to run more than one Logical Host domain at the same time, be sure to configure the port numbers so that they do not conflict with each other. For example, do not set the administration port number for both domains to 18000.

4.4.9 Development and Production Versions

Create separate versions of the Logical Host and the Enterprise Manager Server for the development and production phases.

Figure 12 Development and Production Versions


Repository
Dev
Enterprise Manager Server (Development)
Logical Host (Development)
Enterprise Manager Server (Production)
Logical Host (Production)
Sys Admin

Because the Repository is not needed during the production phase, Figure 12 shows only one Repository installation.

4.5 Optimizing Your System

This section provides information on the following system optimization topics:

- Tuning for high-volume scenarios
- Tuning the eInsight Engine
- IBM AIX tuning and troubleshooting

4.5.1 Tuning for High-Volume Scenarios

If you deploy an application that processes a large number of transactions, consider the following guidance.

Outbound JMS Clients include a property called **Maximum Pool Size**. To avoid exhausting the pool of connections, you might need to increase the default value of this property (for example, to 128). You can edit this property in the Connectivity Map Editor of Enterprise Designer.

If you are running a high-volume JMS application on a Windows platform, and the **TcpTimedWaitDelay** registry parameter is set to the default value, then the application might use all of the ephemeral ports. Try decreasing the value, so that TCP can release closed connections more quickly.

Note: *You can add the* **MaxUserPort** *registry parameter to increase the number of ephemeral ports.*

4.5.2 Tuning the eInsight Engine

This section describes various configuration properties that affect the performance of the eInsight Engine. The suggested ranges are for a typical Project. Each Project might need customized tuning for the best performance. You could start with low values for these properties, and increment the values until the throughput peaks and resource usage is optimal.

Note: *The Sun SeeBeyond eInsight Business Process Manager User's Guide describes how to access and modify the configuration properties from Enterprise Designer.*

Each Project has its own eInsight Engine.

The eInsight Engine creates *work items* and submits them to the application server for execution. This approach enables the application server to pool threads efficiently and to have more control over the runtime environment. The eInsight Engine does not need to create or manage threads directly.

There are two categories of work items: invoke activities, and other types of activities.

The eInsight Engine allows you to limit the maximum number of work items that are submitted for execution by the application server at a given time. In addition, the engine allows you to limit the maximum number of invocations that can be made of the total available work items submission limit. This feature allows the engine to keep processing other activities (if available) and ensures that the total thread pool is not used up in invocations.

Max Concurrent Instances Property

The **Max Concurrent Instances** property specifies the maximum number of instances of a Business Process in a Project that can be processed by the eInsight Engine at a given time. If the engine receives additional requests, then these requests are placed in a waiting state. As soon as any of the instances being processed is completed, one of the waiting requests is obtained for processing.

If a Project has multiple Business Processes, then you can configure each Business Process to have a different number of maximum concurrent instances.

A higher value for this property results in higher memory requirements.

Memory requirements are also based on the type of Business Process. Assume that two Business Processes have the same value for this property. The Business Process that has more defined variables will require more memory.

The default value is 40, which is sufficient for most scenarios. The suggested range is from 40 to 1000.

Receive Timeout Property

The **Receive Timeout** property can affect the performance of certain Projects.

- If the maximum concurrent instances for a Business Process has been reached and the engine receives another request, then the new request is placed into a waiting state. This parameter defines how long the request can be in the waiting state. After this duration, a timeout occurs and the incoming request thread is returned.
- If a Business Process is defined with correlation, it is possible that the instance to be correlated for the incoming request has not been created. In this situation, the incoming request is placed into a waiting state for the Receive Timeout period. When the correlating instance is available, this request is picked up. If a timeout occurs before the correlating instance is available, then the incoming request thread is returned.
- If a Project has multiple Receive Activities and the request for the second Receive Activity arrives before the request for the first Receive Activity, then the request for the second Receive Activity is placed into a waiting state for the Receive Timeout period. The request is notified as soon as the Business Process instance is created for the first Receive Activity.
- If the eInsight Engine has not completely started and an incoming request arrives, then the request is placed into a waiting state for the Receive Timeout period.

The default value is 15 seconds. The suggested range is from 1 second to as needed. Be sure to enter the value in milliseconds, rather than seconds.

Work Item Submit Limit Property

The **Work Item Submit Limit** property specifies the maximum number of work items that the eInsight Engine can submit to the application server for execution at a given time.

The default value is 500. The suggested range is from 50 to 750.

Invocation Allocation Ratio Property

The **Invocation Allocation Ratio** property is closely related to the **Work Submit Limit** property. This property specifies the percentage of threads that can be used for invoke activities, as opposed to other types of activities.

If a Business Process makes a lot of invocations to other processes, then you can increase the ratio to improve performance.

Important: *Do not set this value to 100 percent. If all of the threads are used and the Business Process needs to execute other Activities, then a deadlock situation can occur.*

The default value is 80 percent.

The suggested range depends on the partners that the Business Process is interacting with. For example, if the **Work Submit Limit** property is set to the default value of 500 and the Business Process has a JMS partner with a connection pool size limit of 32, then the highest suggested value for the **Invocation Allocation Ratio** property is 6 percent. The processing needs of the partner might require you to enter a value that is lower than the highest suggested value. Alternately, the partner could increase its pool size limit.

Database Connection Max Idle Time Property

The **Database Connection Max Idle Time** property is used with Business Process instances that have been configured for persistence. This property specifies the maximum number of seconds that a physical connection can remain unused before the connection is closed. If the value is zero (0), then there is no limit.

The default value is 600.

Database Connection Pool Size Property

The **Database Connection Pool Size** property is used with Business Process instances that have been configured for persistence. This property specifies the maximum number of database connections that the eInsight Engine can cache for performance.

The default value is 60. The suggested range is from 10 to 100.

Database Connection Retries Property

The **Database Connection Retries** property is used with Business Process instances that have been configured for persistence. This property specifies how many times the eInsight Engine attempts to establish a database connection after a connection failure.

The default value is Infinity.

Database Connection Retry Interval Property

The **Database Connection Retry Interval** property is used with Business Process instances that have been configured for persistence. This property specifies the milliseconds of pause before each reattempt to establish a database connection after a connection failure.

The default value is 10000. Be sure to enter the value in milliseconds, rather than seconds.

Database Connection Steady Pool Size Property

The **Database Connection Steady Pool Size** property is used with Business Process instances that have been configured for persistence. This property specifies the initial and minimum number of database connections that the eInsight Engine maintains in its database pool.

The default value is 20.

4.5.3 IBM AIX Tuning and Troubleshooting

The IBM web site includes a resource called developerWorks, which provides valuable information that can help you with tuning and troubleshooting on IBM AIX systems.

For example, the following series of articles provide tips and techniques for maximizing Java performance on IBM AIX:

http://www-106.ibm.com/developerworks/eserver/library/es-Javaperf1.html

The following article describes how to find out whether garbage collection is finely tuned and how to resolve any issues:

http://www-106.ibm.com/developerworks/library/i-gctroub/

These URLs were accurate as of this Java CAPS release.

Chapter 5

Testing, Transition to Production, and Maintenance

This chapter describes the final three phases of deploying Java CAPS: pre-transition testing, transition to production, and post-transition maintenance.

What's in This Chapter

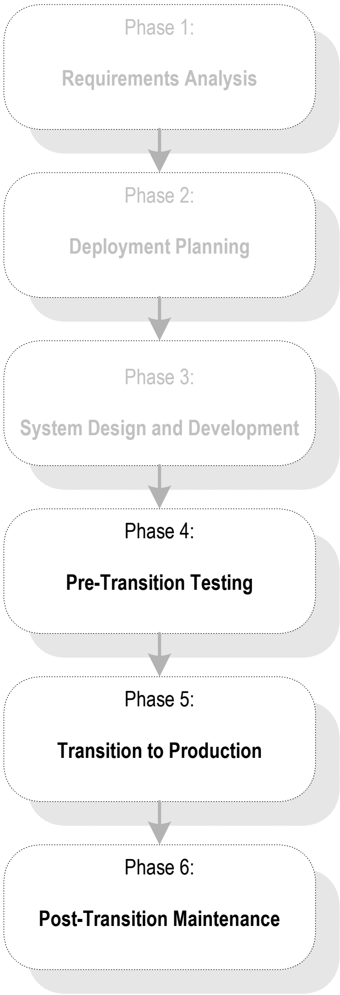
5.1 Introduction

The final three phases of deploying Java CAPS are:

- **Pre-transition testing:** The system must be fully tested before it is transitioned to a production environment. This phase includes unit testing, integration testing, and acceptance testing.
- **Transition to production:** After the system is fully tested, it must be migrated to the production environment. This chapter covers the procedures and considerations for performing the transition to production.
- **Post-transition maintenance:** Once the system has been migrated to the production environment, the system must be monitored for correct performance, possible errors, and the need for changes. System monitoring is a critical step in the long-term success of your deployment project. Routine checks and fine-tuning help to establish long-term performance benchmarks and aid in identifying undesirable changes.

Figure 13 shows where these phases fit into the overall Java CAPS deployment.

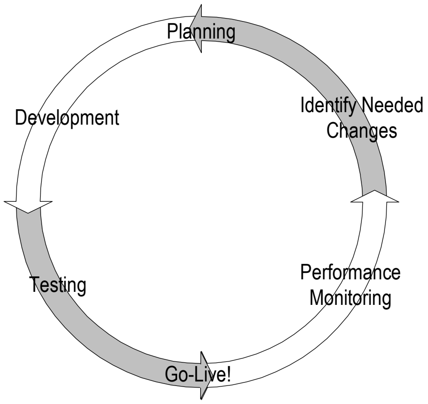
Figure 13 Testing, Transition to Production, and Maintenance Phases



An important part of the entire deployment project is change management. If changes are required, they must be processed through the same cycle of planning, development, testing, transition to production, and maintenance as the rest of the deployment.

Figure 14 illustrates this cycle of change management.

Figure 14 Change Management Cycle



5.2 Pre-Transition Testing

An essential part of the implementation of any complicated system is thorough testing. Table 9 describes the types of testing that must be performed.

Table 9 Testing Types

Type	Description
Unit testing	Testing individual components in isolation.
Integration testing	Testing groups of components together, up to and including the entire system.
Acceptance testing	Testing a completed system (or portion thereof) to ensure that it meets the requirements.

For the most part, unit and integration testing are done in the development phase, while acceptance testing is done as a final check before putting the system into production.

5.2.1 Testing Methodology

While how a system is tested varies, depending on the particulars of the specific system, certain methodologies apply to all system testing.

In general, you test the individual parts of the system before testing the entire system. Similarly, you test individual components in isolation before testing them in a broader context.

5.2.2 Test Plan

Planning for system testing begins with a careful examination of the requirements of the system. An initial test plan is created in the deployment planning phase. The test plan specifies how the system is tested and what requirements the system must meet before it is put into production.

The initial test plan can be enhanced during the system design and development phase.

The functional and technical requirements specifications outline the procedures used to conduct the tests, both at a component level and at an integrated system level. These specifications include:

- Type of data to use
- Expected output
- Who is responsible for the test

Type of Data To Use

The test plan specifies the type of data to use when testing the system. At both the component level and the integration level, be sure to work with data that is typical of the data that the system is designed to process. If possible, use real data from your pre-existing systems. Vary the data enough so that all possible types of processing implemented by the system are tested.

In addition to real-life typical data, use data designed to test the system’s error handling. This data might have to be specially constructed.

Testing the Output

The test plan includes specifications for:

- Proper error handling
- Transaction processing speed
- Correct routing of information
- Correct transformation of data
- Any other special requirements

Responsibility for Testing

In general, the responsibility for testing an individual component belongs to the developer of the component. The responsibility for the testing of the entire system might belong to the project manager or the technical lead for the project.

Acceptance testing is done by or in conjunction with people for whom the system is being created.

5.2.3 Unit Testing

Unit testing checks the individual parts of a larger system for correct functioning prior to integration testing.

Each component used in the system must be unit-tested and its functionality must be verified before the component can be used in the integrated system.

Unit testing is done as part of the development phase by the developer responsible for creating the component. If the functional or technical requirements specifications include a procedure for testing the component, then this procedure must be followed. If the functional and technical requirements specifications do not include a procedure for testing the component, then at a minimum the test must verify that the component performs as outlined in the specifications.

The tools used to test each part are different, depending on what the part is designed to do in the system. In addition, though all are designed to verify correct functioning, the methods employed vary, depending on the component being tested.

Take advantage of the validation tools in various Java CAPS products. For example:

- Enterprise Designer includes a Java Debugger that enables you to debug Java-based Collaboration Definitions.
- eInsight Business Process Manager includes a validation tool that enables you to check for errors in a Business Process model.

5.2.4 Integration Testing

Integration testing verifies how well the new components work together and with the existing Java CAPS infrastructure.

If the system is large and complex, you can break the integration testing into pieces designed to test a self-contained portion of the system.

Partial Integration Testing

Partial integration testing verifies correct data movement from one external system to another (that is, a complete data path).

For example, assume that you have a system that brings in data from a single external system and then sends the data to several other external systems. A partial test of this system would be to test whether the data can be sent to one of the external systems.

Complete System Testing

Complete system testing tests the entire system, including the interaction with all of the external systems. If the system is large and complex, this type of test requires a large amount of coordination. Set up this test to duplicate the actual production system. In many cases, it is used as a dry run prior to doing the actual acceptance test.

Performance Testing

Performance testing tests whether the system works fast enough. This type of testing must be done once a component or system is functioning correctly and transforming the data properly.

Many factors in combination affect performance. Speeding up one component might not speed up the performance of the entire system if the bottleneck is located elsewhere.

The exact requirement or goal in terms of the system’s performance must be specified in the test plan. Whether you meet the goal determines whether you pass the test.

An additional goal in performance testing is to find the bottlenecks in the system. Uncovering these bottlenecks enables additional system resources to be allocated intelligently in order to improve processing.

Speed Testing

This operation tests whether the system processes data fast enough. Ensure that the logging is set to the levels that will be used in production before doing a speed test. Higher-than-normal levels of logging can seriously degrade system performance and slow processing speed.

Stress Testing

This operation tests whether the system can handle the expected load. This type of testing attempts to overload the system with data to see whether or how it could fail. Effective stress tests include large volumes of messages, large-sized messages, and multiple concurrent connections. Many times, network bottlenecks can be uncovered this way. The test plan describes the methodology to use for stress testing.

5.2.5 Acceptance Testing

Acceptance testing is done before moving the system into production. This type of testing serves as a final check to prove to end users that the system performs according to plan.

Acceptance testing can be done for part of a system or for a complete system. If the entire system is not going into production at the same time, then you can test the portion of the system that is going into production.

The test plan specifies all conditions that the system has to meet before being put into production. In addition, the test plan specifies the people who must approve the system and who must be involved in the test.

5.2.6 Troubleshooting

The following tools are available for finding and correcting errors:

- Enterprise Manager
- Log files created by individual components
- Alert Agent
- SNMP Agent

Enterprise Manager

Enterprise Manager gives you control over the components in a Project. Enterprise Manager alerts you to the status of components (for example, whether they are running) and enables you to send commands to them (such as start or shut down).

For more information on using Enterprise Manager, see the following documents:

- *Sun SeeBeyond eGate Integrator System Administration Guide*
- *Sun SeeBeyond eGate Integrator JMS Reference Guide*
- *Sun SeeBeyond eInsight Business Process Manager User's Guide*

Log Files

To gain additional information, use the log files created by any component that fails.

When a component or system is not working, errors are written to the log file to help you diagnose the problem. In a normally functioning system, only the most serious errors (such as a shut-down component) are written to the log, because each entry written to the log uses system resources and slows processing. To learn more about a malfunctioning component, you can enable the system to log more information.

For more information on log files, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Alert Agent

The Alert Agent enables you to send a specified category of alerts to one or more destinations as the alerts occur. For detailed information, see the *Sun SeeBeyond Alert Agent User's Guide*.

SNMP Agent

The SNMP Agent enables you to forward alerts as SNMP version 2 traps to a third-party SNMP management system. For detailed information, see the *Sun SeeBeyond SNMP Agent User's Guide*.

5.3 Transition to Production

After verifying the performance and reliability of the system, the next step is to transition the system to the live production environment.

This phase consists of the following steps:

1. Create a production version of the Deployment Profile
2. Build the EAR file
3. Deploy the EAR file

The *Sun SeeBeyond eGate Integrator User's Guide* describes the first two steps. The *Sun SeeBeyond eGate Integrator System Administration Guide* describes the third step.

5.4 Post-Transition Maintenance

With your project up and running, it is important to monitor system activity, check for errors, and make any needed changes.

Figure 14 on page 53 shows the steps to take when changes are required. You must put changes through the same high scrutiny as the original system design.

5.4.1 Monitoring System Activity

Monitoring system activity is a critical step in the long-term success of the system. Routine checks help to establish long-term performance benchmarks. Such benchmarks are helpful in identifying undesirable changes. If you do not have a good feel for a healthy system's vital signs, you may have a more difficult time recognizing when your system needs attention.

Enterprise Manager

Enterprise Manager provides visual cues to let you know when a component needs attention.

Note: *By default, Enterprise Manager is timed out after three hours. You can disable the timeout.*

If you configure the Alert Agent or SNMP Agent, you can avoid having to run the Enterprise Manager continuously. The agent notifies you when the specified problem occurs.

Log Files

On a daily basis, review the runtime log files and examine any messages that indicate a major problem.

If you are using the Sun SeeBeyond Integration Server, then the server log file is the main log file. This log file uses the Java Logging API.

Note: *You can specify the minimum severity level that a logger outputs to the corresponding log file. Do not use the FINE, FINER, or FINEST levels during production.*

Many of the log files (such as **repository.log**) are configured to behave as a recirculating stack. Each file has a maximum size, and the oldest file is reused when the maximum number of files is reached. Java CAPS includes configuration files that enable you to change the maximum file size and the maximum number of files.

The other log files (such as **connection.log**) do not have configuration files. On a regular basis, check the size of these log files. If a file has grown too large, you can rename the file, delete the file, or move the file to a separate disk or computer.

Repository Backup

On a regular basis, back up the Repository. The frequency will vary depending on the project. One scenario is to back up the Repository once a week at a time of low system activity.

5.4.2 Implementing Changes

After a period of time, you may need to make changes to your project. Changes are common as the needs of your end users evolve and as additional external systems are added.

Implement changes by using the same process that was originally used to deploy your project. Consider the change management process illustrated in **Figure 14 on page 53**. Applying this same process of planning, configuration, testing, migration, monitoring, and re-evaluation helps to ensure a successful deployment.

5.5 Summary

The proper use of a lab environment provides an excellent opportunity to verify and refine the system configuration that was implemented earlier in the deployment process. Focusing on the deployment plan helps to ensure the smoothest possible deployment.

By thoroughly unit testing and system testing the system in the lab, costly errors can be avoided and the end users are more satisfied with the final results.

Going Forward

When the management team, the deployment project team, and the end users agree that the system is up and running according to plan, you have successfully finished the deployment.

If you have any questions about further system operation and maintenance, see the appropriate documents listed in **"Related Documents" on page 12** or contact Sun Microsystems.

Chapter 6

Determining System Requirements

This chapter offers guidelines to help you determine the system requirements for the deployment of Java CAPS.

What's in This Chapter

6.1 System Requirements Introduction

This chapter explains how to assess your needs for the following types of hardware:

- CPUs
- Random access memory (RAM)
- Hard disk space

There are many factors to consider in order to determine the hardware requirements for your particular system. This discussion is limited to issues as they relate directly to Java CAPS.

This chapter does not consider networking topology, and does not address such issues as shared applications, how resources are distributed throughout a network, and how many workstations are included in the network. For databases, this chapter assumes that each database management system is installed on a separate host.

6.2 Initial Considerations

Depending on the number of external connections, the type of data being processed, and how the data is processed, the required resources can vary. Consider the following points as you begin estimating your hardware requirements:

- Each deployment of Java CAPS is different. The configuration of each deployment is unique because there are varying numbers of components as well as variances in interconnectivity. Some components are bidirectional and complex, while others merely pass data through.
- In addition to differences in configuration, the computational requirements will differ both in scope and complexity. The high-performance architecture of Java CAPS is net-centric, which makes Java CAPS highly flexible. This flexibility makes predicting the general requirements of hardware a complex task.
- A Java CAPS solution is distributed using run-time components and is platform independent. System stability and redundancy are important considerations. Server requirements vary greatly, depending on the components on the server, the archiving requirements configured on the customer's system, and other factors.
- Instead of only referring to absolute minimum requirements, it is more meaningful to discuss the hardware needs of an installation in terms of recommendations. By using the methods described in this chapter, a system analyst can estimate the required resources for a given configuration, from a simple deployment to a complex deployment, and thereby define an initial recommendation. Once installed, Java CAPS can be fine-tuned, both in terms of hardware and software, to optimize performance.

6.3 Estimating Requirements

This section lists some of the factors that affect performance, and then provides guidelines for estimating requirements.

6.3.1 Factors

Because Java CAPS is a general-purpose toolkit that is flexible in its deployment and configuration, estimating the processor requirements is a challenge. There are infinite possibilities and numerous factors with complex interactions that affect the estimates. Some factors are more critical than others, depending on the circumstances.

Factors that affect performance include:

- CPU type and architecture
- CPU speed
- Presence of a CPU cache and its size
- Number of CPUs

- Physical memory size
- Swap size
- Disk subsystem (that is, bandwidth, latency, block size, RPM, seek time, and the presence and size of the cache)
- Network bandwidth and load
- Number of external systems and their latencies in servicing messages and acknowledgements
- Complexity and amount of processing to be performed by each component
- Message volume, size, and distribution through the day
- Throughput and response-time requirements
- Complexity of messages, including the number of nodes and complexity of regular expressions
- Bundling of messages (that is, more than one logical record in one physical record)
- Number of transitions between components for a given message (for example, moving data from an eWay to a topic/queue to an eWay or Collaboration)
- Amount of the implementation that can utilize parallel processing
- Other loads on the Logical Hosts (for example, queue cleanup schedules, backups, and other processes)
- Dispersion of solution across multiple CPUs and systems
- Number and architecture of subcomponents participating in the Project

6.3.2 Guidelines

Start with a good base configuration as a development system and use that configuration to predict the processor requirements for a production system for your unique implementation.

The recommended methodology is to:

1. Implement a representative number of interfaces (that exercise a good sample of the various data transformations and communication requirements of the implementation) on this system.
2. Run representative data files through the system.
3. Record the CPU load.

From these measures, you can project what the final production load will be and, therefore, the CPU requirement. The architecture can be tuned to achieve more efficiency using this same technique.

One advantage of the suite's distributed and scalable architecture is that hardware does not need to be replaced but can be included in a multi-system implementation. Therefore, as processing requirements grow, you can add new hardware.

General Considerations

When estimating resource requirements, certain general considerations exist.

Table 10 lists the RAM and disk space considerations for the Repository, Logical Host, and Enterprise Designer computers.

Table 10 General Considerations

Computer Role	RAM	Disk Space
Repository	Memory usage increases with each additional client connection. Every Enterprise Designer connection uses RAM resources on the Repository computer.	The disk space usage increases as additional Java CAPS products are installed.
Enterprise Designer	Memory usage increases based on the number of Projects or components being configured.	The disk space usage increases as additional Java CAPS products are installed.
Logical Host	Memory usage increases when: ▪ The number of Collaborations and other deployed modules increases. ▪ The size and complexity of messages increase.	Disk resources are used only by the products deployed on a Logical Host. A Logical Host running only one component uses less disk space than a Logical Host that contains multiple configured products.

Minimum System Requirements

The following tables list the minimum requirements for installing and running Java CAPS components. The RAM and disk space requirements do not take into consideration the RAM and disk space required by the operating system. Note that the requirements vary based on the operating system and the role of the host computer.

Note: *For optimum performance of Enterprise Designer, avoid running other applications at the same time.*

Table 11 Windows System Requirements

Component	CPU	RAM	Disk Space
Enterprise Designer	1.2 GHz Pentium class	768 MB	250 MB
Repository	1.2 GHz Pentium class	240 MB	1.2 GB
Enterprise Manager	1.2 GHz Pentium class	400 MB	170 MB
Logical Host	1.2 GHz Pentium class	270 MB	250 MB

Table 12 HP Tru64 System Requirements

Component	CPU	RAM	Disk Space
Repository	667 MHz	380 MB	1000 MB
Enterprise Manager	667 MHz	400 MB	150 MB
Logical Host	667 MHz	320 MB	350 MB

Table 13 HP-UX Itanium and PA-RISC System Requirements

Component	CPU	RAM	Disk Space
Repository	540 MHz	280 MB	1150 MB
Enterprise Manager	540 MHz	400 MB	400 MB
Logical Host	540 MHz	450 MB	500 MB

Table 14 IBM AIX System Requirements

Component	CPU	RAM	Disk Space
Repository	450 MHz	180 MB	900 MB
Enterprise Manager	450 MHz	400 MB	180 MB
Logical Host	450 MHz	300 MB	450 MB

Table 15 Red Hat Linux System Requirements

Component	CPU	RAM	Disk Space
Repository	1.2 GHz	240 MB	900 MB
Enterprise Manager	1.2 GHz	400 MB	180 MB
Logical Host	1.2 GHz	360 MB	350 MB

Table 16 Sun Solaris System Requirements

Component	CPU	RAM	Disk Space
Repository	400 MHz	240 MB	850 MB
Enterprise Manager	400 MHz	400 MB	210 MB
Logical Host	400 MHz	360 MB	400 MB

The *Java Composite Application Platform Suite Installation Guide* contains additional information about the system requirements. The Java CAPS Readme file contains the most up-to-date system requirements.

Additional Sun Solaris Requirements

If you want to run the Sun SeeBeyond Integration Server on Sun Solaris in 64-bit mode, then you must do the following:

- Use Sun Solaris 9 or 10.
- Add the **-d64** argument to the **JVM Args** property of the Sun SeeBeyond Integration Server. To access the **JVM Args** property, start Enterprise Designer, display the Environment Explorer, right-click the Integration Server, and click **Properties**.

Important: *You cannot run the Integration Server on Sun Solaris 8 in 64-bit mode.*

Chapter 7

Configuring Failover Support in a Sun Clustering Environment

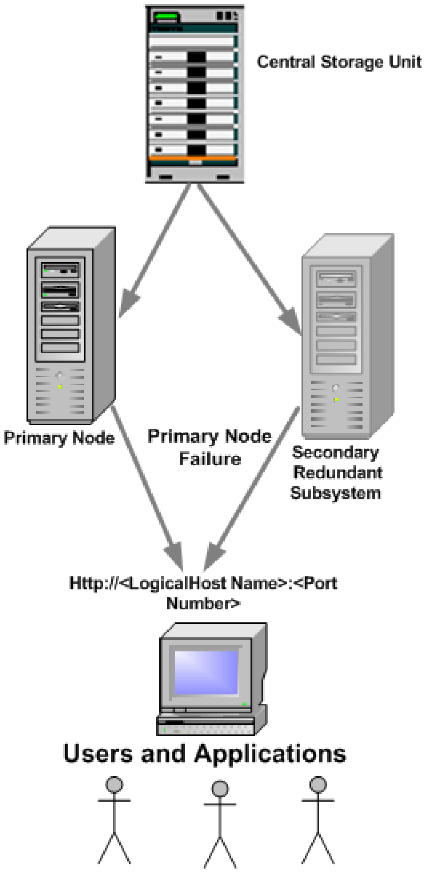
This chapter describes how to configure failover support for the Repository and the Logical Host using Sun™ Cluster 3.1.

What's in This Chapter

7.1 Sun™ Cluster 3.1 Overview

In a cluster, a set of nodes are tied together, such that the end users and applications perceive the nodes as a single server resource. Every node has a subsystem designed to incorporate redundancy. These redundant subsystems take charge when the primary system fails. This section explains the cluster environment by taking an example of two-node cluster. In general, it supports more than two nodes.

Figure 15 Cluster with Two Nodes



If the primary and secondary are active at the same time, then the configuration is termed as active/active. If the primary is active and the secondary is passive, then the configuration is termed as active/passive.

Resource types are public to the cluster. Resources are instances of resource types. Resource type enfolds an application, such that the application can run in a clustered environment. Resource Group Manager controls the clustered environment, and thus also the application running in it.

7.2 Configuring Support for Repository Failover

This section describes how to configure support for Repository failover in a Sun clustering environment.

If you are connecting remotely to the cluster nodes, install third-party software and configure it. Refer to the third-party user manual for instructions.

The utility commands are located in the **/usr/cluster/bin** directory.

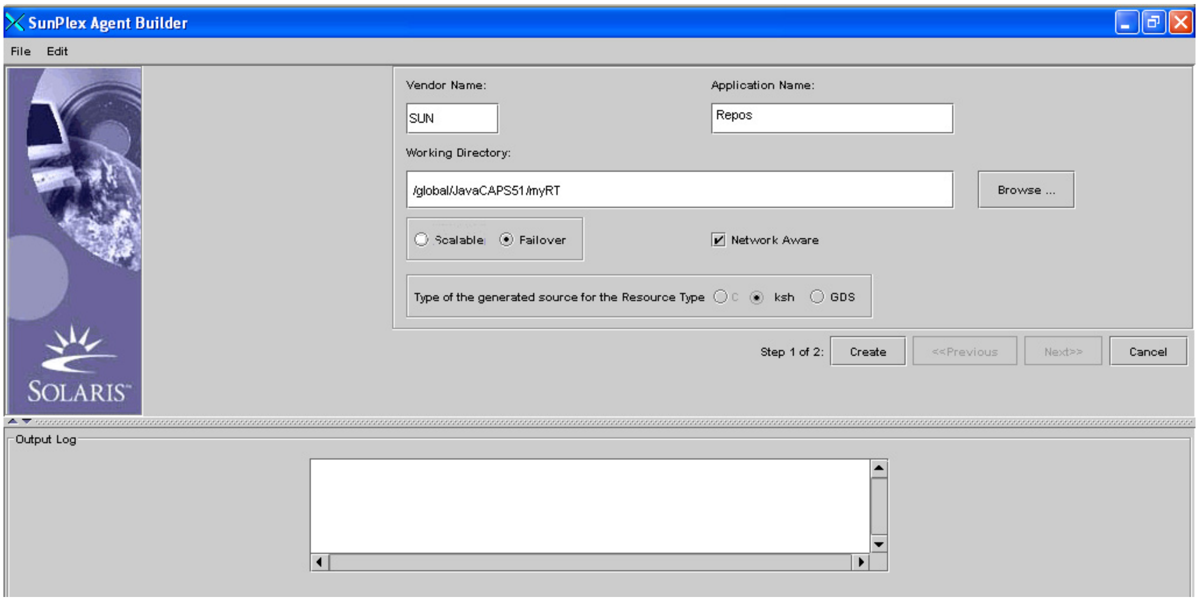
To configure support for Repository failover

1. From either of the nodes, install the Repository on the Central Storage Unit. Make sure that the installed location is available to all of the cluster nodes.
2. Type **scdsbuilder** to create a resource type for the Repository.

   Step 1 of SunPlex Agent Builder appears.

Note: *SunPlex Agent Builder belongs to Sun Cluster software, and is not a part of Java CAPS.*

Figure 16 SunPlex Agent Builder - Create Screen



Vendor Name: SUN

Application Name: Repos

Working Directory: /global/JavaCAPS51/myRT

○ Scalable ● Failover

☑ Network Aware

Type of the generated source for the Resource Type ○ C ● ksh ○ GDS

Step 1 of 2:

Table 17 SunPlex Agent Builder - Field Description

Field	Description	Values for this illustration
Vendor Name	Name of the vendor	**<SUN>** You can enter any name here.
Application Name	Name of the application	**<Repos>** You can enter any name here.

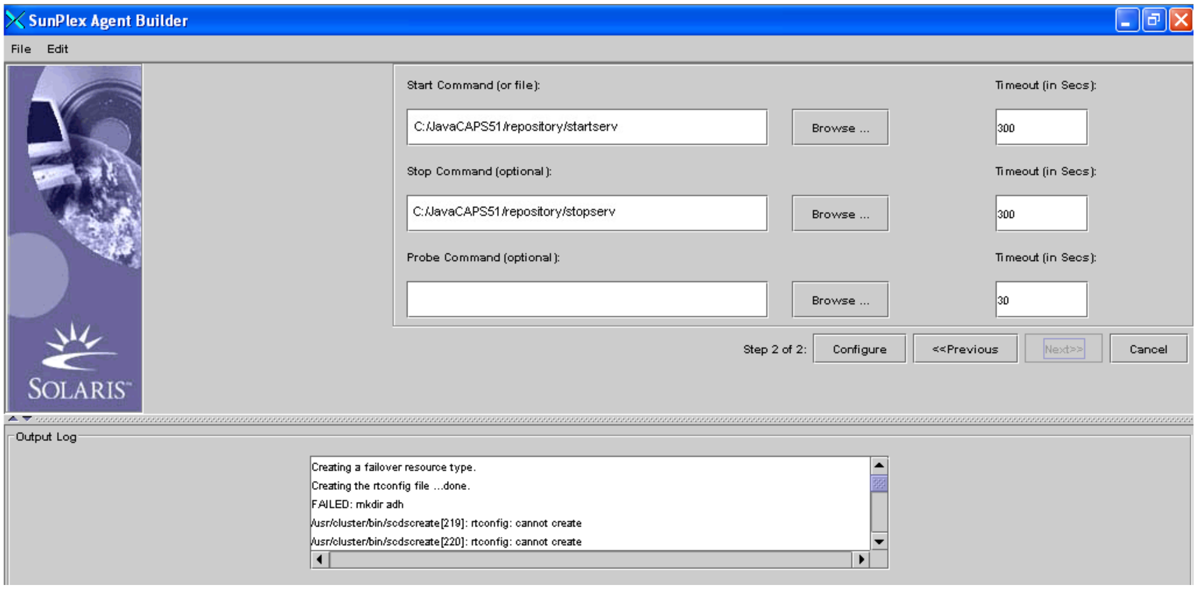
Table 17 SunPlex Agent Builder - Field Description

Field	Description	Values for this illustration
Working Directory	The location where the utility should store the resulting resource type.	**</global/JavaCAPS51/myRT>** You specify the working directory.
Scalable/Failover	Scalability provides high availability along with increased performance. Failover provides high availability.	Select **Failover**
Network Aware	If enabled, gives the provision for the application to query the network connectivity. It also gets notified in case of changes.	Enabled by default
Type of the generated source for the Resource Type	The different source options available are **C**, **ksh**, and **GDS**.	Select **ksh**.
Output Log	The log is generated when you click **Create**.	

3 Click **Create**.

Step 2 of SunPlex Agent Builder appears.

Figure 17 SunPlex Agent Builder - Configure Screen



Start Command (or file): C:/JavaCAPS51/repository/startserv

Timeout (in Secs): 300

Stop Command (optional): C:/JavaCAPS51/repository/stopserv

Timeout (in Secs): 300

Probe Command (optional):

Timeout (in Secs): 30

```
Creating a failover resource type.
Creating the rtconfig file ...done.
FAILED: mkdir adh
/usr/cluster/bin/scdscreate[219]: rtconfig: cannot create
/usr/cluster/bin/scdscreate[220]: rtconfig: cannot create
```

4 Click **Browse** to enter the full path of the start, stop, and probe scripts for the Repository. The probe command is optional. The paths of the scripts are:

Start Script

<Sun_JavaCAPS51_install_dir>/Repository/startserv.bat

For example, **C:\JavaCAPS51\repository\startserv.bat**

Stop Script

<Sun_JavaCAPS51_install_dir>/Repository/stopserv.bat

For example, **C:\JavaCAPS51\repository\stopserv.bat**

Probe Script

This field is optional and can be left blank.

Note: *The start, stop, and probe scripts are available in the Repository where the clustered application is created. Fault monitor constantly checks the data service to ensure that everything is running properly and the clients are served. Actions are taken based on the message sent by the probes.*

5 Click **Configure** to create the resource type as a Solaris package in the **<working directory>/<vendor+application name>/pkg** directory.

Note: *A new folder is created in the specified working directory. The name of the folder is the concatenated string of vendor and application name. The **pkg** folder is created under the newly created folder. This **pkg** folder location is the package location where the resource types are installed. For example, **/global/JavaCAPS51/myRT/SUNRepos/pkg**.*

6 Type **# pkgadd -d <Package Location>** to install the resource type on the nodes. Run this command separately in every node of the cluster.

7 Type **# scrgadm -a -g <Application Name> -<Resource Group Name> -h <list of hostnames of nodes separated by comma>** to create a resource group to hold the resources. For example:

# scrgadm -a -g Repos -RG -h host1, host2

Note: *The resources are grouped for better manageability. In case of failover, the resource groups are transferred as a whole. Scalable service employs scalable resource group and failover resource group. Failover service employs failover resource group and network resources.*

8 Type **# scrgadm -a -L -g <Resource Group Name> -l <Repository resource>** to add a resource of type Logical Host name to the Resource Group. For example:

# scrgadm -a -L -g Repos -RG -l sunJavaCAPS51cluster

Note: *You can access all nodes in the cluster by the LogicalHost name, which has a single IP address. Make sure that the system administrator has added an entry of the Logical Host name resource in the **/etc/hosts** directory with a valid IP address.*

9 Type **# scswitch -e -j <LogicalHost name resource>** and **# scswitch -e -M -j <LogicalHost name resource>** to enable the LogicalHost name resource. For example:

# scswitch -e -j sunJavaCAPS51cluster

# scswitch -e -M -j sunJavaCAPS51cluster

10 Type **# scrgadm -a -t <Vendor Name>.<Application Name>** to register the resource type. For example:

# scrgadm -a -t SUN.Repos

11 Type **# scrgadm -a -j <Application Name> -res -g <Application Name> -<Resource Group Name> -t <Vendor Name>.<Application Name> -y Scalable=False** to add a resource for Repository of the resource type. For example:

# scrgadm -a -j Repos -res -g Repos -RG -t SUN.Repos -y Scalable=False

12 Type **# scswitch -Z -g <Application Name> -<Resource Group>** to enable Repository resource. This command also brings it online. For example:

# scswitch -Z -g Repos -RG

13 Access the **http://<Logicalhostname>:<portnumber>** URL to view the application installed in the cluster environment**.** For example:

http://sunJavaCAPS51cluster:12000

7.3 Configuring Support for Logical Host Failover

This section describes how to configure support for Logical Host failover in a Sun clustering environment.

The utility commands are located in the **/usr/cluster/bin** directory.

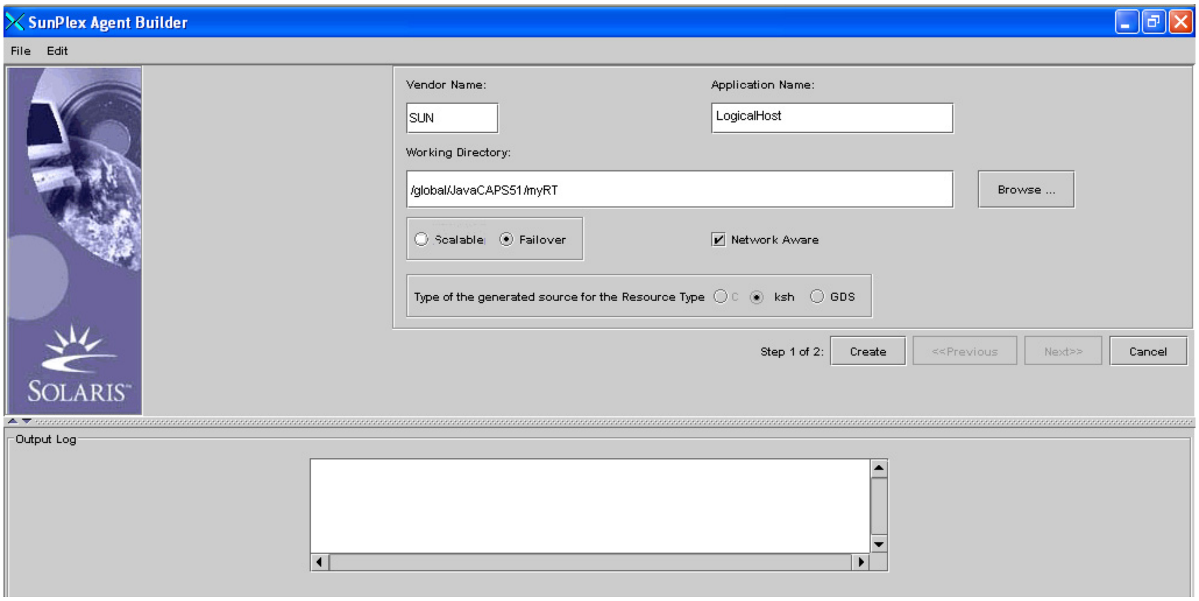
To configure support for Logical Host failover

1. From either of the nodes, install the Logical Host on the Central Storage Unit. Make sure that the installed location is available to all of the cluster nodes.
2. Type **scdsbuilder** to create a resource type for the Logical Host.

   Step 1 of SunPlex Agent Builder appears.

Note: *SunPlex Agent Builder belongs to the Sun Cluster software, and is not a part of Java CAPS.*

Figure 18 SunPlex Agent Builder - Create Screen



Vendor Name: SUN

Application Name: LogicalHost

Working Directory: /global/JavaCAPS51/myRT

○ Scalable ● Failover

☑ Network Aware

Type of the generated source for the Resource Type ○ C ● ksh ○ GDS

Table 18 SunPlex Agent Builder - Field Description

Field	Description	Values for this illustration
Vendor Name	Name of the vendor	**<SUN>** You can enter any name here.
Application Name	Name of the application	<**Logical Host**>. You can enter any name here.
Working Directory	The location where the utility should store the resulting resource type.	**</global/JavaCAPS51/myRT>** You specify the working directory.

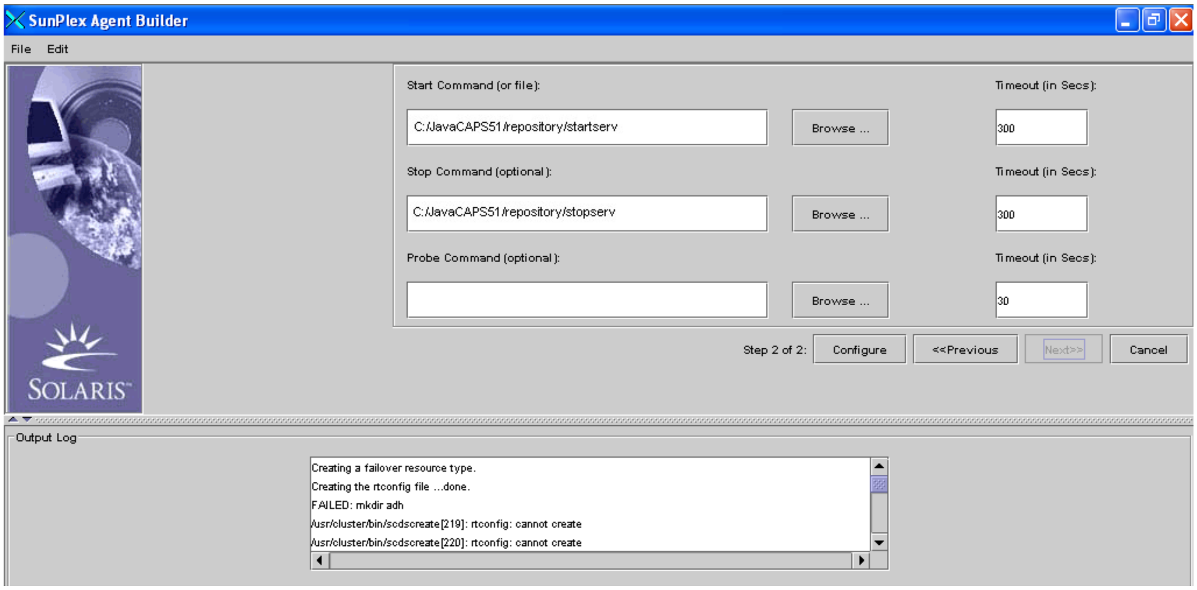
Table 18 SunPlex Agent Builder - Field Description

Field	Description	Values for this illustration
Scalable/Failover	Scalability provides high availability along with increased performance. Failover provides high availability.	Select **Failover**
Network Aware	If enabled, gives the provision for the application to query the network connectivity. It also gets notified in case of changes.	Enabled by default
Type of the generated source for the Resource Type	The different source options available are **C**, **ksh**, and **GDS**.	Select **ksh**.
Output Log	The log is generated when you click **Create**.	

3 Click **Create**.

Step 2 of SunPlex Agent Builder appears.

Figure 19 SunPlex Agent Builder - Configure Screen



Start Command (or file): C:/JavaCAPS51/repository/startserv

Timeout (in Secs): 300

Stop Command (optional): C:/JavaCAPS51/repository/stopserv

Timeout (in Secs): 300

Probe Command (optional):

Timeout (in Secs): 30

Step 2 of 2:

Output Log

```
Creating a failover resource type.
Creating the rtconfig file ...done.
FAILED: mkdir adh
/usr/cluster/bin/scdscreate[219]: rtconfig: cannot create
/usr/cluster/bin/scdscreate[220]: rtconfig: cannot create
```

4 Click **Browse** to enter the full path of the start, stop, and probe scripts for the Logical Host. The probe command is optional. The paths of the scripts are:

Start Script

<Sun_JavaCAPS51_install_dir>/logicalhost/is/domains/<domain name>/bin/startserv.bat

Stop Script

<Sun_JavaCAPS51_install_dir>/logicalhost/is/domains/<domain name>/bin/stopserv.bat

Probe Script

<Sun_JavaCAPS51_install_dir>/logicalhost/util/scripts/pingis.bat

Note: *The start, stop, and probe scripts are available in the domain where the clustered application is created. Fault monitor constantly checks the data service to ensure that everything is running properly and the clients are served. Actions are taken based on the message sent by the probes.*

5 Click **Configure** to create the resource type as a Solaris package in the **<working directory>/<vendor+application name>/pkg** directory.

Note: *A new folder is created in the specified working directory. The name of the folder is the concatenated string of vendor and application name. The* ***pkg*** *folder is created under the newly created folder. This* ***pkg*** *folder location is the package location where the resource types are installed. For example,* ***/global/JavaCAPS51/myRT/SUNLogicalHost/pkg****.*

6 Type **# pkgadd -d <Package Location>** to install the resource type on the nodes. Run this command separately in every node of the cluster.

7 Type **# scrgadm -a -g <Application Name> -<Resource Group Name> -h <list of hostnamesof nodes separated by comma>** to create a resource group to hold the resources. For example:

# scrgadm -a -g LogicalHost -RG -h host1, host2

Note: *The resources are grouped for better manageability. In case of failover, the resource groups are transferred as a whole. Scalable service employs scalable resource group and failover resource group. Failover service employs failover resource group and network resources.*

8 Type **# scrgadm -a -L -g <Application Name> -<Resource Group Name> -l <LogicalHost resource>** to add a resource of type LogicalHostname to the Resource Group. For example:

# scrgadm -a -L -g LogicalHost -RG -l sunJavaCAPS51cluster

Note: *You can access all nodes in the cluster by the Logical Host name, which has a single IP address. Make sure that the system administrator has added an entry of the logicalhost resource in the* ***/etc/hosts*** *directory with a valid IP address.*

9 Type **# scswitch -e -j <LogicalHost resource>** and **# scswitch -e -M -j <LogicalHost resource>** to enable the LogicalHostname resource. For example:

# scswitch -e -j sunJavaCAPS51cluster

# scswitch -e -M -j sunJavaCAPS51cluster

10 Type **# scrgadm -a -t <Vendor Name>.<Application Name>** to register the resource type. For example:

# scrgadm -a -t SUN.LogicalHost

11 Type **# scrgadm -a -j <Application Name> -res -g <Application Name> -<Resource Group Name> -t <Vendor Name>.<Application Name> -y Scalable=False** to add a resource for LogicalHost of the resource type. For example:

# scrgadm -a -j LogicalHost -res -g LogicalHost -RG -t SUN.LogicalHost -y Scalable=False

12 Type **# scswitch -Z -g <Application Name> -<Resource Group>** to enable LogicalHost resource. This command also brings it online. For example:

# scswitch -Z -g LogicalHost -RG

13 Access the **http://<Logicalhostname>:<portnumber>** url to view the application installed in the cluster environment. For example:

http://sunJavaCAPS51cluster:18000

Chapter 8

Configuring Failover Support in a Windows Clustering Environment

This chapter describes how to configure failover support for the Repository and the Logical Host using Windows Server 2003 clustering technologies.

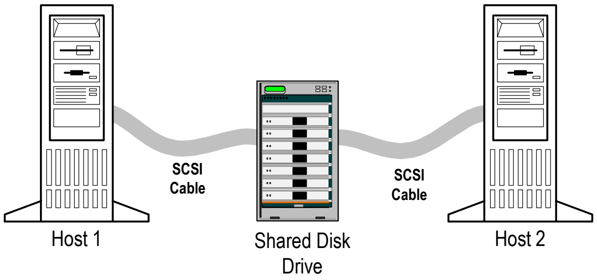
What's in This Chapter

- **"Windows Clustering Overview" on page 77**
- **"Configuring Support for Repository Failover" on page 78**
- **"Configuring Support for Logical Host Failover" on page 88**

8.1 Windows Clustering Overview

A *cluster* consists of two or more independent computers that work together as one system. These computers are referred to as *nodes*. Each node includes an instance of the Cluster service. Each node is connected to a shared storage device. Figure 20 shows a cluster with two nodes.

Figure 20 Cluster With Two Nodes



A group is a collection of resources that are assigned to a node. Examples of resources include physical disks, IP addresses, and network names.

A node can be either active or passive. You can set up a cluster in which one node is active and the other node(s) is passive. This configuration is called *active/passive***.** You can also set up a cluster in which all the nodes are active. This configuration is called *active/active.*

8.2 Configuring Support for Repository Failover

This section describes how to configure support for Repository failover in a Windows clustering environment.

8.2.1 Overview

In the Repository failover environment, one node is active and the other node(s) is passive. This configuration is called *active/passive*.

During the configuration process, you use Microsoft's Cluster Administrator tool to create a group that contains the following resources:

- Shared disk drive
- IP address
- Network name
- The Repository

In addition, you must install the Repository on the shared disk drive.

The active node is the owner of the group. If one of the resources fails, the group is automatically moved to another node and started on that node. This process is called *failover*. The new node now becomes the active node.

The Repository is not cluster aware. When a failover is completed, the Repository does not remember the state that existed before the failover. This limitation could affect the users. For example,

- If an Enterprise Designer user is saving objects to the Repository during the failover, then the save will not succeed. Once the failover is completed, the user will need to save the objects again. If an Enterprise Designer user is not saving objects during the failover, then the failover will be transparent to the user.
- When the failover begins, the browser session of all Enterprise Manager users will expire. Once the failover is completed, the users will need to login to Enterprise Manager again.

8.2.2 Requirements

Before you perform the procedures, you must have the following:

- A functional active/passive cluster that consists of two or more nodes. The nodes must be running on Windows Server 2003.

  For detailed information about how to set up a cluster, see the Microsoft clustering documentation.

Note: *Microsoft has a certification program for hardware vendors that supply a "cluster system." When you go to a vendor to buy computers for a cluster, the vendor has to sell you two computers, set up in the specific way dictated by Microsoft. See the appropriate Microsoft documentation or Web page for details.*

- A cluster shared drive that is available for the Repository installation. The shared drive must be mountable from each cluster node.
- CAPS 5.1 was tested using Windows Server 2003 with Cluster Administrator version 5.2.

8.2.3 Creating the Cluster Group

The first task is to create a cluster group in which you can include a set of resources.

Note: *If you are using an existing group, you can skip this procedure.*

To create the Cluster Group

1. Start Cluster Administrator.
2. Select the cluster in which the Repository will run.
3. Right-click **Groups** folder under the cluster, select **New**, and then choose **Group**.

   The **New Group** wizard appears.
4. Enter a name for the group (for example, **CAPS Cluster Group**). Enter a description for the group. Click **Next**.

   The **Preferred Owners** dialog box appears.
5. Move the nodes to the **Preferred Owners** list and arrange them in order of preference. Click **Finish**.

8.2.4 Creating a Physical Disk Resource

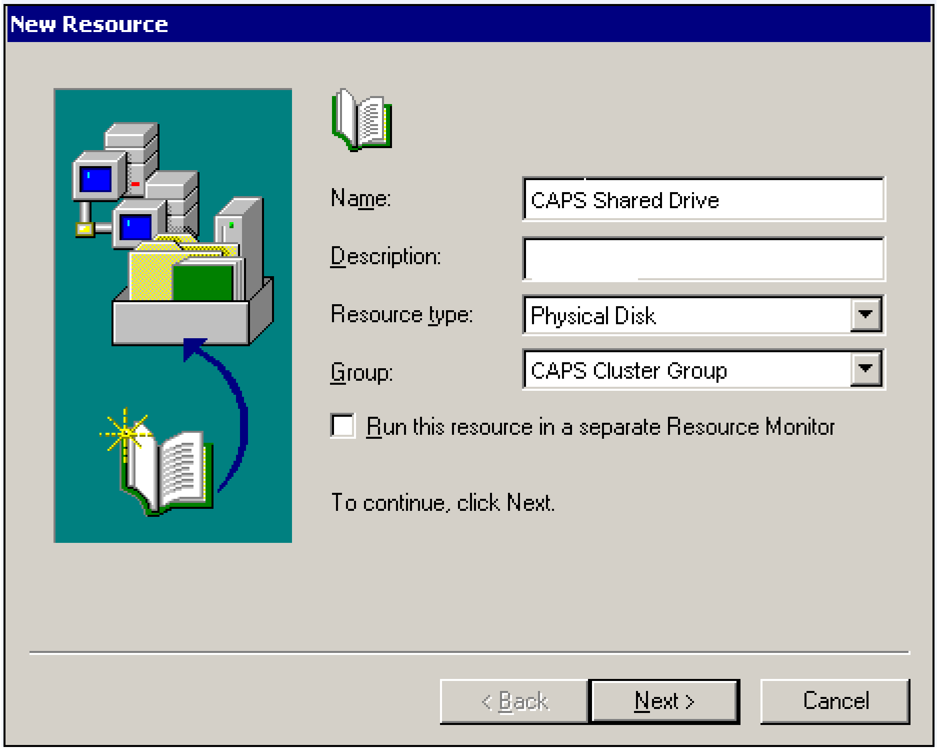
You must now create a Physical Disk resource within the CAPS Cluster Group. From the Cluster Administrator, perform the following steps.

To create a Physical Disk resource

1. Right-click **CAPS Cluster Group**, select **New**, and then choose **Resource**.

   The **New Resource** dialog box appears.

Figure 21 New Resource Dialog Box



New Resource

Name: CAPS Shared Drive

Description:

Resource type: Physical Disk

Group: CAPS Cluster Group

□ Run this resource in a separate Resource Monitor

To continue, click Next.

2 Enter a name for the resource (for example, **CAPS Shared Drive**).

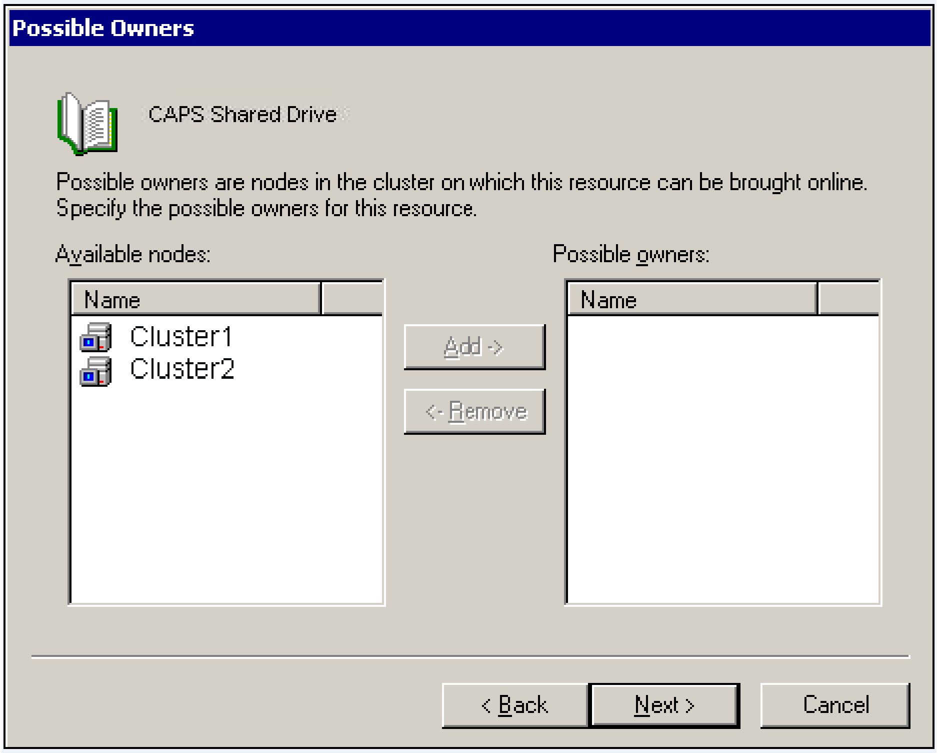
3 Enter a description for the resource.

4 Set the resource type to **Physical Disk**.

5 Set the group to the **CAPS Cluster Group**. When finished, click **Next**.

The **Possible Owners** dialog box appears.

Figure 22 Possible Owners Dialog Box



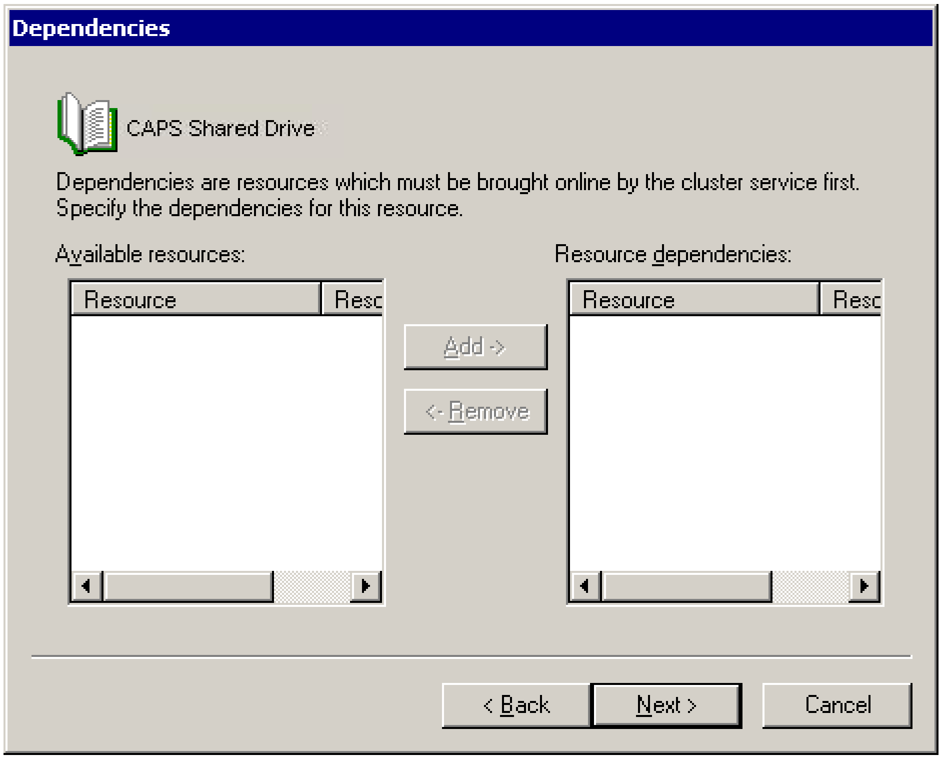
CAPS Shared Drive

Possible owners are nodes in the cluster on which this resource can be brought online. Specify the possible owners for this resource.

6 Move the nodes that will be part of the cluster to the **Possible Owners** list. Click **Next**.

The **Dependencies** dialog box appears.

Figure 23 Dependencies Dialog Box



CAPS Shared Drive

Dependencies are resources which must be brought online by the cluster service first. Specify the dependencies for this resource.

7 The Physical Disk resource is not dependent on other resources. Therefore, click **Next**.

The **Disk Parameters** dialog box then appears.

8 Choose the shared disk. Click **Finish**.

9 To ensure that the Physical Disk resource is accessible from each node, move the **CAPS Cluster Group** into each node's Active Groups folder, and verify that the group can be brought online at each node.

8.2.5 Creating an IP Address Resource

You must now create an IP Address resource within the **CAPS Cluster Group**. From the Cluster Administrator, perform the following steps.

To create an IP Address resource

1 Right-click **CAPS Cluster Group**, select **New**, and then choose **Resource**.

The **New Resource** dialog box appears.

2 Enter a name for the resource (for example, **CAPS IP Address**).

3 Enter a description for the resource.

4 Set the resource type to **IP Address**.

5 Set the group to the **CAPS Cluster Group**. When finished, click **Next**.

The **Possible Owners** dialog box appears.

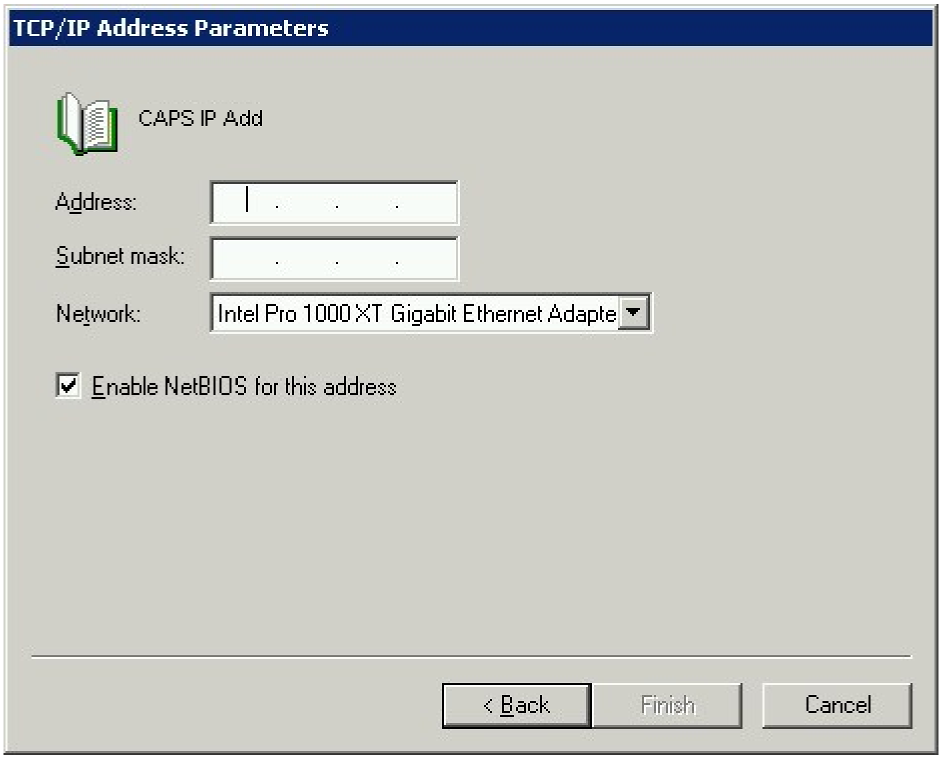
6 Move the nodes that will be part of the cluster to the **Possible Owners** list. Click **Next**.

The **Dependencies** dialog box appears.

7 The IP Address resource is not dependent on other resources. Therefore, click **Next**.

The **TCP/IP Address Parameters** dialog box appears.

Figure 24 TCP/IP Address Parameters Dialog Box



TCP/IP Address Parameters

CAPS IP Add

Address: . . .

Subnet mask: . . .

Network: Intel Pro 1000 XT Gigabit Ethernet Adapte

☑ Enable NetBIOS for this address

8 Enter the **IP address** and **subnet mask** of the cluster. Choose the network card. Click **Finish**.

9 To ensure that the IP Address resource is accessible from each node, move the **CAPS Cluster Group** into each node's Active Groups folder and verify that the group can be brought online at each node.

8.2.6 Creating a Network Name Resource

You must now create a Network Name resource within the **CAPS Cluster Group**. From the Cluster Administrator, perform the following steps.

To create a Network Name resource

1 Right-click the **CAPS Cluster Group**, select **New**, and then choose **Resource**.

The **New Resource** dialog box appears.

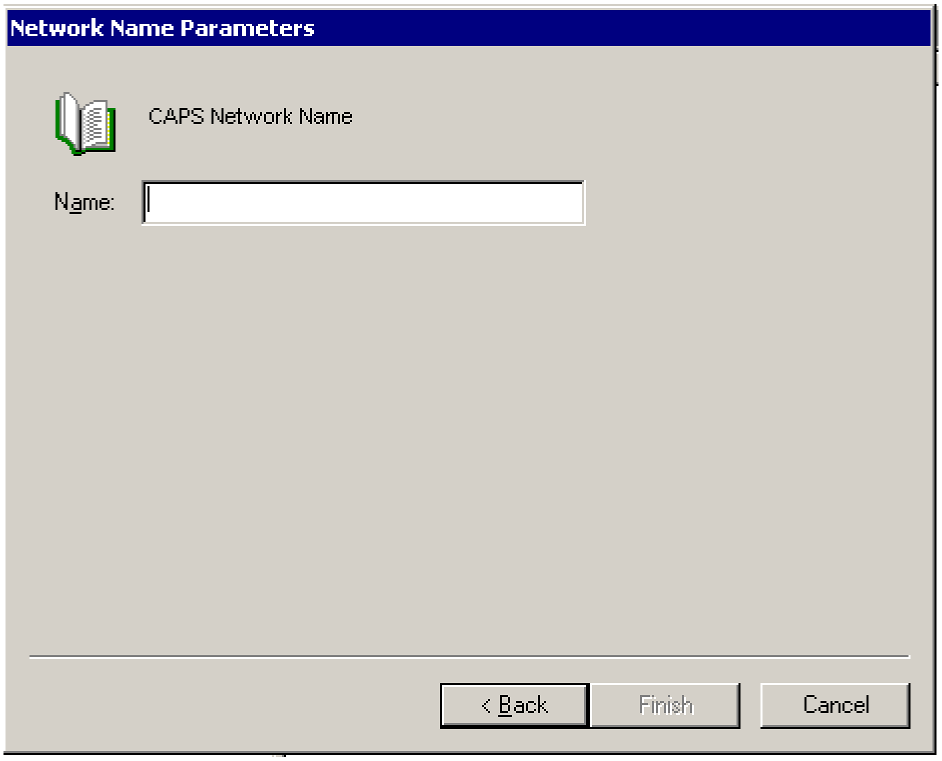
2. Enter a name for the resource (for example, **CAPS Network Name**).
3. Enter a description for the resource.
4. Set the resource type to **Network Name**.
5. Set the group to the **CAPS Cluster Group**. When finished, click **Next**.

   The **Possible Owners** dialog box appears.
6. Move the nodes that will be part of the cluster to the **Possible Owners** list. Click **Next**.

   The **Dependencies** dialog box appears.
7. The Network Name resource is dependent on the IP Address resource. Therefore, move the IP Address resource to the **Resource dependencies** list. When finished, click **Next**.

   The **Network Name Parameters** dialog box appears.

Figure 25 Network Name Parameters Dialog Box



8. Enter the network name of the cluster. Click **Finish**.
9. To ensure that the Network Name resource is accessible from each node, move the **CAPS Cluster Group** into each node’s Active Groups folder and verify that the group can be brought online at each node.

8.2.7 Installing the Repository as a Windows Service

Install the Repository on one of the cluster nodes where the shared disk drive is accessible. During the installation process, you must do the following:

- Use a shared drive letter in the Repository location (for example, **X:\CAPS...**).
- Specify that you want to run the Repository as a Windows service. If you are using the GUI installation program, do this by selecting the **Run repository as Windows Service** check box in the **Repository Configuration** dialog box.

Note: *For detailed instructions on installing the Repository, see the Sun Java Composite Application Platform Suite Installation Guide.*

8.2.8 Duplicating the Registry Keys

When you installed the Repository as a Windows service in the previous subsection, a series of registry keys were created on that cluster node. You must copy these registry keys to the other cluster nodes.

Important: *The drive letter must be the same on each cluster node.*

The steps to be followed are:

1. From the Cluster Administrator, move the **CAPS Cluster Group** to a node on which the Repository was *not* installed.
2. Run the **installwinsvc.bat** script in the **<shared_drive>:\<CAPS_home>\repository** directory on the node.
3. Repeat Step 1 and Step 2 for the other nodes (if any) on which the Repository was *not* installed.

8.2.9 Creating a Generic Service Resource

You must now create a Generic Service resource within the **CAPS Cluster Group.** This resource represents the Repository. From the Cluster Administrator, perform the following steps.

To create a Generic Service resource

1. Right-click the **CAPS Cluster Group**, select **New**, and then choose **Resource**.

   The **New Resource** dialog box appears.
2. Enter a name for the resource (for example, **CAPS Repository**).
3. Enter a description for the resource.
4. Set the resource type to **Generic Service**.
5. Set the group to the **CAPS Cluster Group**. When finished, click **Next**.

   The **Possible Owners** dialog box appears.

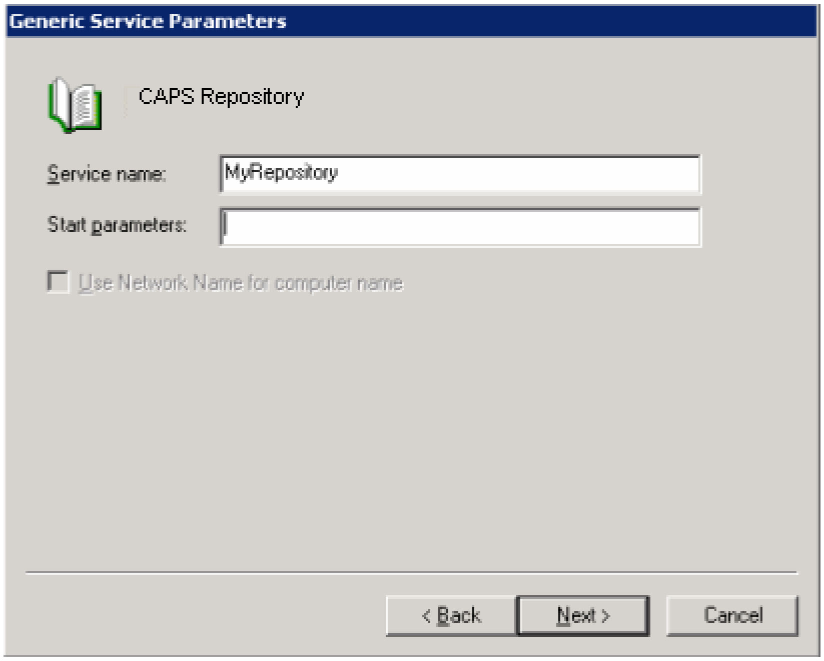
6 Move the nodes that will be part of the cluster to the **Possible Owners** list. Click **Next**.

The **Dependencies** dialog box appears.

7 The Generic Service resource is dependent on the Physical Disk, IP Address, and Network Name resources. Therefore, move these resources to the **Resource dependencies** list. When finished, click **Next**.

The **Generic Service Parameters** dialog box appears.

Figure 26 Generic Service Parameters Dialog Box



Service name: MyRepository

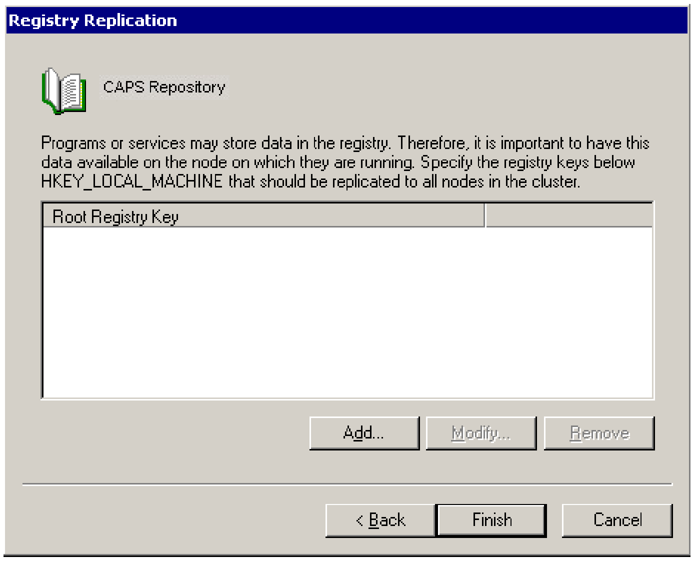
Start parameters:

□ Use Network Name for computer name

8 In the **Service name** field, enter the Repository name. Leave the **Start parameters** field blank. Click **Next**.

The **Registry Replication** dialog box appears.

Figure 27 Registry Replication Dialog Box



Registry Replication

CAPS Repository

Programs or services may store data in the registry. Therefore, it is important to have this data available on the node on which they are running. Specify the registry keys below HKEY_LOCAL_MACHINE that should be replicated to all nodes in the cluster.

Root Registry Key	

9 Click **Finish**.

10 To ensure that the Generic Service resource is accessible from each node, move the **CAPS Cluster Group** into each node's Active Groups folder and verify that the group can be brought online at each node.

8.2.10 Using the Repository in a Windows Clustering Environment

In a Windows clustering environment, you **start** and **stop** the Repository from Cluster Administrator. Right-click the Generic Service resource (which represents the Repository) and choose **Bring Online** or **Take Offline**, respectively.

Note: *The context menu also contains an item called* **Initiate Failure**. *You can choose this item for testing purposes to simulate a failover.*

When the Repository is running, you can connect to it from Enterprise Designer and Enterprise Manager. In the hostname URL, use the *cluster name* rather than a node name. You will automatically be connected to the active node.

For example, assume that:

- You have a two-node cluster.
- The cluster name is **MyWindows2003Cluster.acme.com**.
- The name of node 1 is **MyWindows2003Node1.acme.com**.
- The name of node 2 is **MyWindows2003Node2.acme.com**.
- The base port number of the Repository is **12000**.

To access Enterprise Designer or Enterprise Manager, you would use

```
http://MyWindows2003Cluster.acme.com:12000
```

8.3 Configuring Support for Logical Host Failover

This section describes how to configure support for Logical Host failover in a Windows clustering environment.

8.3.1 Overview

In the Logical Host failover environment, one node is **active** and the other node(s) is **passive**. This configuration is called *active/passive*.

During the configuration process, you use Microsoft's Cluster Administrator tool to create a group that contains the following resources:

- Shared disk drive
- IP address
- Network name
- The Logical Host itself

In addition, you must install the Logical Host on the shared disk drive.

The active node is the owner of the group. If one of the resources fails, the group is automatically moved to another node and started on that node. This process is called ***failover***. The new node becomes the active node.

8.3.2 Requirements

Before you perform the following procedures, you must have the following:

- A functional active/passive cluster that consists of two or more nodes. The nodes must be running on **Windows Server 2003**.

  For detailed information about how to set up a cluster, see the *Microsoft clustering documentation.*

Note: *Microsoft has a certification program for hardware vendors that supply a "cluster system." When you go to a vendor to buy computers for a cluster, the vendor has to sell you two computers, set up in the specific way dictated by Microsoft. See the appropriate Microsoft documentation or Web page for details.*

- A cluster shared drive that is available for the Logical Host installation. The shared drive must be mountable from each cluster node.
- CAPS 5.1 was tested using Windows Server 2003 with Cluster Administrator version 5.2..

8.3.3 Creating the CAPS Cluster Group

The first task is to create a cluster group in which you can include a set of resources.

Note: *If you are using an existing group, you can skip this procedure.*

To create the CAPS Cluster Group

1. Start Cluster Administrator.
2. Select the cluster in which the Logical Host runs.
3. Right-click the **Groups** folder under the cluster, select **New**, and then choose **Group**.

   The **New Group** wizard appears.
4. Enter a name for the group (for example, **CAPS LH Group**).
5. Enter a description for the group. Click **Next**.

   The **Preferred Owners** dialog box appears.
6. Move the nodes to the **Preferred Owners** list and arrange them in order of preference. Click **Finish**.

8.3.4 Creating a Physical Disk Resource

You must now create a Physical Disk resource within the **CAPS Cluster Group**. From the Cluster Administrator, perform the following steps.

To create a Physical Disk resource,

1. Right-click the **CAPS Cluster Group**, select **New**, and then choose **Resource**.

   The **New Resource** dialog box appears.
2. Enter a name for the resource (for example, **CAPS LH Shared Drive**).
3. Enter a description for the resource.
4. Set the resource type to **Physical Disk**.
5. Set the group to the **CAPS Cluster Group**. When finished, click **Next**.

   The **Possible Owners** dialog box appears.
6. Move the nodes that will be part of the cluster to the **Possible Owners** list. Click **Next**.

   The **Dependencies** dialog box appears.
7. The Physical Disk resource is not dependent on other resources. Therefore, click **Next**.

   The **Disk Parameters** dialog box appears.
8. Choose the shared disk. Click **Finish**.
9. To ensure that the Physical Disk resource is accessible from each node, move the **CAPS Cluster Group** into each node’s Active Groups folder and verify that the group can be brought online at each node.

8.3.5 Creating an IP Address Resource

You must now create an IP Address resource within the **CAPS Cluster Group**. From the Cluster Administrator, perform the following steps.

To create an IP Address resource

1 Right-click **CAPS Cluster Group**, select **New**, and then choose **Resource**.

The **New Resource** dialog box appears.

2 Enter a name for the resource (for example, **CAPS LH IP Address**). Enter a description for the resource. Set the resource type to **IP Address**. Set the group to the **CAPS Cluster Group**. When finished, click **Next**.

The **Possible Owners** dialog box appears.

3 Move the nodes that will be part of the cluster to the **Possible Owners** list. Click **Next**.

The **Dependencies** dialog box appears.

4 The IP Address resource is not dependent on other resources. Therefore, click **Next**.

The **TCP/IP Address Parameters** dialog box appears.

5 Enter the IP address and subnet mask of the cluster. Choose the network card. Click **Finish**.

6 To ensure that the IP Address resource is accessible from each node, move the **CAPS Cluster Group** into each node’s Active Groups folder and verify that the group can be brought online at each node.

8.3.6 Creating a Network Name Resource

You must now create a Network Name resource within the **CAPS Cluster Group**. From the Cluster Administrator, perform the following steps.

To create a Network Name resource

1 Right-click the **CAPS Cluster Group**, select **New**, and then choose **Resource**.

The **New Resource** dialog box appears.

2 Enter a name for the resource (for example, **CAPS LH Network Name**). Enter a description for the resource. Set the resource type to **Network Name**. Set the group to the **CAPS Cluster Group**. When finished, click **Next**.

The **Possible Owners** dialog box appears.

3 Move the nodes that will be part of the cluster to the **Possible Owners** list. Click **Next**.

The **Dependencies** dialog box appears.

4 The Network Name resource is dependent on the IP Address resource. Therefore, move the IP Address resource to the **Resource dependencies** list. When finished, click **Next**.

The **Network Name Parameters** dialog box appears.

5 Enter the network name of the cluster. Click **Finish**.

6. To ensure that the Network Name resource is accessible from each node, move the **CAPS Cluster Group** into each node's Active Groups folder and verify that the group can be brought online at each node.

8.3.7 Installing the Logical Host as a Windows Service

Install the Logical Host on one of the cluster nodes where the shared disk drive is accessible. During the installation process, you must do the following:

- Use a shared drive letter for the Logical Host location (for example, **X:\CAPS..**)

Creating Domains

To deploy applications to the Sun SeeBeyond Integration Server, you must first create a domain.

A *domain* is an instance of a Logical Host. You create a domain using the Domain Manager.

Using the Domain Manager

The Domain Manager is included with the Windows installation of the Logical Host.

To create a domain using the Domain Manager

1. In the **<CAPS-root>\logicalhost** directory, run the **domainmgr.bat** script.
2. If there are currently no domains, a dialog box indicates that you can create a domain now. If you click **Yes,** the **Create Domain** dialog box appears displaying information about existing domains (if any). To allow the Domain Manager choose the port numbers for you, click **AutoPick Port.**
3. On the **Action** menu, click **New Domain.** The **Create Domain** dialog box next appears. If desired, change the default values

If you want to install the Sun SeeBeyond Integration Server as a Windows service, select the **Install Runtime** as **Windows Service** check box. The Service name appears as **Domain1** by default.

Note: *For detailed instructions on installing the Logical Host, see the Sun Java Composite Application Platform Suite Installation Guide and for detailed instructions on configuring the Logical Host to run as a Windows service, see the Sun SeeBeyond eGate Integrator System Administration Guide.*

8.3.8 Duplicating the Registry Keys

When you installed the Logical Host as a Windows service in the previous subsection, a series of registry keys were created on that cluster node. You must copy these registry keys to the other cluster nodes.

The steps for performing this are:

1. From the Cluster Administrator, move the **CAPS Cluster Group** to a node on which the Logical Host was *not* installed.
2. Run **<shared_drive>:\logicalhost\is\bin\service_domain1.bat** on the node.
3. Repeat Step 1 and Step 2 for the other nodes (if any) on which the Logical Host was *not* installed.

8.3.9 Creating a Generic Service Resource

You must now create a Generic Service resource within the **CAPS Cluster Group**. This resource represents the Logical Host. From the Cluster Administrator, perform the following steps.

To create a Generic Service resource

1. Right-click the **CAPS Cluster Group**, select **New**, and then choose **Resource**.

   The **New Resource** dialog box appears.
2. Enter a name for the resource (for example, **CAPS Logical Host**).
3. Enter a description for the resource.
4. Set the resource type to **Generic Service**.
5. Set the group to the **CAPS Cluster Group**. When finished, click **Next**.

   The **Possible Owners** dialog box appears.
6. Move the nodes that will be a part of the cluster to the **Possible Owners** list. Click **Next**.

   The **Dependencies** dialog box appears.
7. The Generic Service resource is dependent on the Physical Disk, IP Address, and Network Name resources. Therefore, move these resources to the **Resource dependencies** list. When finished, click **Next**.

   The **Generic Service Parameters** dialog box appears.
8. In the **Service name** field, enter the name of the Logical Host Windows service. Leave the **Start parameters** field blank. Click **Next**.

   The **Registry Replication** dialog box appears.
9. Click **Finish**.
10. To ensure that the Generic Service resource is accessible from each node, move the **CAPS Cluster Group** into each node’s Active Groups folder and verify that the group can be brought online at each node.

8.3.10 Using the Logical Host in a Windows Clustering Environment

In a Windows clustering environment, you **start** and **stop** the Logical Host from Cluster Administrator. Right-click the Generic Service resource (which represents the Logical Host) and choose **Bring Online** or **Take Offline**, respectively.

Note: *The context menu also contains an item called* **Initiate Failure***. You can choose this item for testing purposes to simulate a failover.*

Chapter 9

eInsight Load Balancing and Failover

This chapter describes how to configure Sun SeeBeyond eInsight Business Process Manager for load balancing and failover.

What's in This Chapter

9.1 Configuring Load Balancing

When a Business Process needs to be scaled to meet heavier processing needs, you can distribute the Business Process across multiple engines to increase throughput. eInsight's load balancing algorithm automatically distributes processing across multiple engines; however, eInsight cannot load balance correlated messages.

To configure load balancing

1. Ensure that eInsight Persistence is enabled.
2. In the eInsight Engine Properties, set **eInsight Persistence** to **Multiple Engines**.
3. Configure all eInsight Engines to share the same database.

9.2 Configuring Failover

When your Business Process is configured for load balancing, eInsight's failover capabilities ensure throughput of running Business Process instances. When Business Process instances encounter an engine failure, eInsight load balances those instances across all available engines. As with load balancing, eInsight's failover capabilities are limited to non-correlated messages.

To configure failover

1. In the eInsight Engine Properties, set **Engine Expiry Interval (sec)** so that it registers itself as alive frequently enough to meet the demands of your system. Optimizing this property setting might require some testing. This property also applies to the interval for the recovery of dangling instances. The default setting is 120.

2 In the eInsight Engine Properties, set **Failover Grace Period (sec)** for the optimal elapsed time period before moving running Business Process instances from an unavailable engine to an available engine. Optimizing this property setting might require some testing.

Index