IBM

Intentia Movex Java on the IBM *e*server iSeries Server

An Implementation Guide

Overview of Movex Java on the iSeries server

Movex Java on iSeries installation and configuration

Operational tips and techniques

Yessong Johng
Per Danielsson
Per Ehnsiö
Mats Hermansson
Mika Jolanki
Scott Moore
Lars Strander
Lars Wettergren

Redbooks

ibm.com/redbooks

IBM

International Technical Support Organization

Intentia Movex Java on the IBM @server **iSeries Server: An Implementation Guide**

November 2002

SG24-6545-00

Take Note! Before using this information and the product it supports, be sure to read the general information in "Notices" on page ix.

First Edition (November 2002)

This edition applies to OS/400 V5R1 (product number 5722-SS1) and to Intentia Movex Java Version 12.4.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

ix

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

IBM eServer™	iSeries™	Redbooks™
Redbooks (logo)™	Operating System/400®	SP™
AS/400®	OS/2®	SP1®
DB2®	OS/400®	WebSphere®
IBM®	Perform™	xSeries™
IPDS™	PowerPC®	

The following terms are trademarks of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both:

Lotus®	Lotus Notes®
Word Pro®	Notes®

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Preface

The Intentia Movex Java solution improves Intentia customers' business processes. It offers of an optimum set of knowledge, tools, methods, and functionality for a successful configuration and implementation of Movex. This IBM Redbook provides a detailed guide that explains the specific tasks associated with implementing Movex Java on the IBM @server iSeries server. It is based on a collection of knowledge gathered by the architects and developers behind Movex Java, and by the Intentia professionals who have implemented Movex Java at customer sites.

This redbook is designed to assist Movex Java customers, Movex Java consultants, business partners, and IBM technical and service representatives. It targets these professionals who are directly involved with implementing a total business solution consisting of the Movex Java solution, the iSeries, the DB2 for iSeries database, and supplemental solution products.

This redbook explores the following topics:

► An introduction to Intentia and Movex Java
► An overview of Movex Java
► Application architecture
► Database architecture
► Java overview and iSeries implementation
► Installation
► Work management
► Printing
► Security
► Backup and recovery
► Movex Java Utilities
► System sizing

The material in this redbook applies to the current version of Movex Java and is subject to change.

You can find the most current Movex Java information on the Intentia Wire. This is an Intentia intranet site that is available only to Intentia representatives and requires a user ID and password. If you do not have access to the Intentia Wire, contact your local Intentia office. You can locate the Intentia Wire at:

http://www.intentia.com/Intentia/wire2000.nsf/

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Rochester Center.

Yessong Johng is a Senior iSeries Specialist at the International Technical Support Organization, Rochester Center.

Per Danielsson is a Technical Domain Expert at Intentia R&D, Sweden.

Per Ehnsiö is a Chief Technology Officer at Intentia R&D, Sweden.

Mats Hermansson is a Documentation Team Manager at Intentia R&D, Sweden.

Mika Jolanki is an IT Specialist at IBM Sweden and a Project Manager for Movex Java on iSeries technical optimization.

Scott Moore is an IT Specialist at IBM Rochester and an OS/400 and Java expert.

Lars Strandner is a Technical Domain Expert at Intentia R&D, Sweden.

Lars Wettergren is the Director of the Intentia IBM Competence Center at Intentia R&D, Sweden.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

        ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ► Use the online **Contact us** review redbook form found at:

    `ibm.com`/redbooks

- ► Send your comments in an Internet note to:

    redbook@us.ibm.com

- ► Mail your comments to:

    IBM Corporation, International Technical Support Organization
    Dept. JLU Building 107-2
    3605 Highway 52N
    Rochester, Minnesota 55901-7829

1

Introduction to Intentia and Movex Java

Intentia is one of the world's leading suppliers of enterprise applications. Its business concept is to develop and make business processes more effective by combining software and know how. Enterprise Application Movex is one of the most advanced systems on the market. It is available in a number of industry applications to meet the unique requirements of different industry sectors. Movex is another cornerstone of the Intentia solution that provides a unique set of tools, methods, and functionality for configuring and implementing enterprise applications.

This chapter introduces you to Intentia and presents an overview of its enterprise solution, Movex. It also shows you the value and benefits that Movex can bring to your organization.

1.1 Intentia's history

Since 1983, Intentia grew from a newly founded company in Sweden to one of the world leaders in enterprise management systems. Intentia's success is based on its ability to link software expertise and industry knowledge with an understanding of business processes.

Movex was originally designed to work locally and globally. It allowed people in different countries and regions to access the same system at the same time, in their own language.

Intentia customers around the world have declared Movex to be the best enterprise management system on the market. These loyal customers will help Intentia to continue to drive the development of Movex forward.

1.2 The Intentia solution

Implementing an enterprise resource planning (ERP) system is not simply a matter of loading software on the computer, getting it to run smoothly, and training personnel to use it. Intentia wants its customers to truly justify their investment by making real improvements in their business and taking full advantage of the potential of new applications.

Intentia helps its customers carry out the specific actions to make any implementation project a success. The configurations and implementation methods offered by the Intentia solution help Intentia customers avoid simply automating inefficient old processes and making only limited improvements.

1.3 Introduction to Movex Java

Intentia's solution improves their customers' business processes. It consists of:

- ► Movex component repository (the software)
- ► Enterprise Process Manager (EPM): Process mapping and configuration tool
- ► Implex: The implementation method
- ► Professional services

Together these parts form an optimum set of knowledge, tools, methods, and functionality for a successful configuration and implementation of Movex.

All parts are fully integrated into a single system. This results in higher product and implementation quality and greater effectiveness when upgrading your system with new or enhanced functionality.

Movex component repository: Selecting the right parts

The Movex component repository holds more than 1,000 business components, including generic components and industry-specific components that are used to support the needs of different industries. By selecting and configuring the appropriate components, the solution fully supports your unique business processes.

Enterprise Process Manager: Using the best tools

The Enterprise Process Manager manages process design and application configuration. The EPM consists of the Enterprise Process Designer and the Component Configurator. The Enterprise Process Designer, a fully integrated tool for mapping business processes, is used to design new processes for a customer or redesign their existing processes. The Component Configurator is an advanced tool that is used to set up a tailored execution environment of Movex.

Implex: Putting it all in place

Implex is what Intentia calls the method and tools it developed for implementing Movex. Implex can be divided into five phases. Each phase has predefined tasks to be carried out and goals to be achieved. Each phase ends with a quality assessment and milestone that must be approved before implementation can advance to the next phase.

Implex is based on experience from over 4,000 implementation projects worldwide. Once implementation is completed, you receive more than a system that is up and running. You gain a solid foundation for quality certification at your company.

Professional services: The people behind the product

Intentia's professional services offering consists of project management and business and technology consulting. Intentia's business consultants are experts in such areas as logistics, production, finance, and human resources.

1.4 Why Movex

Today, Intentia holds a competitive position. This position was attained by steadfastly offering an open-ended product and powerful configuration tools through a global organization that uses uniform methods for implementation. The Intentia solution further strengthens Intentia's position and affirms its role in the industry as a supplier of fully integrated ERP systems.

There are many advantages of providing such an integrated system. For example, the quality is higher, and the system can be upgraded with new or enhanced functionality more efficiently and with more precision.

The introduction of an ERP system is a critical step for a company. Taking the right steps from the start and obtaining a measurable gain in business performance requires a tried and proven implementation method and efficient information processing. It is just as vital to have rapid and reliable deployment to minimize delays in operations and to pay back customer investments as soon as possible.

1.5 Conclusion

Intentia's vision is clear: To be the most respected company in helping customers improve their business processes. Intentia's objective is to make the complicated simple.

Intentia continues to develop Movex to be the best enterprise management system and Implex to be the best and most cost-effective implementation methodology. Intentia will continue to exploit its unique combination of know-how in application and business skills.

2

Movex Java overview

Movex offers the key to success with its e-collaboration applications, which are all enabled and supported by e-business. These applications target:

- ► Customer relationship management (CRM)
- ► Enterprise resource planning (ERP)
- ► Partner relationship management (PRM)
- ► Supply chain planning and execution (SCP&E)
- ► Business performance measurement (BPM)

Intentia is also a leading supplier of collaborative trading portals and corporate portals.

The Movex component repository contains more than 2,000 business components. It includes generic components as well as industry-specific components used to support the needs of different industries. By selecting and configuring the appropriate components, the solution fully supports your unique business processes.

This chapter describes the internal arrangement and structure of Movex. The structure described here represents the structure in which Movex is delivered to the Intentia partner network.

2.1 Movex: A multi-dimensional e-collaboration application suite

This section outlines the Movex multi-dimensional architecture. Movex can be scrutinized from various angles as shown in Figure 2-1.
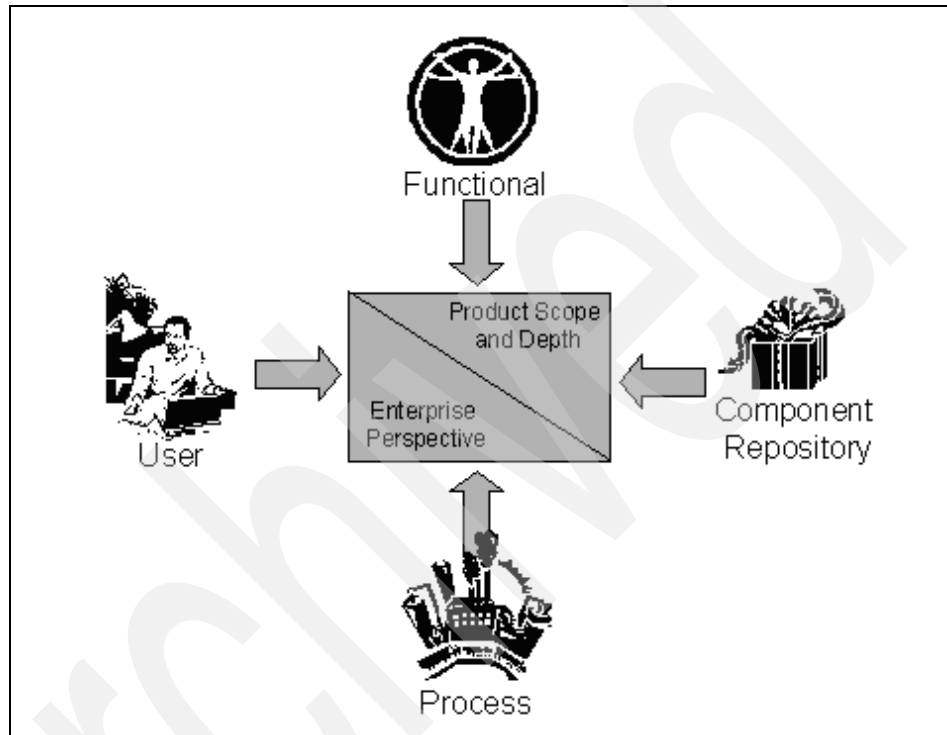
Figure 2-1   Movex multiple view support

Movex can be viewed from the following perspectives:

► **Functional**

The conventional presentation of Movex describes all of the functional capabilities in Movex. For a more detailed description, see the "Movex Application Summary", which is available on the Intentia Wire.

► **Component repository**

The component repository is the Movex internal organization and architecture. It serves as the basis for mapping Movex application components to business processes in the Enterprise Process Manager (EPM). The base iSeries menus are aligned with the component repository. For the sake of consistency and clarity, this means that every program and file only occur once and are connected to a component group.

► **Processes**

This refers to business processes that are mapped out in the EPM and can actually be seen as the conclusion of an implementation project. The generic process model is used as an outline for presenting Movex. It also serves as a starting point for various marketing oriented publications.

► **User**

This perspective refers to the customized Movex menus.

2.2 Movex component repository

To make it easy to navigate in the Movex component repository, components are logically organized into three categories:

► **Application**

An application represents a major functional management area in an enterprise. A set of components groups is connected for each application.

► **Component group**

A component group consists of a group of components. It is used to refine navigation within Movex. A component group generally defines and describes a set of components.

► **Application component**

An application component represents the basic building block for configuration of a user-specific application. The behavior of a component is exposed by its methods. Components can be configured to meet specific customer requirements, such as operations to be carried out, choice of workflow, and data fields to expose. An application component consists of:

– Business component
– Documentation component
– Education component

A Business component conforms to the notion of a business object, which is a representation of a real-world artifact, for example, customer, item, warehouse, budget, order, etc. Business components, in turn, often use finer-grained components to contribute to their responsibility.

Table 2-1 shows the applications and component groups in the component repository structure.

Table 2-1   Applications, component groups in component repository structure

	Application		Application
	Component Group		Component Group
SMS	Sales and Marketing	SLS	Sales & Distribution
	Contact Management Opportunity Management Marketing Management	COP PCR COS COB CDS COQ SST TPL	Customer Order Processing Product Configurator Sales Prices & Discounts Bonus & Commissions Customer Delivery Schedules Sales Quotations Sales Statistics & Performance Transportation Planning
PJM	Project Management	SRV	Service and Rental
PJQ PJP	Project Quotations Project Processing	SEP SAG STR	Service Order Processing Service Agreement Rental Agreement
MAI	Maintenance	WHS	Warehouse Management
MCO PRM WOP MCM ICC DIM	Maintenance Order Processing Preventive Maintenance Work Order Processing Maintenance Performance & Costing Equipment & Component Structure Diagnostic Management	WAC WPI WIO WLA WIS	Warehouse Activity Control Physical Inventory Internal Orders Lot & Allocation Control Inventory Statistics
PDM	Product Management	RPL	Resource Planning
PDA PCO ECM SDS TDO	Product Data Product Costing Engineering Change Management Material Safety Data Sheet Technical Document Management	FOR MAP MPS RCP CRP DRP	Forecasting Material Planning Master Production Scheduling Rough-Cut Capacity Planning Capacity Requirements Planning Distribution Requirements Planning
MAN	Manufacturing	APS	Advanced Planning and Scheduling

	Application			Application
MOP PST LIC	Manufacturing Order Processing Production Statistics & Costing Laboratory & Inspection Control	APP		Advanced Production Planner
POP PDS POQ PSS	Purchase Order Processing Purchase Delivery Schedules Inquiries & Request for Quotations Supplier Evaluation & Statistics			
FIM	Financial Management	FIC		Financial Controlling
GLR RGR ARL APL FAS CUR GCO MUC	General Ledger Report Generator Accounts Receivable Accounts Payable Fixed Assets Current Assets Group Consolidation Multiple Unit Coordination	BUD CFM CAC TAC		Budgeting Cash Flow Management Cost Accounting Time Accounting
HRE	Human Resources	PRS		Personnel Services
HRS HRR HRD	Personnel Skill Management Personnel Recruitment Personnel Development	PAY TIM TEX		Payroll Administration Time & Attendance Travel Expenses (Sweden)
MIF	Management Information	INT		Interoperability Support
BPW	Business Performance Warehouse	API LNI DCI CAD		Application Program Interfaces Lotus Notes Integration Data Collection Interface CAD Integration
MEB	Movex e-business	SYS		System Foundation
WEB EDI	Web Enable EDI Enable	OUT SEC ENV CRS		Output Management Security Management System Environment Cross Application Management

2.3 Enterprise Process Manager

EPM is a fully integrated process-mapping tool, which manages process design and application configuration. The tool is used for designing new processes or redesigning current ones. It is also possible to map responsibilities to the processes.

Enterprise Process Manager consists of the Enterprise Reference Model (ERM). The ERM is a repository of business processes that is used as a reference when improving current processes and designing new ones.

A component connection defines which components are to be used where in the process. Workflow configuration sets the execution behavior of each component and the execution interaction between components. Screen design provides the ability, if necessary, to redesign the specific screens suggested by the configured systems.

2.4 Implex

Implex is the implementation method developed specifically for the implementation and management of all activities within an implementation project of Enterprise Application Movex. Once implementation is completed, you will have received more than a system that is up and running. You will have a solid foundation for quality certification at your company.

With the Intentia approach, customers work with the same team of dedicated professionals who develop and implement the software.

2.5 Logical deployment dimensions

Movex Java can be in a country's local language. Apart from the language-dependent view and print definition files, different databases for different languages can be used on the same system. The advantage is that the user, when using database tools such as Query/400, receives the field descriptions in their own language. This is helpful considering that the field descriptions mostly correspond to the fields on screen, the documents, and the lists.

> **Note:** When using a Multi Unit Coordination (MUC) solution, you may only use one database.

Companies and divisions

Movex Java contains a number of levels where data is managed. Each level used to manage data represents a logical level. For example, Movex distinguishes between enterprise structure and stock levels. The abbreviations used are referred to as *field names*. All relationships under the various levels are considered one-to-multiple relationships.

Multi-site and single-site

In Movex, an enterprise is designated as multi-site if logistics, production, and sales are managed in common for a number of legal entities. For example, the stock levels are referred to the company, but the bookkeeping is (depending on facility) carried out in different divisions (legal entities). If there are such companies, separate bookkeeping and separate logistics and sales are necessary since they will normally be handled as separate companies in Movex.

A single-site installation is an enterprise consisting of one company (one legal entity). In this case, the company corresponds to the division, but in Movex, a division must be defined (for establishing a one-to-one relationship).

A multi-site installation is designated as an MUC installation. In cases where, at the time of system setup, project management is unsure about whether one or more "divisions" will be defined in the future, you must register the company as an "MUC company" at the outset.

2.6 Deployment options for Movex Java on iSeries

Movex Java is a server-centric application that resides on the iSeries platform. This means that it uses the integrated DB2 database that is included in the operating system. This allows the Movex Java technical components (for example, physical files, logical files, program objects, etc.) to take full advantage of the server-based technology.

The presentation components are the main target for a wide range of distribution options. The Movex Java presentation tier supports intranet, extranet, and Internet usage with products like Movex Explorer, Movex Web Explorer, and the e-business applications. Each can reside on the server topology (for example, file server, Windows Terminal Server, etc.) that is best suited for a specific situation.

Movex Java also supports such mechanisms as IBM MQSeries, distributed data management (DDM), clustering applications, and the high availability products that are available on the iSeries. Splitting the application across several servers must always be synchronized with the usage of the application from a business component standpoint before all the benefits are noticeable. This also applies to the database. However, remember to take real-time considerations into account before using this as an efficient alternative.

The architecture in Movex makes option implementation adaptable to a wide range of situations and demands. Along with the interoperability and national language support (NLS), Movex Java on the iSeries has proven to be a solid and highly configurable mission-critical application for small local installation to multinational sites.

Based on how the functional components of application and database servers are grouped, a variety of hardware configuration scenarios is possible. The following sections discuss these options in a Movex Java environment.

2.6.1 Server-centric implementation

In a server-based configuration, the application and database server functions are installed on a single iSeries server. The considerable scalability of the iSeries server range allows adequate capacity for many organizations to implement Movex Java through a centralized server.

2.6.2 Client/server implementations

In a client/server configuration, the application and database server functions are installed on the iSeries and the client portion of the application on a stand-alone Windows server or Integrated xSeries Server for iSeries.

This approach provides a greatly increased capacity for growth for Intel-based interfaces. The application server can reside on multiple machines, due to the Movex Java Super Dispatcher.

3

Application architecture

The architecture of a system defines its broad outlines and the precise mechanisms used. Architecture is a term applied to both the process and the outcome of planning and specifying the overall structure, logical components, and logical interrelationships. It is extremely important with a well-crafted architecture for an application systems' complete life cycle from development to implementation and operation.

The Movex robust architecture grows as your business grows. It allows you to run your business instead of running into problems each time technology changes. The architecture is based on the principles that are described in this chapter.

3.1 Configurability

Each business component in Movex can be configured and reconfigured to suit changes in the customer's business. Also, the forward compatibility that Movex provides ensures a future-proof solution that can be migrated from version to version. In this way, Intentia customers can keep pace with technological changes at all times.

3.2 Object orientation

Movex Java is a modern application built on object-oriented mechanisms on all levels of the object model. Here are examples of four of the most important constructions:

▶ **Encapsulation**

Movex business components are not "aware" of the type of interface or database that is used. You can add or remove without changing the business logic.

▶ **Inheritance**

The inheritance mechanism is used to enforce standard behavior throughout the whole application and to maximize reuse of functionality within the object model. Subclassing also makes it easy to benefit from and extend existing functionality.

▶ **Polymorphism**

A good example of polymorphism in the architecture is the control mechanisms in the business components. The well-defined interfaces make the method (actually defined in a super class) general throughout the system and, at the same time, make the components easy to replace.

▶ **Reusability**

By basing Intentia's technical architecture on layered components, Intentia has significantly reduced the amount of code in the system. This improves product quality and enables Intentia to more quickly deliver new functionality.

3.3 Portability

Because it is programmed in 100% Pure Java, you can transfer Movex from one operating system to another with no or few required changes. This allows your enterprise to have the right technical platform for the right size and type of business. And as new technological advances take place, you can move your solution from your old technical platform to your new one. This minimizes your IT costs while maximizing your IT agility.

3.4 Scalability

Since the Intentia customer base consists of a wide variety of organizations ranging considerably in size, Intentia has built a scalable solution based on N-tier architecture. It can operate on a large multiprocessor iSeries and support several thousands of users down to a single user with a Windows laptop.

Figure 3-1 shows the layers and tiers within the architectural model of Movex.
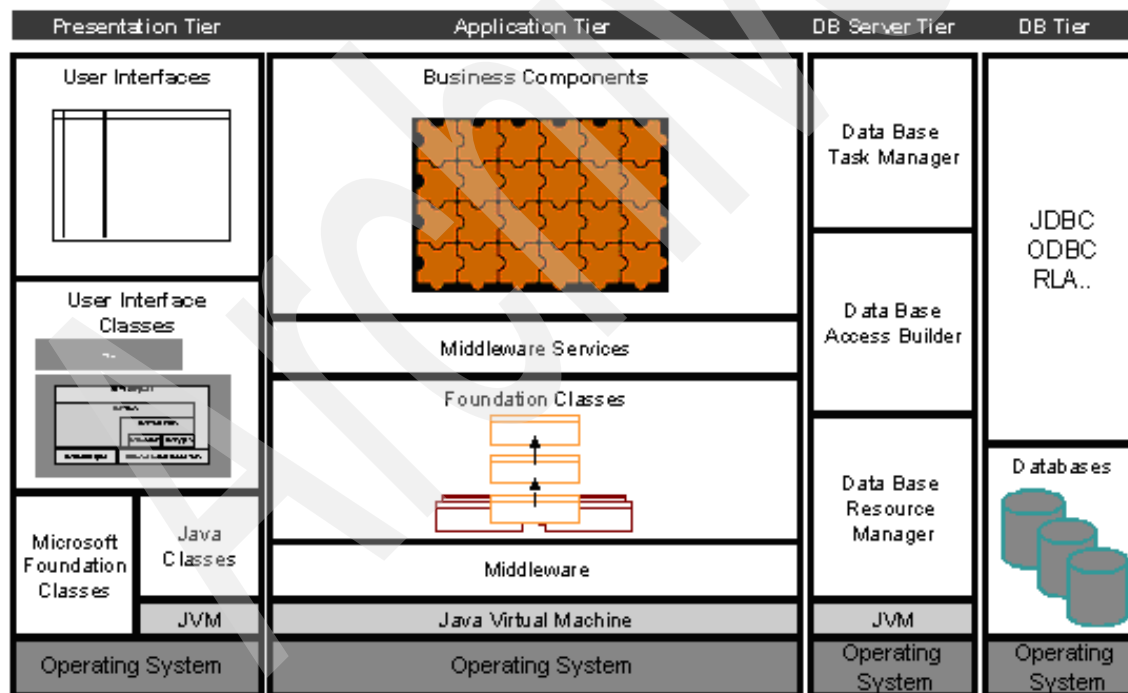
Figure 3-1   The layers and tiers within the architectural model of Movex

3.5 Layered architecture

Layered architecture constitutes the framework for how groups of components relate to each other and where in the dependency chain they belong.

The notion of reusability and portability is addressed by the concept of layered architecture. Components are organized in a hierarchy of layers. The most generic components are in the lower layers (reused by many), and the most specific components are in the upper layers (reused by few). Each layer provides a well-defined interface to the above layers.

An important rule is that components can only depend on other components in lower layers. This minimizes the complexity of dependencies between components. This approach is the key to achieve a high degree of reuse in a business application. That is designing the architecture for the application in a layered structure and placing the most generic (most reusable) components in the lower layers.

Portability is actually a consequence of using this approach. This is done by totally isolating the platform-dependent components. For an example, consider components related to operating systems platforms or RDBMS platforms.

3.6 N-tier architecture

The concept of an n-tier architecture primarily addresses the notion of scalability. N-tier architecture provides flexibility in the configuration of scalable topologies such as multiple application servers, multiple database servers, and Web servers. N-tier architecture also constitutes the well-defined borders between tiers. This enables flexibility to add new types of user interfaces (clients) or alternative databases.

In addition to scalability, the layered architecture of Movex offers Intentia customers component reusability and portability. This constitutes the framework for how groups of components relate to each other and where in the dependency chain they belong.

Not only is layered architecture the key to achieving a high degree of reuse for Movex, it also provides increased portability by totally isolating the platform dependant components.

Figure 3-2 shows the n-tier architecture and layered architecture within the architectural model of Movex.

Presentation Tier	Application Tier	DB Server Tier	DB Tier
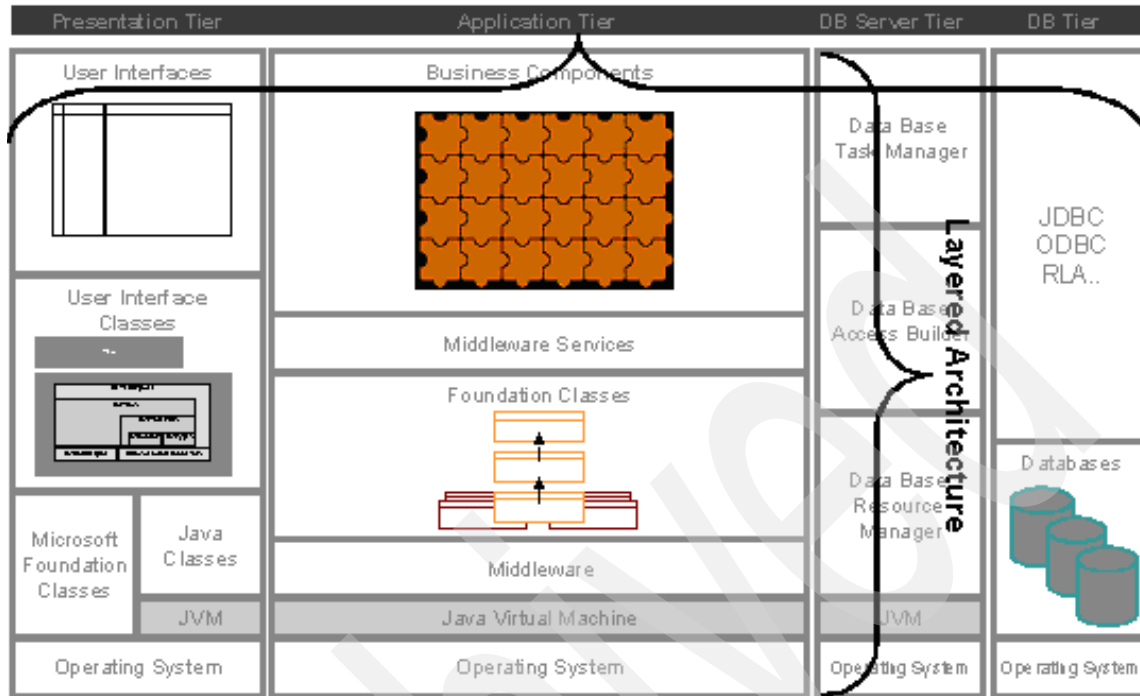
Figure 3-2   N-tier architecture: Layered architecture in an architectural model of Movex

3.6.1 Presentation tier

The presentation tier handles any kind of presentation service to the application tier. It is important to note that the presentation logic is separated from the business logic. This separation means that new kinds of user interfaces can be added to the system and affect only the presentation tier classes and not the business logic.

The separation of concerns used in a true n-tier architecture is key to achieve transparent isolation between the different tiers.

3.6.2 Application tier

Business components at the highest level in the application tier represent the artifacts that constitute the Movex business logic. These components are subjects for configuration to suit the specific requirements of a customer by using the Movex Configuration Series of products. These components are further described in the object model.

The Foundation classes represent the environmental parts needed to run a business application. Such tasks as thread handling and queuing are carried out by these classes.

3.6.3 Movex database (MDB) server tier and database tier

The separation of concern approach also applies to the MDB server tier and the database tier. By the inclusion of a middle-tier database server tier, different relational databases can be used without affecting the actual business logic.

A key benefit of the database server tier in the architecture is data store independence. This means that applications are uncoupled from the storage technique used and are unaware of the physical location of the data store being used.

Therefore, implementors of Movex applications are free to switch to the latest data storage technology, support multiple storage techniques in the same application, and apply their applications on a variety of system configurations without recompiling their code. Data is encapsulated within business data objects.

3.7 Object model

The object model, with its programming model and class hierarchies, defines the software standard. This design was proven before starting the "industrial" transition. The transition was made during an iterative and incremental development process including analysis, design, implementation, and testing.

The highest layer in the architecture relates to the business components in Movex. The object model describes how the components are constructed.

The class hierarchy shows that business component classes at the bottom inherit from parent or super classes higher up in the model. This way of structuring the general or commonly used code patterns higher up in the hierarchy has several benefits:

► Code reduction
► Forced use of standard
► Higher quality
► Increased development efficiency
► Isolation/encapsulation of complex environmental code

Figure 3-3 shows the Movex Java object model.
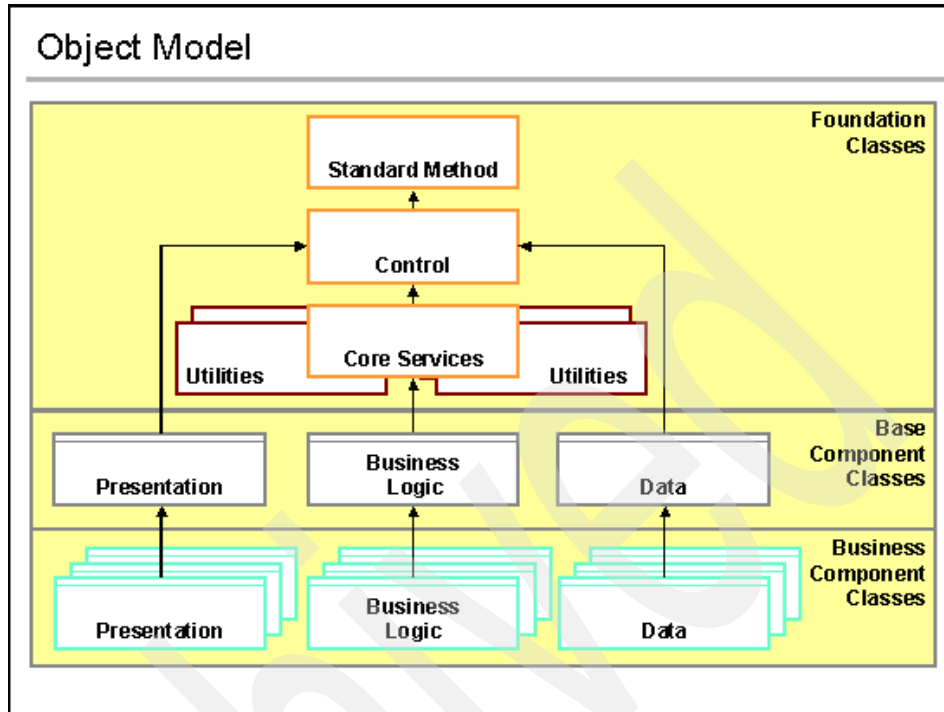
Figure 3-3   The Movex Java object model

This object model is simplified and structured to explain the most important parts
of the class hierarchy. An example of this is that the Foundation classes are
made up of three major classes:

► Core Services
► Control
► Standard Methods

The Foundation classes also include a number of supporting utility classes.

3.7.1 Foundation classes

Foundation classes are central in the object model. They establish the Movex
standard behavior. Because all Business Component classes inherit from Base
Component classes, they also inherit, or aggregate, from different Foundation
classes. Apart from being the standard, Foundation classes with the Utility
classes, including components for thread management and messaging, define
the rules and behavior for the environment. This means that Foundation classes
are often middleware related.

3.7.2 Base Component classes

The Base Component layer is where all standard code that applies to all components lower down in the inheritance chain is placed. This is often referred to as *generic code*. Should a widespread modification need to be carried out that relates to all presentation classes, a programmer only needs to adjust the presentation class in the Base Components layer. This change is then automatically inherited by all the presentation subclasses in the Business Components Layer. This is also true for the Business Logic and Data classes.

This results in greater productivity, quality, and development efficiency and substantially reduces the amount of code in the system.

3.7.3 Business Components classes

The Business Components layer includes business components that meet the requirement of a substantially configurable offering. Business components are subjects for configuration and are usually grouped into coarser-grained Business Domain components.

An essential aspect of Business Components is that their interfaces expose the behavior of the actual components through an application programming interface (API). This is of major importance, since the API serves as the enabler for integration and interoperability between applications in numerous ways.

These interfaces provide the openness needed for interoperability with third-party components, using industry-wide standards such as Microsoft's Component Object Model (COM) standard, Sun's Enterprise JavaBeans (EJB) standard, C++, and the emerging Business Quality Messaging (BQM) standard.

100% Pure Java has been used as the implementation language of all business components and related classes. This guarantees the highest degree of portability for Movex with exactly the same source code across different platforms.

The Movex object model maximizes development efficiency and development productivity. The inheritance mechanism forces the use of standards implemented and facilitates rapid modifications to the application when required.

3.8 Technical innovations in the Movex Java architecture

In the process of creating Movex Java, several characteristics required by Intentia either did not exist or did not meet its technical or functional requirements. Therefore, in creating the Movex Java architecture, Intentia R&D created software components to bridge some gaps in the available technology.

The implementations were developed according to our layered architecture with encapsulated components that can easily be changed or replaced by standard middleware (drivers, Java virtual machines (JVM), etc.) as appropriate solutions become available.

The following sections covers some of the most important Movex middleware constructions.

3.8.1 Super Dispatcher technology

The Movex Java Super Dispatcher is an advanced dispatcher that can be used to dedicate special types of workload to a specific JVM or server. It is used to distribute workload, provide high availability by recognizing when one JVM goes down, and shuffle requests to separate machines.

3.8.2 High performance JDBC driver

The n-tier architecture in Movex Java provides a structure that can use the best data access mechanism available for any given database or topology. This means that during an onsite installation, Movex Java can use the most efficient data access mechanism provided for a combination of other components.

To bridge the gap where performance of current drivers on some databases is found to be lacking, Intentia R&D developed a database driver that can handle the large transaction volumes.

3.8.3 Intelligent Object Reuse

Movex Java Intelligent Object Reuse is a proprietary, intelligent tuning algorithm for all types of object reuse units in the runtime version Movex Java technology and architecture. All object instances in the runtime version of Movex Java are reused in the architecture and middleware.

Intelligent Object Reuse decreases the frequency in which pauses caused by "old object" reclamation occur significantly for most programs. It also increases reclamation efficiency significantly for most programs running in current JVM. This greatly improves performance scalability of applications that use large amounts of "live" object memory.

3.8.4 Database Connection Optimizer

Movex Java Database Connection Optimizer is a database connection technology. The database access is designed as encapsulated container objects for persistent data rather than as in other technologies, where persistent data is treated as entity beans.

Database Connection Optimizer, where all the transaction handling is managed in the database (single or distributed), releases the middleware from coordination problems in the runtime clusters of each JVM.

The Movex Java high performance architecture heavily uses intelligent database connection pooling in combination with a prepared statement pool. This achieves optimum performance and high-level reuse of allocated resources.

3.9 Movex Java package structure

The package structure implemented for Movex Java is basically divided into three sections:

► Business Logic (binopt)
► MovexCore
► Common

Figure 3-4 shows in which section each package is found. Note that MovexCore contains both Base and Foundation classes. This structure is the same in all different scenarios where the Movex system occurs. Above this package, structure different scenarios have their own file system structure to facilitate different needs.
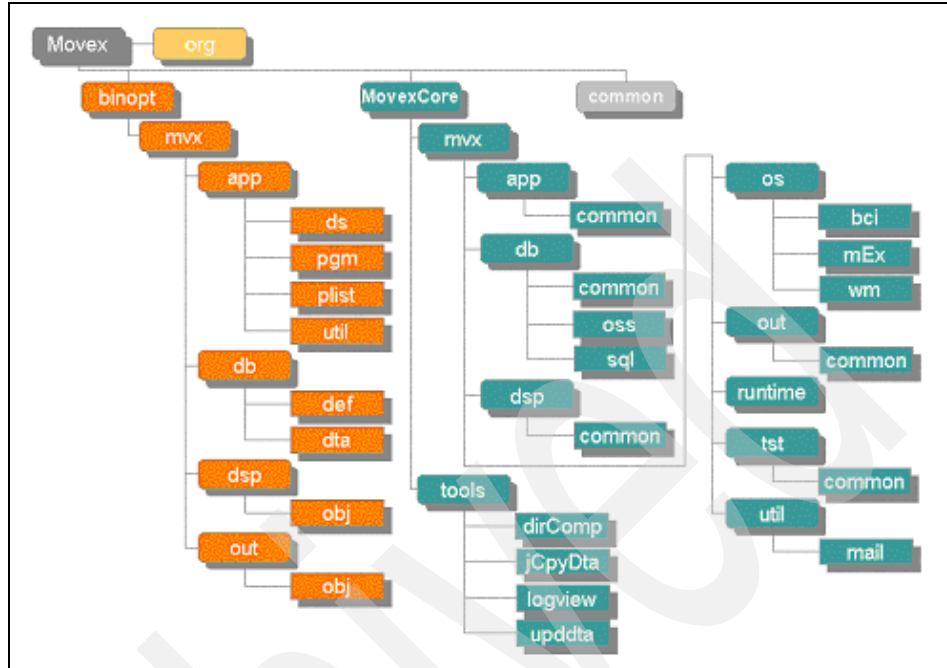
Figure 3-4   Movex package map

3.9.1 File system structure overview

The Java package structure is normally implemented in the file structure used to store the application. Figure 3-5 shows the Movex file system structure used above the package structure that was mentioned in the previous section. Each entry on the left side of the figure is described by its matching description on the right side of the figure.

This structure may be subject to changes. Note that the root folder for the runtime environment is the same independently of the release of Movex Version 12 that is implemented.

Movex_v12	**Root folder for Movex version 12.x**
Ccusname	**Customer Modified (one per cusname)**
databaseC_cus	Customer modified database components
SPvvrsCydddn_cus	Service pack consolidation level for customer modifications
vvrsC_cus	Base folder for a given version of customer modified files
environment	**Runtime environment definitions**
Java13re	Bundled JavaSoft JDK 1.3 components
template	Template structure used when creating runtime environments
foundation	**Foundation Framework and super class structure**
SPvvrsFydddn	Service pack consolidation level for foundation
vvrsF	Base folder for given version of foundation
HUM	**Human Resources**
databaseS_HUM	HUM database components
SPvvrsSydddn_HUM	Service pack consolidation level for HUM
vvrsS_HUM	Base folder for given version of HUM
MEX_12rs	**Movex Explorer**
Help	Help files (can be placed on another file server or client)
Languages	Language files (can be placed on another file server or client)
Menus	Menu files (can be placed on another file server or client)
ViewDefinitions	View Definition files (can be placed on another file server or client)
MOM_12rs	**Movex OUT**
MovexLng	Language files (can be placed on other file server or client)
outDefinitions	Output definition files (can be placed on other file server or client)
MVX	**Movex (standard)**
databaseS	Standard database components
languageS_ll	Language components (for example, MVXCONGB, MVXMSGGB, etc.)
SPvvrsSydddn	Service pack consolidation level for standard
vvrsS	Base folder for given version and release of Movex
Mxx	**Market adoptions Mxx (MSE, MFR, …)**
databaseM_Mxx	Mxx database components
SPvvrsMydddn_Mxx	Service pack consolidation level for Mxx
vvrsM_Mxx	Base folder for giver version of Mxx (for example, MGB, MFR, etc.)
utilities	**Utilities and Tools**
OS400	iSeries and OS400 specific utilities and tools
Solaris	Not used on OS400
SPvvrsUydddn	Service pack consolidation level for utilities
Windows	Not used on OS/400

Figure 3-5   Movex file system structure

3.9.2 Movex Java implementation scenarios

Another representation of the structure is in which scenarios the code may appear. The following list gives an overview of these different combinations:

► **Standard Movex runtime**

The pure runtime environment facilitates the possibility to execute Movex with maximum performance. Most parts are installed as JAR files to avoid problems with accidental mismatch of executables. There is still the ability to handle individual fixes by using the file structure and correcting the code as necessary within it. Routines are developed to ensure the consistency and seamless updates are added to the installation process.

► **Movex HUM**

Movex HUM is stored under a separate subfolder in the main catalog for Movex, as shown in the structure example in Figure 3-5. This is the recommended way for storage to simplify the handling.

► **Market modifications**

Market modifications are stored under a separate subfolder in the main catalog for Movex, as shown in Figure 3-5. This is the recommended way for storage to simplify the handling.

For inheritance reasons, an extra level is introduced to the package mvx.app.pgm, that is mvx.app.pgm.market. Note that in this case the name "market" should not be replaced with the real market code.

► **Customer modifications**

Customer modifications are stored under a separate subfolder in the main catalog for Movex, as shown in Figure 3-5. This is the recommended way for storage to simplify handling.

For inheritance reasons, an extra level is introduced to the package mvx.app.pgm. That level is *customer* as in mvx.app.pgm.customer. Note that in this case, do not replace "customer" with the real customer name.

► **Movex development**

The development environment does not have as many similarities with the runtime environment, at least not on the upper levels. That is because of the fact that the development is more focused upon supporting the life-cycle structure for the Movex system. Deeper down in the file system, the package map *is* recognized.

4

Database architecture

The relational database management system (RDBMS) used on the iSeries server is DB2 Universal Database (UDB) for iSeries. It is fully integrated with the OS/400 operating system. The two products are shipped by IBM and installed as a single entity. The close integration between the operating system and the database allows iSeries developers to implement database functions where they are most efficient. In fact, some of the database functions are even implemented in the hardware.

This high level of integration makes the complexity of a database management system quite transparent to the users of an iSeries server. This enhances the ease of use and management of the two products. Resulting from this integration, separately authorized users are not required for the two components. Also, when OS/400 is started at IPL time, DB2 UDB for iSeries starts automatically.

The Movex database consists of a number of physical files (tables) and logical files (views and indexes) that are structured on the DB2 UDB for iSeries database.

Note: DB2 UDB for iSeries is a member of the IBM DB2 family of products.

4.1 Database representation

Data representation for Movex Java on DB2 UDB for iSeries and OS/400 is in Unicode Worldwide Character Standard. This is different than American National Standard Code for Information Interchange (ASCII) used on other platforms.

iSeries database files are record (or row) oriented. They almost always have subdivisions that consist of data fields (or columns). These objects have an iSeries object type of FILE, with an attribute such as PF or LF. Any application capable of accessing the database can process the contents of a database file.

Nearly all data on an iSeries server is normally stored as relational tables. However, enhancements to the product have allowed for a variety of file systems to be supported on the iSeries server to make it an efficient and effective central store house of information.

4.1.1 Terminology

This section introduces you to the terminology used in an iSeries environment that relates to information storage and retrieval.

Library

All traditional iSeries objects are contained in a special object type called a *library* (*LIB). However, there is a special library called QSYS, which contains the names of all the other libraries.

Physical and logical files

Data in the relational database on an iSeries server is stored in objects called *physical files*. Physical files consist of records (rows) with a predefined layout of data fields (columns). Physical files can have a keyed index in its definition that allows the retrieval of information in a predefined sequence.

Logical files provide a different view of the physical data. They allow data to be retrieved in a sequence other than that specified in the physical file. They also allow field (or column) subsetting, record selection, joining multiple database files, and so on.

An *access path* describes the order in which records are to be retrieved and presented to the application program. Records in a physical or logical file can be retrieved using an arrival sequence access path or a keyed sequence access path. For logical files, you can also select and omit records based on the value of one or more fields in each record.

iSeries integrated file system (IFS)

The iSeries IFS encompasses all file systems currently supported by the iSeries server. File systems supported by the iSeries IFS include the following components:

- ► **Root file system:** The root (/) file system is designed to take advantage of stream file support and the hierarchical directory structure of the IFS. The object names are not case sensitive.

- ► **QSYS.LIB:** This file system represents the traditional iSeries library and file structures. It provides access to iSeries libraries and SQL collections defined under the iSeries QSYS library and all of the iSeries objects under them.

- ► **QOpenSys:** iSeries Open Systems support is designed to be compatible with UNIX-based Open Systems standards, such as POSIX and X/Open Portability Guide (XPG). This file system supports case-sensitive object names.

- ► **QDLS:** This is a document library services system that supports the folder structure and provides access to documents and folders within folders.

- ► **QLANSrv:** The LAN Server file system provides access to the same directories and files that are accessed through the LAN Server/400 licensed program product.

- ► **QOPT:** This is the optical file system and supports stream files stored on optical media, including the iSeries's integrated CD-ROM drive.

- ► **QFileSvr.400:** This is a special OS/400 facility that provides access to file systems residing on other iSeries servers.

- ► **NFS:** This file system supports access to data and objects stored on remote Network File System (NFS) servers other than iSeries. NFS objects can be exported from the NFS server and dynamically mounted by NFS clients.

- ► **QNetWare:** This file system provides access to objects that are stored on a server running Novell NetWare 3.12 or 4.10.

- ► **QNTC:** This file system provides access to the objects that are stored in Windows 2000 either on the Integrated xSeries Server (formerly known as the Integrated PC Server and the Integrated Netfinity Server) or a stand-alone PC server.

4.1.2 Accessing the database and IFS

This section presents an overview of how information is stored and retrieved from an iSeries database and the IFS.

iSeries directory structure

The IFS presents a hierarchical directory structure, which enables users and application programs to access all objects in the iSeries server. Path names are used to access objects. The support is similar to the support that is available in PCs.

Hard and soft links

Each entity in the path is linked to the next entity in the path through *hard links*. An object has a hard link to the directory in which it is created.

On the other hand, *symbolic links* (or *soft links*) merely point to another object described through another path. The object pointed to by a symbolic link is resolved only when the link is used. Therefore, a symbolic link can point to a non-existent object until it is accessed. Also, symbolic links can cross file systems, where hard links cannot.

The Add Link (ADDLNK) command (or ln Qshell command) on the iSeries server enables you to create a link between two objects. Figure 4-1 shows an example of the ADDLNK command.

```
                          Add Link (ADDLNK)

 Type choices, press Enter.

   Object . . . . . . . . .

   New link . . . . . . . .

   Link type . . . . . . . . . *SYMBOLIC *SYMBOLIC, *HARD




 Bottom
 F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
 F24=More keys
```

Figure 4-1   Add Link (ADDLNK) command display

The first parameter, Object (OBJ), specifies the path name of the object to which you want to add a link. This object must exist unless a symbolic link is being added. The second parameter, New link (NEWLNK), specifies the new path name that can be used to refer to the object. The new name must not exist.

QFileSvr.400

The QFileSvr.400 file system is used to communicate between the application server and the database server. Other iSeries servers that can be communicated with must be identified in the QFileSvr.400 directory. A soft link can be defined to point to the file system on the other iSeries servers.

4.2 Movex database structure

Movex database consists of a well-defined set of physical and logical files. This section offers a general description on the different types of physical files and the logical files that make the Movex standard runtime database.

Note that this is described from a runtime perspective. For development or modified environments, refer to the information and naming conventions on Intentia Wire. It is located under **Intentia Solution-> Deliverables-> Instructions**.

4.2.1 Physical files in Movex

In Movex, physical files only contain data and never key definitions. Because of this, access to physical data is always sequential. Data is processed via a logical file with a corresponding access path. This concept has been chosen to make changes and extensions of unique key definitions simpler.

► **Master tables**

Master tables contain general data that is used in most parts of all business functions. Examples are customer master, items master, supplier master, etc. Master tables are frequently used, considering there is a low frequency of changes to the data.

► **System files**

System files contain data that has a connection to the runtime environment or that use system-like functions such as users, access control lists, authority, etc. The usage characteristics are similar to those for master tables. However, there is distention since there is no direct connection to functional entities.

► **Transaction files**

Transaction files are used to store data in or between business functions. Data in these files is used, and the changes are intensive.

► **History files**

History files are used when performing Movex archiving from transaction files on data that is considered by business functions to be used for historical data or statistics reasons. Transfer to the history files is often done on a scheduled basis and is initiated by a user or on a weekly, monthly, or other routine.

► **Work files**

Work files are used for temporary storage of data that is being processed to feed a specific business function. Generally work files should be empty when the system is in backup or offline mode.

4.2.2 Logical files in Movex

In Movex Java, logical files are used to access the records stored in physical files. These files contain all the key information. Besides the keys (sorting), select or omit definitions are also partly entered. Because of this additional selection, only certain records can be selected or omitted. This allows for optimal processing of online inquiries, among other items.

Logical files always receive the same names as the physical file in which they are based, but they are numbered with the seventh and eighth positions. Logical files with the sequence number 00 receive the main key, with some exceptions. For an item master table, for example, the main key is the item number. The main key is meant to be the unique definition of a record. The key always includes the company number (first key field), but the division is indicated only in files that require this number.

The logical files always contain all the fields according to the physical file on which they are based. This is a prerequisite for updating the files in update and write operations.

4.3 Journaling on the Movex Java database

Movex Java database setup in relation to journaling is different than Movex ThisGen. In Movex Java, a basic transaction model is placed under the functionality, even if the programs are not fully adapted to a new transaction model than earlier releases. This means that if a transaction, from the database point of view, is not fully executed, the transaction is undone.

On the OS/400, DB2 UDB for iSeries is implemented to handle attempts, using journaling, to make transactions to a physical file. From an overview perspective, this is performed in two steps:

1. Create the environment needed for the journals and journal receivers.
2. Start journaling for the Movex databases.

There is also a function to stop journaling for a database. The creation step is normally performed under the Movex installation and setup, while the other functions may be used through normal operations.

Note that journaling must be started to use the database from the application. If journaling is used for other purposes, such as in a high availability solution scenario, that set of journals may be used instead of the ones handled in the Movex utility library.

The Intentia Wire explains the full instructions for how to use these functions. On the Wire, look under **Intentia Solution**-> **Deliverables**-> **Installation**.

4.4 Unicode

Another difference between Movex ThisGen and Movex Java is that the Movex Java database is delivered in Unicode format, which is officially called the Unicode Worldwide Character Standard. It is a system to interchange, process, and display the written text of several, diverse languages worldwide. It also supports many classical and historical texts in a number of languages.

Currently, the Unicode standard contains 34,168 distinct coded characters, derived from 24 supported language scripts. Unicode allows Movex customers to store any language in the same database tables.

The iSeries server supports the storage and access of Unicode character strings within the database. Since Java runs in Unicode, no conversion is needed on character fields when transferring data from the database to the Java runtime. This is yet another performance advantage that helps the iSeries and Movex Java excel.

Movex Java can also run in any country or region without deciding in which CCSID to store the data. Currently with ThisGen, a consciencious decision must be made in regard to which CCSID to store the data in to avoid "lost" characters. Since a Movex customer can span many countries or regions, this is a more difficult task than to choose the host country (region) CCSID. NextGen simplifies this scenario considerably. When the data is stored in Unicode, it can be displayed in almost all languages, without any advance planning. This is a major advantage because dealing with national language support (NLS) issues can

take a significant amount of resources from an international software company. With the Movex WebShop product, the customer base of Movex Java is the world, and with a Unicode database, the country (region)-specific modifications can be kept to a minimum.

The size of the database increases when moving to a Unicode database from a single byte EBCDIC database, but the result is not two times larger as you may expect. Of course, most ASCII and EBCDIC databases store a character in 1 byte, and Unicode stores a single character in 2 bytes. But, data that is stored in numeric or binary data types is not affected. Much of the size of a database can be linked to the metadata of the database tables (for example, the size and types of the columns within the database). The size of this is not affected when the database moves to Unicode. The net is that the database becomes larger in terms of bytes used, but not twice the size.

4.5 Database access methods used by Movex Java application

Since Movex Java is a true n-tier client server application, it (the business functions) can access the database (data) in a dynamic but optimal way for any given situation. For the iSeries server, this means usage of the integrated access method provided by DB2 UDB for iSeries record level access (RLA).

4.5.1 Record level access

The only platform-specific code within the Movex Java application resides in a layer of database access. On the iSeries platform, the customer can choose to use either native I/O (also called record level access) or DB2 UDB SQL access.

To the Movex Java end user, the only difference is the superior performance and scalability advantages of RLA. This database access layer has been abstracted to a defined internal interface. Therefore, the method of database access is invisible to the developer as long as they code to the interface.

During the installation, the MVXRLA file is downloaded to the iSeries server. This component is then used when specifying usage of RLA as an access method in the Movex.properties file.

4.5.2 SQL

Despite the fact that Movex Java uses RLA for most of the database access from the application's database layer, Movex Java also uses standard SQL features in OS/400 for some queries, where performance was a concern with RLA. The application database layer makes this decision automatically at runtime.

4.5.3 Database related settings in the Movex.properties

In the Movex.properties files, several settings relate to the database. The most important settings are:

Property	Description
db.con.source	Specifies the ODBC name that will be used to reach the database. On the iSeries platform, this should be the entry made on the Relational Database Directory Entry (WRKRDBDIRE) during installation.
db.con.user	Tells the user profile that should be used to connect to the database from the application. The default value is MDBUSR.
db.con.password	Tells the password of the user that is used to connect to the database.
db.con.libraryList	Specifies in what order the different libraries (schemas) that constitute the database should be searched to find the desired tables.
db.con.defaultschema	Tells the schema to should be used when tables are created in runtime.

Other database-related settings offer the possibility to select database access method, to switch on trace of database access, and to specify more development- or troubleshooting-related settings. Normally these settings do not need any attention from a pure runtime aspect.

5

Java overview and iSeries implementation

This chapter discusses the Java platform architecture. The first part shows the Sun Microsystems, Inc. definition of the Java platform and its implementation in the Java Development Kit (JDK).

The next part presents the Java implementation on the iSeries platform. It explains some of the iSeries-specific aspects of this implementation.

The final part examines how Intentia exploits some of the iSeries advantages in the implementation of the Movex Java implementation.

37

5.1 Java platform

Java is a full-fledged object-oriented (OO) programming language. The Java language syntax is similar to the syntax of C or C++, while its behavior is more closely related to Smalltalk. Some features of the Java language, such as strongly typed data definitions, no direct memory addressing through pointers, or automatic garbage collection, make it well suited to develop robust Enterprise core business applications. Although Java can be seen as yet another programming language, it is easy to learn, simple to debug, and has reduced maintenance costs.

The main advantage of Java is its cross-platform portability. Java is portable, because of the core Java application programming interfaces (APIs) or Java classes that provide a rich set of platform-neutral APIs. The existence of this set of Java APIs provides the I/T industry with the capability of developing sophisticated, state-of-the-art, client or server Internet-enabled applications that you can deploy and run on any Java-enabled platform. Therefore, Java is not only a promising programming language, it is a software platform that you can implement and run on any existing hardware or software platforms.

You can see the Java platform as the combination of three components:

► Java virtual machine (JVM)
► Java APIs
► Java Utilities

5.1.1 Java virtual machine

The JVM is the centerpiece of the Java platform. It is the "engine" of Java. The JVM is responsible for running Java applications or applets in any given hardware or software environment, and it is platform-dependent. Every company that wants to implement Java on a given platform must implement the JVM on its hardware or software platform.

The JVM generally includes these components:

► Class loader
► Bytecode verifier
► Bytecode interpreter
► Garbage collector
► Java native interface (JNI)
► Other miscellaneous components

Class loader

The class loader is capable of dynamically locating and loading the various classes that the application uses. This is a powerful feature, because it allows programmers to develop applications or applets that consist of many classes that can be provided by several different vendors. As the acceptance of Java grows in the entire industry, many companies are developing standard, ready-to-use software components or *Java beans* that greatly simplify the job of application developers.

The dynamic nature of the class loader simplifies the application packaging process. You are no longer required to go through the complex and error-prone process of building the executable program. You can compile each class separately from the other classes and load each class as required by the class loader.

The class loader determines which classes to load from the CLASSPATH environment variable or the command line argument. The class path is similar to the library list for the JVM. It is a list of directories that the class loader looks at to find a class that it needs. In the start program for a Movex Java configuration, the class path to use is specified in the start script. Additions and changes to the class path are then done dynamically as defined in the library list configuration.

However, for ease of handling and performance reasons, developers tend to package related classes together into what is known as Java ARchive (JAR) files. A JAR file is a compressed (zipped) package that includes several classes. The process of packaging an application into a JAR file is simple and much easier than the traditional building steps that were required when using more conventional languages, such as C or C++.

Often, all of the classes that make up an application are packaged in a single JAR file. For example, all Java Core API classes are shipped within a single file named rt.jar. The class loader is capable of finding the required class within a JAR file and expanding (unzipping) it on the fly.

Bytecode verifier

The bytecode verifier performs extensive checks before running a Java program. This ensures that the Java bytecode has not been altered and that it still conforms to Java security specifications. This involves such checks as type matching.

For example, when an arithmetic operation code is encountered, the bytecode verifier checks that all the operands involved in the operation are of the integer type. If the bytecode does not pass the verifier checks, the JVM throws a runtime exception and the program is terminated. It is especially important to perform extensive checks in an open network environment where someone could run an applet from an unknown source.

Bytecode interpreter

The bytecode interpreter is responsible for reading the bytecode and carrying out the operations that they specify. The bytecode is interpreted on the fly as the Java application steps from one instruction to the next. As new versions of the JDK are introduced, the overall performance of the bytecode interpreter improves, because of the new algorithms that are being developed.

Garbage collector

The garbage collector provides fully automated memory allocation and de-allocation. This is unlike C/C++, where the programmer is responsible for allocating memory to store new objects and freeing unused memory when objects are discarded.

The garbage collector solves one of the main problems found in many C++ applications, which is known as memory leaks. This is one of the most difficult bugs to deal with when developing C++ applications. Often C++ applications fail with an "out of memory" error because of poor memory management.

In Java, the JVM allocates the memory needed when a new object is created. Meanwhile a background task, running in a separate thread, continuously scans the memory and de-allocates space that objects (without an active reference in any of the running classes) occupy. The garbage collector is key to both the performance and the reliability of Java programs.

Java native interface (JNI)

You can view the Java native interface as "glue" code that allows a Java program to start a method that is written in a language other than Java. This interface allows for interoperation between Java applications and legacy applications. However, by using native methods you lose portability. By construction, native methods are written to a specific execution environment and are platform dependent.

Movex uses the JNI, for database access, for performance reasons.

Miscellaneous components

Some Java implementations may include other components. One of the most commonly found components is a Just-In-Time (JIT) compiler. This is often tightly integrated with the bytecode interpreter. It performs additional tasks such as setting aside in memory the real instructions that correspond to the bytecode. Any further reference to a bytecode that was executed once results in the execution of the corresponding real machine instruction that already exists.

JIT compilers also perform code optimization functions to further improve performance. Such functions as inlining (which includes a piece of code in another sequence rather than performing a branch or a call to a subroutine and dynamic dead code elimination) are becoming common. In fact, sophisticated compiler optimizing techniques are being implemented in JIT compilers.

Other functions, such as *reflection* or *serialization*, are also found in a JVM. Reflection in Java refers to the ability of a Java class to reflect upon itself (to "look inside itself"). The reflection technique allows a Java program to inspect and manipulate any Java class. The Java beans "introspection" mechanism uses this technique to determine the properties, events, and methods that a bean supports. You can use reflection to query and set the values of fields, start methods, or create new objects. Java does not allow methods to pass directly as data values, but the reflection technique makes it possible for methods that pass by name to start indirectly.

Serialization is the ability to write the complete state of an object (including any object to which it refers) to an output stream, and then to recreate that object at a later time by reading its serialized state from an input stream. This technique is used as the basis for transferring objects through cut-and-paste and between a client and a server or vice versa for remote method invocation (RMI). It can also be used by Java beans to provide preinitialized serialized objects rather than a simple class file.

The serialization technique is also used as an easy way to save user preferences and application states. Serialization includes information about the class version. Obviously, an early version of a class cannot de-serialize a serialized instance that was created by a newer version of the same class.

5.1.2 Java APIs

The Java platform provides a set of Java classes or APIs that mimic a complete modern, yet platform-neutral operating system. The Java platform, which is based on Sun Microsystems, Inc. JDK, consists of two kinds of APIs. Each is discussed in the following sections.

Core Library APIs

Core Library APIs belong to the minimal set of APIs that form the standard Java platform. Core Library APIs are available on the Java platform, regardless of the underlying operating system. They can run on smaller, dedicated embedded systems such as set-top boxes, printers, copiers, and cellular phones. The core library grows with each release of the JDK.

Standard Extension library APIs

Standard Extension library APIs are a set of APIs outside of the Core API for which JavaSoft has defined and published an API standard.

5.2 Java on the iSeries server

Starting with V4R2 of OS/400, the iSeries implements the Java platform. This implementation fully complies with JDK 1.1.8, JDK 1.2, and JDK 1.3 as defined by Sun Microsystems, Inc. As in any other Java platform implementation, the iSeries also provides these components:

- ► Java virtual machine
- ► Java APIs
- ► Java Utilities
- ► Integrated JVM
- ► Static compilation of class files
- ► Dynamic class loading
- ► Remote Abstract Windowing Toolkit (AWT)
- ► Scalable garbage collector
- ► DB2 UDB for iSeries JDBC driver
- ► Multi-process design point

5.2.1 iSeries Java virtual machine

On the iSeries, the JVM is implemented within System Licensed Internal Code (SLIC), beneath the Technology Independent Machine Interface (TIMI). JVM is an integral part of OS/400. By installing OS/400 (5722-SS1) and the iSeries Developer Kit for Java (Licensed Program Product (LPP) 5722-JV1) via the standard OS/400 Install Licensed Program Procedures, you install a standard JVM on your system. You must install one additional, no charge AS/400 TCP/IP Connectivity Utilities/400 (LPP 5722-TC1) before you can use Java on your iSeries server.

The iSeries Developer Kit for Java is a "skip release" LPP that follows Sun Microsystems JDK future versions as closely as possible. This happens independently from OS/400 versions and releases whenever possible. The OS/400 JVM includes all of these components of a standard JVM:

► Class loader
► Bytecode verifier
► Bytecode interpreter
► Garbage collector

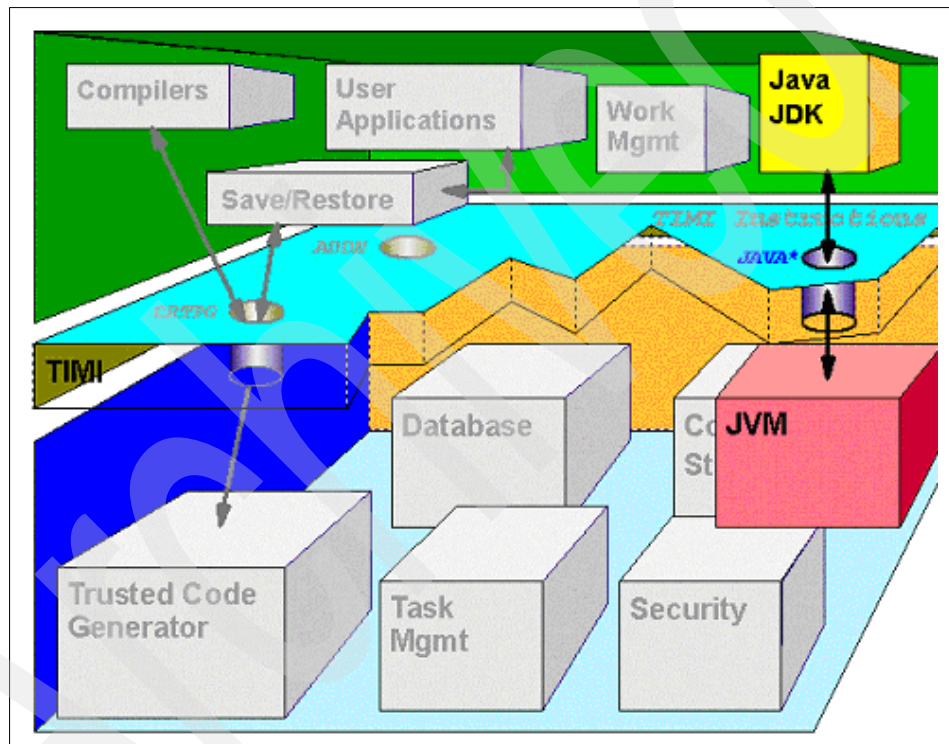Figure 5-1 shows how Java is integrated on the iSeries server.

Figure 5-1   Java integration on the iSeries server

5.2.2 Java transformer

The iSeries implementation of Java provides a unique component called the *Java transformer*. The Java transformer preprocesses Java bytecode that is produced by any Java compiler on any platform and is contained in a class file, JAR file, or ZIP file to prepare them to run using the OS/400 JVM.

The Java transformer creates an optimized Java program object that is persistent and associated with the class file, JAR file, or ZIP file. This program object contains RISC PowerPC 64-bit machine instructions. The optimized program object is not interpreted by the bytecode interpreter at runtime, but directly runs when the class is loaded.

No action is required to start the Java transformer. It automatically starts the first time that a Java class file is run on the system when you use the `java` command from the Qshell Interpreter or the RUNJVA command or JAVA command on iSeries. It is especially important to use the CRTJVAPGM command on JAR files and ZIP files. Unless the entire JAR file or ZIP file has been optimized using the CRTJVAPGM command, each individual class is optimized at runtime and the resulting program objects are temporary.

This boosts performance on most applications that have class files that change infrequently, as Movex Java does.

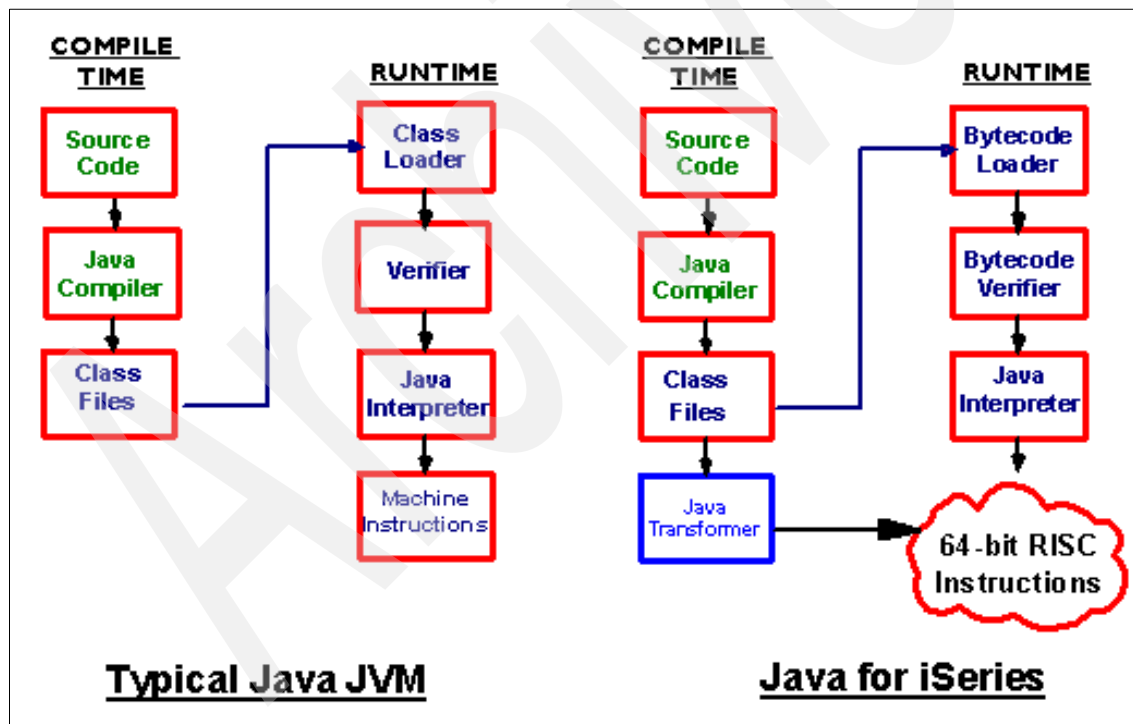Figure 5-2 compares typical Java JVM and Java for iSeries.

Figure 5-2   Typical Java JVM versus Java for iSeries

5.2.3 Java garbage collector

Java provides a fully automated memory allocation and de-allocation known as *automatic garbage collection*. Unlike most other JVM implementations, the OS/400 automatic garbage collection is not "stop and copy", where other threads stop while garbage collection runs. The OS/400 JVM usually performs garbage collection asynchronously without having to stop the other threads.

There are two threads associated with garbage collection. They have the next higher thread numbers after the initial thread with a gap of one hexadecimal increment. The threads and their functions are:

► **Collector garbage collection thread**: Locates objects that are no longer being referenced.

► **Finalize garbage collection thread**: When an object is collected by the garbage collection, this garbage collection thread invokes the FINALIZE method.

Figure 5-3 compares synchronous garbage collection with asynchronous garbage collection.
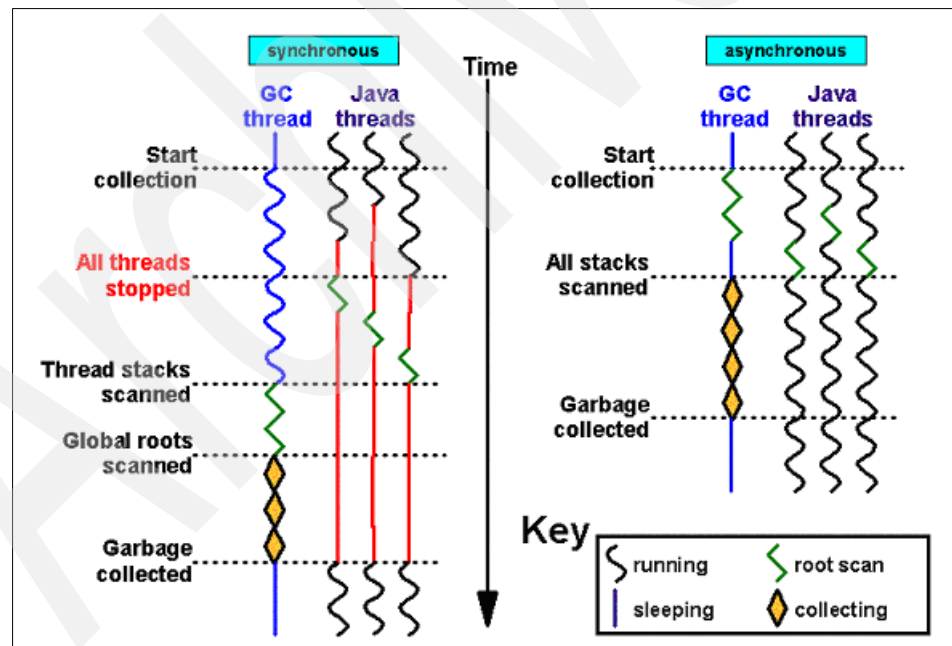

Figure 5-3   Synchronous versus asynchronous garbage collection

The easiest way to monitor the heap size is to specify *VERBOSEGC in the RUNJVA command during startup. This parameter takes effect after a restart of the JVM. The heapsize information output from the JVM is placed in the QSYSPRT printfile for the MVXJVA job.

Another way to monitor the heap size is through the Server View tool. Click the **Counters** link and you see both Heap Total and Heap Free values. You can learn more about the Server View tool in 7.3, "Server View" on page 103.

5.2.4 Java APIs and iSeries

The Java Core Library APIs, as defined by Sun Microsystems, are packaged as a separate, no charge, LPP (5722-JV1). They include:

► The *java.io APIs* are linked to the integrated file system support of OS/400 and provide access to any UNIX or PC style stream file within the integrated file system.

► The *java.net APIs* use the standard TCP/IP support that is provided by the TCP/IP Connectivity Utilities for iSeries (5722-TC1) LPP product.

► The *java.sql APIs* use the standard Structured Query Language (SQL) Call Level Interface (CLI) of OS/400 to access the iSeries database.

Java Utilities and iSeries

Most Java Utilities are supported on the iSeries server. They are run from within the Qshell Interpreter. The Qshell Interpreter is an option of OS/400 (5722-SS1 option 30) that you must install on your iSeries server to run any Java utility on your system.

5.3 iSeries-specific implementation

This section explains some aspects of the Java implementation that are specific to the iSeries, such as:

► OS/400 commands that help you perform Java-related functions in a standard iSeries way with command prompting and online help text

► Qshell Interpreter environment

► Remote AWT support

5.3.1 OS/400 Java commands

You can use OS/400 commands to perform certain Java-related functions. Some commands, such as the Run Java (RUNJVA) command or JAVA command, are OS/400 equivalents to existing Java Utilities, such as the `java` command. These commands are specific to the iSeries implementation:

► Create Java Program (CRTJVAPGM) command
► Change Java Program (CHGJVAPGM) command
► Delete Java Program (DLTJVAPGM) command
► Display Java Program (DSPJVAPGM) command
► Dump Java Virtual Machine (DMPJVM) command

The CRTJVAPGM command

By using the CRTJVAPGM command on a JAR file or ZIP file, it causes all of the classes that are contained in that file to be optimized. Then the resulting optimized Java program object becomes persistent. This results in a much better runtime performance.

You also specify the optimization level of the resulting iSeries Java program. For OPTIMIZE(*INTERPRET), the resulting Java program interprets the class file bytecode when it starts. If the class file has not been run through the CRTJVAPGM command, a persistent service program is created internally the first time the class is run. It then runs subsequently under the OPTIMIZE (10) level by default (V5R1M0). This program object is persistent until the class is deleted or the Delete Java Program (DLTJVAPGM) command is run on it.

The possible values for the OPTIMIZE parameter are:

► ***INTERPRET**: The Java programs that you create are not optimized. When you start the program, it interprets the class file bytecode. You can display and change variables while debugging.

► **10**: The Java program contains a transformed version of the class file bytecode, but has only minimal additional compiler optimization. You can display and change variables while debugging. This is the default value for the OPTIMIZE parameter.

► **20**: The Java program contains a compiled version of the class file bytecode and performs additional compiler optimization. You can display variables, but not change them while debugging.

► **30**: The Java program contains a compiled version of the class file bytecode and has more compiler optimization than optimization level 20. You can display, but not change variables while debugging. The values that are presented may not be the current value of the variable.

► **40**: The Java program contains a compiled version of the class file bytecode and performs more compiler optimization than optimization level 30. All call and instruction tracing is disabled.

> **Note:** If your Java program fails to optimize or throws an exception at optimization level 40, use optimization level 30. The REPLACE parameter allows you to specify whether an existing iSeries Java program should be replaced.

The ENBPFRCOL parameter allows you to specify whether performance data should be collected. Make sure you choose the proper value if you want to analyze the performance of your Java application. The default value of *NONE disables performance data collection for that class or set of classes. The possible values for the ENBPFRCOL parameter are:

► ***NONE**: The collection of performance data is not enabled. No performance data is to be collected. This is the default for the ENBPFRCOL parameter.

► ***ENTRYEXIT**: Performance data is collected for procedure entry and exit.

► ***FULL**: Performance data is collected for procedure entry and exit. Performance data is also collected before and after calls to external procedures.

The CHGJVAPGM command

The CHGJVAPGM command changes the attributes of a Java program, which is attached to either a Java class file or a set of Java programs that are attached to a JAR file. A file is a JAR file if the file name ends in .jar or .zip.

The CHGJVAPGM command uses these parameters:

► CLSF parameter
► OPTIMIZE parameter
► ENBPFRCOL parameter
► LICOPT parameter

You can use these parameters just as you would with the CRTJVAPGM command. See "The CRTJVAPGM command" on page 47 for details about each of these. However, when you use the CHGJVAPGM command, *SAME (the value does not change) is the default for the OPTIMIZE parameter, ENBPFRCOL parameter, and LICOPT parameter. In addition, the MERGE parameter specifies whether you merge Java programs, which are attached to a JAR file, into the minimum number of Java programs possible. This parameter is ignored if you are processing a class file.

Note these options:

- ► ***RPL:** Specifies that you merge Java programs, which are attached to a JAR file, only if the Java programs need to be recreated and replaced, because other Java program attributes are being changed. If no attributes are changed and no Java programs need to be recreated and changed, merging Java programs does not occur.

- ► ***YES:** You merge all Java programs, which are attached to a JAR file, into the minimum number of Java programs possible to save space or improve class loader time.

DLTJVAPGM command

The DLTJVAPGM command deletes an iSeries Java program that is associated with a Java class file, JAR file, or ZIP file. If no Java program is associated with the class file that is specified, an informational message (JVAB526) is sent and command processing continues.

DSPJVAPGM command

The DSPJVAPGM command displays information about the iSeries Java program that is associated with a Java class file. If no Java program is associated with the class file that is specified, an error message is sent and the command is cancelled.

The OUTPUT parameter allows you to specify to where the output should be directed. Specify ***** to display the results and *PRINT to send the results to a spooled file. You can specify the name of a class file, JAR file, or ZIP file.

The DMPJVM command

The DMPJVM command dumps information about the JVM for a specified job. The information is dumped using the QSYSPRT printer file. The dump includes formatted information about the class path, garbage collection, and threads associated with the JVM. The DMPJVM command, in OS/400 V5R1, also lists the objects that are currently active within the JVM heap. This is especially important for Intentia consultants to diagnose problems on site.

The RUNJVA (or JAVA) command

The RUNJVA or JAVA command runs the iSeries Java program that is associated with the Java class that is specified. If no *JVAPGM object is associated with the class file, one is created and associated permanently with the class file. It is used for running the class file, rather than for interpreting the bytecode.

If you specify the special value *VERSION on the CLASS parameter instead of a valid Java class name, the build version information for the JDK and the JVM is displayed. No Java program is run.

You can specify these parameters on the RUNJVA (or JAVA) command:

► **PARM**: Specifies one or more parameter values that are passed to the Java program. You can pass a maximum of 200 parameter values.

► **CLASSPATH**: Specifies the path that is used to locate classes. Directories are separated by colons. If the special value *ENVVAR is used, the class path is determined by the environment variable CLASSPATH. You can set the CLASSPATH environment variable by using the Add Environment Variable (ADDENVVAR) command, be part of an export directive in the system wide /etc./profile file, or specify at the user profile level with an export directive that is contained in the profile file in the home directory of each user.

► **CHKPATH**: Specifies the level of warnings given for directories in the class path that have public write authority. A directory in the class path that has public write authority is a security exposure, because it may contain a class file with the same name as the one you want to run. The first class file found is the first class file that is run. The possible values for this parameter are:

  – *WARN: A warning message is sent for each directory in the class path that has public write authority. This is the default value.

  – *SECURE: A warning message is sent for each directory in the class path that has public write authority. If one or more warning messages are sent, an escape message is sent, and the Java program does not run.

  – *IGNORE: Ignores the fact that directories in the class path may have public write authority. No warning messages are sent.

► **OPTIMIZE**: Specifies the optimization level of the iSeries Java program that is created if no Java program is associated with the Java class file. The created Java program remains associated with the class file after the Java program is run. You can disable optimization by specifying OPTIMIZE(*INTERPRET) on the RUNJVA (or JAVA) command. This requires that the classes are interpreted regardless of the optimization level you set in the associated Java program object. This is useful if you want to debug a class that was optimized with an optimization level of 30 or 40.

► **PROP**: Specifies a list of values to assign to Java properties. Up to 100 Java properties can have a value assigned.

► **GCHINL**: Specifies the initial size (in kilobytes) of the garbage collection heap. This is used to prevent garbage collection from starting on small programs.

► **GCHMAX**: Specifies the maximum size (in kilobytes) to which the garbage collection heap can grow. This prevents runaway programs that consume all of the available storage. Normally, garbage collection runs as an asynchronous thread in parallel with other threads. If the maximum size is reached, all other threads are stopped while garbage collection takes place.

- ► **GCFRQ**: This parameter is no longer supported. It exists solely for compatibility with releases earlier than V4R3 of the iSeries.

- ► **GCPTY**: This parameter is no longer supported. It exists solely for compatibility with releases earlier than V4R3 of the iSeries.

- ► **OPTION**: Specifies special options that you can use when running the Java class. The possible values are:

  - – *NONE: No special options are used when running the Java class.

  - – *DEBUG: Allows the iSeries debugger to be used for this Java program.

  - – *VERBOSE: A message is displayed each time a class file is loaded.

  - – *VERBOSEGC: A message is displayed for each garbage collection sweep.

  - – *NOCLASSGC: Unused classes are not reclaimed when garbage collection runs.

6

Installing Movex Java

This chapter presents an overview over the installation of Movex Java on the iSeries platform.

Note: The hardware and software specifications provided in this chapter are included for reference purposes only. They are not valid as a certified solution. For the appropriate and current specifications, refer to the Intentia Wire.

53

6.1 Platform

The Movex Certified Configuration Platform is based on a completely validated Movex solution that consists of the Movex software, hardware (client, application, and database server), and bundled software.

The Movex Java-certified configuration is provided through a dedicated engineering team, under the wings of the Intentia IBM International Competence Center. It performs ongoing compatibility testing of the components included, installation validations, performance measurement, and optimization.

The goal is to simplify and package the technical planning required for implementing Movex and to quickly deliver an efficient Movex implementation. In this context, it is very important to highlight that certified configuration solutions have been validated from a functional and performance point of view. This also means that when a combination ends up on the list of certified configurations, the given characteristics fit Movex as well as the underlying technologies.

6.1.1 Hardware

The current Movex Certified Configurations include:

► IBM @server iSeries server
► IBM @server xSeries server
► Sun Enterprise Servers

For detailed information about models and sample configurations for small (less than 50 users), medium (between 50 and 150 users), and large (greater than 150 users) installations, refer to the Intentia Wire. On the Wire, look under **Intentia Solution-> Technology-> Hardware**.

6.1.2 Software

To install and run Movex Java on the iSeries platform, a number of pre-requisites must be met. The following sections present a brief overview over the different software components that must be installed.

For specific software information, see the Intentia Wire.

Licensed programs

The software needed to run Movex Java on an iSeries is divided in different categories. The following software products are needed as a minimum:

Licensed program	Version	Description
5722-SS1	V5R1	Operating System/400 (Charge)
5722-SS1 Opt 12	V5R1	OS/400 Host Servers (Free)
5722-SS1 Opt 30	V5R1	OS/400 Qshell Interpreter (Free)
5722-JC1	V5R1	AS/400 Toolbox for Java (Free)
5722-JV1	V5R1	AS/400 Developer Kit for Java (Free)
5722-JV1 Opt 5	V5R1	Java Developer Kit 1.3 (Free)
5722-TC1	V5R1	TCP/IP Connectivity Utilities (Free)

The following software products are optional, but strongly recommended:

Licensed program	Version	Description
5722-WSD	V5R1	WebSphere Development Tools (Charge) (*BASE and Option 21)
5722-JV1 Opt 4	V5R1	Java Developer Kit 1.1.8 (Free)
5722-XE1	V5R1	Client Access Express for Windows (Charge)
5722-XW1	V5R1	Client Access Windows Family Base (Charge)

The following software products are optional, but not required:

Licensed program	Version	Description
5722-ST1	V5R1	DB2 Query Manager and SQL Developers Kit for AS/400 (Charge)
5722-PT1	V5R1	Performance Tools for AS/400 (Charge)

Program temporary fixes (PTFs)

In addition to the software products listed in the previous section, PTFs may need to be applied to the installation. To find current information about which PTFs need to be applied, refer to the Intentia Wire.

6.2 Installation prerequisites

The following prerequisites must be met:

► All necessary components of the operating system must be installed.

► All necessary PTFs must be applied.

► The system must have TCP/IP configured and be reachable from the network.

- The / (root) file system must be published to the Windows Network Neighborhood or at least be accessible from the Windows environment.

- For a new installation, the user QSECOFR must be available for the initial steps.

- For any upgrade, a proper backup must be taken of both the database and the application before the installation or upgrade work begins.

- At least one client should be available with Operations Navigator and a 5250 Emulation session should be installed and configured.

- Movex Explorer and Movex OUT should be installed before or during the installation described in this redbook. (This redbook does not cover those installations.)

6.3 Installation concepts

The following section offers a brief overview over the physical distribution that will be used for the installation on an iSeries platform. For more information, refer to the installation instructions on the Intentia Wire.

6.3.1 iSeries distribution

A normal delivery, distribution targeted for the iSeries platform consists of at least one tape and one CD. The reason for this is that tape is better suited to carry the amount of data needed for a Movex Java installation.

In theory, the same build of the application could be used over all available platforms. However, when it comes to the iSeries server, the creation of Java programs is done better in one central location than at each installation. Because of this, the distribution grows from approximately 300 MB to around 5 GB and is easier to place on a tape. This also makes it possible to prepare this distribution for the target platform, the iSeries server.

The tape contains some or all of the following components:

Component	Save command used
QINSTAPP	SAVOBJ
MVXJJVA	SAVLIB
Movex_v12	SAV
Movex_v12 /HUM	SAV
MVXJDTAMST	SAVLIB

QINSTAPP is the installation program that enables the use of the operating system function LODRUN to install from the tape. This program performs the first steps of the installation. It sets up users and groups, sets some of the system values, and finally installs the installation utility library MVXCJVA. This library contains all other functions necessary to install, operate, and maintain Movex Java on the iSeries platform.

By the help of functions, found in this utility library, the file structure constituting Movex Java is installed in the IFS of the iSeries machine. On the tape, this part is found in the component named Movex_v12. If the delivery contains Movex Human Resources (HUM), that component is placed next on the tape, and the Demo database (MVXJDTAMST) is placed last, if included.

Note that the delivery database is not placed on the tape as a library, but is included in the IFS file structure from where it is installed by the help of functions in the utility library.

The following components are placed on the accompanying CD:

Component	Description
MEX_12rs	Movex Explorer content
MOM_12rs	Movex Out content

Both Movex Explorer and Movex Out are supposed to be installed on a Windows server in the local network.

6.3.2 Pre-creation of Java programs

The ordinary distribution for the iSeries server contains pre-created Java programs, as mentioned previously. These programs are created with the optimization level set to 40 and do not have performance collection enabled. This means that the programs could not be used for debugging purposes.

On the other hand, platform-specific debugging is seldom needed on site, and therefore the debug-enabled version of the programs is available only through the Intentia Modification Center (IMC). In a normal debug scenario, an IMC representative connects to the installation database from a separate application server.

6.4 Installation workflow

This section provides an overview of the installation workflow. You can find the complete installation instructions on the Intentia Wire.

To ease installation as much as possible, LODRUN, which is a part of the iSeries server, is used. By simply issuing one single command, you install and execute the installation program. The installation program then takes you through a guided tour where necessary user profiles and groups are created and where some essential system values and other settings are presented.

> **Note:** The information in the following sections is always overridden by information provided for a specific version or release. Refer to the Intentia Wire for the latest instructions. On the Wire, look under **Deliverables-> Installation-> Movex Java**.

6.4.1 Base installation

Use this flow to install Movex Java version 12 on an iSeries server on which Movex Java was not previously installed. Complete these steps:

1. Mount the tape in the proper tape or optical station.

2. Sign on as QSECOFR and enter the LODRUN command. If there are several tape stations, prompt the command so you can give the right device name:

   a. LODRUN creates the proper users and groups that are needed for Movex Java.

   b. LODRUN goes through a number of system values and setup issues to enable the environment for Movex Java.

   c. LODRUN finally installs the Java Utility library MVXCJVA and displays the main menu, MVXSTART, in the library.

3. Sign off as QSECOFR and sign on as MSRVADM.

4. When prompted, change the password for the user MSRVADM.

5. Enter GO MVXAPP and press Enter.

6. Create the installation directory.

7. Install the Movex Java version. Wait until this installation is finished before you proceed with the next step.

8. Install the Movex database.

9. If a demo database is desired, install that now by using the Restore Library (RSTLIB) command.

10. Set authority for the databases.

11. Create a journal environment.

12. Start journaling.

13. Install any required service packs.

14. Install any HUM.

15. Install any service packs for HUM.

16. Install any market modifications.

17. Install any service packs for market modifications.

18. Install any additional languages.

19. Install any customer modifications.

20. Install any service packs for customer modifications.

21. Update Movex Explorer with language, menus, help, and view definitions. You can find these on the CD that is a part of the delivery.

22. Update Movex OUT with language and OUT definitions. You can find these on the CD that is a part of the delivery.

23. Set authority for the application.

24. Create the proper environments.

25. Update Movex.properties.

26. Create the runtime environment.

27. Update the start program.

28. Test the installation.

6.4.2 Upgrade installation

Use this procedure to install Movex Java version 12 on an iSeries machine on which a previous version of Movex Java was already installed.

1. Sign on as `QSECOFR`.

2. Stop the running version of Movex.

3. Check the service pack level of the database and upgrade the database if necessary.

4. Rename the existing Utility library `MVXCJVA`.

5. Mount the tape containing Movex Java Version 12 in the proper tape station.

6. Enter the `LODRUN` command. If there are several tape stations, make sure to prompt the command to give the right device name. Use QSECOFR rather than MSRVADM to allow eventual changes to that user profile to take place.

   a. LODRUN checks and changes the users and groups that are needed for the Movex Java version.

   b. LODRUN goes through a number of system values and setup issues to enable the environment for the Movex Java version.

    c. LODRUN installs the Java Utility library MVXCJVA and displays the main menu, MVXSTART, in the library.

7. Sign off as QSECOFR and sign on as MSRVADM.

8. Enter GO MVXAPP and press Enter.

9. Create the installation directory.

10. Install the Movex Java version. Wait until this installation is finished before you proceed to the next step.

11. Manage the database upgrade according to the database upgrade description found in the folder databases\db2_400 in the file system.

12. Set authority for the databases.

13. Create a journal environment.

14. Start journaling.

15. Install any required service packs.

16. Install any HUM.

17. Install any service packs for HUM.

18. Install any market modifications.

19. Install any service packs for market modifications.

20. Install any additional languages.

21. Install any customer modifications.

22. Install any service packs for customer modifications.

23. Update Movex Explorer with language, menus, help, and view definitions. You can find these on the CD that is a part of the delivery.

24. Update Movex Out with language and out definitions. You can find these on the CD that is a part of the delivery.

25. Set authority for application.

26. Create the proper environments

27. Update Movex.properties.

28. Create the runtime environment.

29. Update the start program.

30. Test the installation.

If the upgrade is done to set up a test environment, while the production environment is running the previous version, make sure that the production environment can be started in parallel to the new environment.

6.4.3 Installing a service pack

Follow these instructions to install a service pack to the Standard Movex Java version:

1. Download the service pack from the Intentia Wire.

2. Extract the service pack to the hard disk of the Administrative client.

3. Read the service pack information text file in the service pack folder.

4. Copy the service pack folder into the root folder of the Movex installation using Windows Explorer.

5. Create the Java programs.

6. Create the common.jar file, if it is included in the service pack.

7. Upgrade the database, if the service pack contains such components.

8. Install upgraded database drivers or upgrades of the utilities, if the service pack contains such components.

9. Activate the database drivers.

10. Upgrade Movex Explorer, if the service pack contains such components.

11. Upgrade Movex OUT, if the service pack contains such components.

> **Note:** If this service pack installation is a part of a new installation or an upgrade, you may return to the instructions now. Otherwise continue with the following steps.

12. Update Movex.properties with new View definition path.

13. Update the start program.

14. Test the installation.

6.5 Movex Java application users and user groups

During the installation, the following users and groups are created. It is very important that they exist and are set up correctly to have functional access control in place for the file system. If the user profiles already exist, meaning that you see a Change User Profile rather then a Create User Profile, press Enter to make any updates active.

The following users and groups are created:

User/group	Description
MSRVADMS	MOVEX - Server Administrators Group
MSRVADM	MOVEX - Server Administrator
MDBUSR	MOVEX - Database User
MDBREADS	MOVEX - Database Access Group

For a more thorough description of the users and groups, refer to Chapter 9, "Security" on page 117.

6.6 OS/400 system values and other settings for Movex Java

Movex Java is a highly sophisticated technical application that uses OS/400 security and a runtime system. This means that to have an optimal environment, the settings in Movex (Movex.properties) must be synchronized with user profiles and OS/400 system values.

This section describes the basic constructions and settings that are managed by the Movex installation routines. The installation of Movex Java (LODRUN) takes you through a number of system values and other settings that need to be checked and, in some cases, changed. These changes are essential for the function of the Movex Java version on OS/400.

> **Attention:** The affect of some of these changes may conflict with other applications running in parallel on the system. Be sure to watch for such conflicts.

Recovery information

If you accidentally leave a step without making the proper changes, you can use option 30 "Check User profiles". This options is located on the MVXTOOLS menu in the utility library. This option allows you to go through the steps again after the first time.

6.6.1 Work with Relation Database Directory Entries (WRKRDBDIRE)

The WRKRDBDIRE command allows you to note the value that already exists for the *LOCAL entry or to create a new entry if it does not already exist. Make a note of this value for later use in the Movex.properties file. An example of the WRKRDBDIRE command display is shown in Figure 6-1.

```
            Work with Relational Database Directory Entries

 Position to  . . . . . .

 Type options, press Enter.
   1=Add   2=Change   4=Remove   5=Display details   6=Print details

          Relational           Remote
 Option   Database             Location                    Text

          SEIRDPAC             *LOCAL




 Bottom
  F3=Exit   F5=Refresh   F6=Print list   F12=Cancel
  (C) COPYRIGHT IBM CORP. 1980, 1999.
```

Figure 6-1 The WRKRDBDIRE command display

A proper entry should look like the example shown on Figure 6-2. Note that the Type field should be left blank. For the system name, use the system name as the name of the relational database, if nothing else is decided.

```
                    Change RDB Directory Entry (CHGRDBDIRE)

 Type choices, press Enter.

 Relational database  . . . . . . > SEIRDPAC      Character value
 Remote location:
   Name or address  . . . . . . .   *LOCAL




   Type . . . . . . . . . . . . .                  *SNA, *IP, *SAME
 Text . . . . . . . . . . . . . .   *BLANK







 Bottom
  F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
  F24=More keys
```

Figure 6-2   The CHGRDBDIRE command display

6.6.2 Coded character set identifier (WRKSYSVAL QCCSID)

To check or change the system value QCCSID, a WRKSYSVAL QCCSID command is issued. On the Work with System Values display (Figure 6-3), check to see that the system value is not 65535. If it is, you have to consider a change. On new iSeries server, change it to the CCSID of most of the users of the system as shown on Figure 6-4.

If it is an older iSeries model that is already running applications and databases, leave the value as is. In this case, after the first installation steps, set a CCSID on the user profiles of MSRVADM and MDBUSR. This is not a good solution. You should consider changing the system value, but only with careful planning first.

```
                        Work with System Values
 System:   SEIRDPAC
  Position to  . . . . . . Starting characters of system value
  Subset by Type . . . . .            F4 for list


  Type options, press Enter.
    2=Change   5=Display


          System
 Option  Value      Type     Description
         QCCSID     *SYSCTL  Coded character set identifier










 Bottom
  Command
  ===>
  F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F11=Display names only
  F12=Cancel
```

Figure 6-3   The WRKSYSVAL QCCSID command display

```
                        Change System Value

 System value . . . . . :   QCCSID
 Description  . . . . . :   Coded character set identifier


 Type choice, press Enter.

   Coded character set
     identifier . . . . .   37         1-65535









 F3=Exit   F5=Refresh   F12=Cancel
```

Figure 6-4   The CHGSYSVAL QCCSID command display

6.6.3 Job message queue full action (WRKSYSVAL QJOBMSGQFL)

To avoid problems with canceled jobs during installation, or other long running tasks, set the system value for QJOBMSGQFL by using the WRKSYSVAL command. Figure 6-5 shows an example of the Work with System Values display.

```
                     Work with System Values
 System:   SEIRDPAC
 Position to . . . . . .                Starting characters of system value
 Subset by Type . . . . .                F4 for list

 Type options, press Enter.
   2=Change   5=Display

        System
 Option  Value      Type      Description
         QJOBMSGQFL *ALC      Job message queue full action




Bottom
 Command
 ===>
 F3=Exit  F4=Prompt  F5=Refresh  F9=Retrieve  F11=Display names only
 F12=Cancel
```

Figure 6-5   The WRKSYSVAL QJOBMSGQFL command display

Select option 2 (change) and you see the Change System Value display as shown in Figure 6-6. Change the value to something other than *NOWRAP, which is the shipped default value from IBM. Consider using the *WRAP or *PRTWRAP option.

```
                        Change System Value

 System value . . . . . :   QJOBMSGQFL
 Description  . . . . . :   Job message queue full action


 Type choice, press Enter.

   Job message queue full
     action . . . . . . .    *WRAP              *NOWRAP
                                                *WRAP
                                                *PRTWRAP












 F3=Exit   F5=Refresh   F12=Cancel
```

Figure 6-6   The CHGSYSVAL QJOBMSGQFL command display

6.6.4 Performance adjustment (WRKSYSVAL QPFRADJ)

To make things easier, set the performance adjustment to be automatic. On the
Work with System Values display shown in Figure 6-7, select option 2 to change
the value.

```
                        Work with System Values
 System:   SEIRDPAC
  Position to  . . . . . .                   Starting characters of system value
  Subset by Type . . . . .                    F4 for list

  Type options, press Enter.
    2=Change   5=Display

         System
 Option  Value       Type      Description
         QPFRADJ     *SYSCTL   Performance adjustment







 Bottom
  Command
  ===>
  F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F11=Display names only
  F12=Cancel
```

Figure 6-7   The WRKSYSVAL QPFRADJ command display

Then, on the Change System Value display (Figure 6-8), select option 2 for Performance adjustment. This allows for adjustment at IPL and automatic adjustment.

```
                        Change System Value

 System value . . . . . :   QPFRADJ
 Description  . . . . . :   Performance adjustment


 Type choice, press Enter.

   Performance adjustment . . . .   2          0=No adjustment
                                               1=Adjustment at IPL
                                               2=Adjustment at IPL and
                                                  automatic adjustment
                                               3=Automatic adjustment










 F3=Exit   F5=Refresh   F12=Cancel
```

Figure 6-8   The CHGSYSVAL QPFRADJ command display

6.6.5 Changing printer file definition for QPRINT (CHGPRTF QPRINT)

The Max spooled output records parameter, for the system printer file QPRINT, must be changed to *NOMAX. This helps to avoid hang situations where the application server fills the log with records.

If you receive the error message CPF7304, type I for "Ignore". Make a note that you have to perform this change later. This error message indicates that the printer file is locked by another process and cannot be changed at this time. Make an appropriate change as illustrated on Figure 6-9 and Figure 6-10.

```
                          Change Printer File (CHGPRTF)

 Type choices, press Enter.

 File . . . . . . . . . . . . . . > QPRINT        Name, generic*, *ALL
   Library  . . . . . . . . . . .    *LIBL        Name, *LIBL, *ALL, *ALLUSR...
 Device:
   Printer  . . . . . . . . . . .  *JOB           Name, *SAME, *JOB, *SYSVAL
 Printer device type  . . . . . .  *SCS           *SAME, *SCS, *IPDS, *LINE...
 Page size:
   Length--lines per page . . . .  66             .001-255.000, *SAME
   Width--positions per line  . .  132            .001-378.000, *SAME
   Measurement method . . . . . .  *ROWCOL        *SAME, *ROWCOL, *UOM
 Lines per inch . . . . . . . . .  6              *SAME, 6, 3, 4, 7.5, 7,5...
 Characters per inch  . . . . . .  10             *SAME, 10, 5, 12, 13.3, 13...
 Overflow line number . . . . . .  60             1-255, *SAME
 Record format level check  . . .  *NO            *SAME, *YES, *NO
 Text 'description' . . . . . . .  'Default spool output print file for OUTQ(*
 JOB)'


 More...
  F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
  F13=How to use this display        F24=More keys
```

Figure 6-9   The CHGPRTF QPRINT command display (Part 1 of 2)

Press Page Down to reach the page with the suggested change.

```
                        Change Printer File (CHGPRTF)

 Type choices, press Enter.


                           Additional Parameters

 Max spooled output records . . . > *NOMAX        1-999999, *SAME, *NOMAX











 Bottom
  F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
  F13=How to use this display        F24=More keys
```

Figure 6-10   The CHGPRTF QPRINT command display (Part 2 of 2)

6.6.6 Changing prestart job entry for QSQSRVR (CHGPJE QSQSRVR)

To make sure that the system job for internal SQL transactions to the database starts, set the Start jobs parameter to *YES on the Change Prestart Job Entry display (Figure 6-11). This starts the job automatically when the subsystem starts.

A change can also turn on this job directly. This change is important even if you plan to use record level access (RLA) as the database access method in the installation. SQL is still used for some kind of functionality that involves dynamic retrieval of data from the database.

```
                  Change Prestart Job Entry (CHGPJE)

 Type choices, press Enter.

 Subsystem description  . . . . . > QSYSWRK       Name
   Library  . . . . . . . . . .     *LIBL        Name, *LIBL, *CURLIB
 Program  . . . . . . . . . . . > QSQSRVR        Name
   Library  . . . . . . . . . .     *LIBL        Name, *LIBL, *CURLIB
 User profile . . . . . . . . .    *SAME         Name, *SAME
 Start jobs . . . . . . . . . . > *YES           *SAME, *YES, *NO
 Initial number of jobs . . . .    *SAME         1-9999, *SAME
 Threshold  . . . . . . . . . .    *SAME         1-9999, *SAME
 Additional number of jobs  . . .  *SAME         0-999, *SAME
 Maximum number of jobs . . . . .  *SAME         1-9999, *SAME, *NOMAX




 Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
 F13=How to use this display       F24=More keys
```

Figure 6-11   The CHGPJE QSQSRVR command display

6.6.7 Changing shared storage for default Movex pool

The runtime environment for Movex uses SHRPOOL5 as the default memory pool. To make this active, define the shared pool. Assign the shared pool a starting value and make it possible for the automatic performance adjustment to do the rest the suggested values. If you did not switch on the automatic adjustment, you have to set better values here.

On the Change Shared Storage Pool display (Figure 6-12), set the Maximum size % parameter to a value that is less then 100. This value tells how much of the main storage may be allocated to this pool.

```
                 Change Shared Storage Pool (CHGSHRPOOL)

 Type choices, press Enter.

 Pool identifier  . . . . . . . . > *SHRPOOL5     *MACHINE, *BASE,
*INTERACT...
 Storage size . . . . . . . . . . > 256          Number, *SAME, *NOSTG
 Activity level . . . . . . . . . > 1            Number, *SAME
 Paging option  . . . . . . . . . > *CALC         *SAME, *FIXED, *CALC
 Text 'description' . . . . . . . > 'Movex Java'



                         Additional Parameters


 Maximum size % . . . . . . . . . > 80            0-100, *SAME, *DFT





 Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
 F13=How to use this display        F24=More keys
```

Figure 6-12   The CHGSHRPOOL command display

6.6.8 Configuring TPC/IP (GO CFGTCP)

The TCP/IP configuration of the system must be correct to get Movex up and running. You can do this by selecting option 12 (Change TCP/IP domain information) on the CFGTCP menu (Figure 6-13).

```
CFGTCP                      Configure TCP/IP
                                                    System:    SEIRDPAC
 Select one of the following:

      1. Work with TCP/IP interfaces
      2. Work with TCP/IP routes
      3. Change TCP/IP attributes
      4. Work with TCP/IP port restrictions
      5. Work with TCP/IP remote system information

     10. Work with TCP/IP host table entries
     11. Merge TCP/IP host table
     12. Change TCP/IP domain information

     20. Configure TCP/IP applications
     21. Configure related tables
     22. Configure point-to-point TCP/IP



 Selection or command
 ===>


 F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

Figure 6-13   The Configure TCP/IP menu

Then you see the Host name and Domain name parameters as illustrated on Figure 6-14. Verify that the correct names are there.

```
                      Change TCP/IP Domain (CHGTCPDMN)

 Type choices, press Enter.

 Host name  . . . . . . . . . . .    'SEIRDPAC'

 Domain name  . . . . . . . . . .    'IRD.INTENTIA.SE'



 Host name search priority  . . .    *REMOTE       *REMOTE, *LOCAL, *SAME
 Domain name server:
   Internet address . . . . . . .    '10.20.20.247'
                                     '10.20.15.251'




                                                                     Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
 F13=How to use this display       F24=More keys
```

Figure 6-14   The CHGTCPDMN command display

Then go back to the command line and ping the machine using both the host name and the domain name together. In this example, you would issue the command:

```
PING SEIRDPAC.IRD.INTENTIA.SE
```

If you receive an answer that the ping is verified, the configuration is OK. In this example, a remote name server is used to resolve the name.

If the ping test does not produce a good result, you may need to change the Host name search priority parameter to *LOCAL. In this case, use option 10 (Work with TCP/IP host table entries) on the CFGTCP menu to set the proper settings as shown on Figure 6-15.

```
                     Work with TCP/IP Host Table Entries
                                                         System:    SEIRDPAC
 Type options, press Enter.
   1=Add   2=Change   4=Remove   5=Display   7=Rename


      Internet         Host
 Opt  Address          Name

      10.20.20.236      SEIRDPAC
                        SEIRDPAC.IRD.INTENTIA.SE
      127.0.0.1         LOOPBACK
                        LOCALHOST









                                                                      Bottom
  F3=Exit   F5=Refresh   F6=Print list   F12=Cancel   F17=Position to
```

Figure 6-15   Work with TCP/IP Host Table Entries display

Make sure that one of the entries contains the host and domain names entered in
the TCP/IP domain information.

6.6.9 Installation utility library (MVXCJVA)

An important part of the installation is to download the tools and utilities that are
provided to support the technical aspects and system maintenance of Movex.
The functions are structured on menus related to application and database tasks.

Refer to the Intentia Wire for instructions. On the Wire, look under
Deliverables-> Installation-> Movex Java.

6.6.10 Installing an application

The Movex – Application menu (MVXAPP), shown in Figure 6-16, lists several functional options. These options allow you to:

- ► Install application class files
- ► Set object authorities
- ► Specify runtime environment settings
- ► Modify Movex.properties
- ► Pre-creating Java service programs (*SRVPGM)

```
MVXAPP                 Movex - Application
                                                        System: SEIRDPAC
 Select one of the following:

    1. Create install directory
   10. Install Movex 12.r.s           11. Install SP to Movex


   20. Set authority for application

   30. Create runtime environment

   40. Select Movex.properties
   41. Edit Movex.properties

   50. Create Java programs           51. Create Java programs for SP
    52. Create Java pgm for common 53. Create Java pgm for common in SP

 Selection or command
 ===>

 F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
 F13=Information Assistant  F16=AS/400 main menu
```

Figure 6-16   The Movex – Application menu

6.6.11 Installing a database

The Movex – Database Setup menu (MVXDB), as shown in Figure 6-17, focuses Movex database tasks on the DB2 UDB for iSeries. The functions include handling the installation of different levels of databases (core and market), service packs, creating logical files, and a number of control functions for the most common known problems.

Journaling management of the Movex databases is available through this menu.

```
MVXDB                    Movex - Database Setup
                                                        System: SEIRDPAC
 Select one of the following:

   10. Install database             11. Install SP to database
   15. Install Market database      16. Install SP to Market database
   20. Display installed SP level   25. Install additional languages

   30. Recreate LFs in Market database  35. Check Db for wrong connections
   36. Check Db file for no of records  40. Set authority for database
   37. Check files with <10% free       38. Reorganize files in database
   50. Install/Update database drivers

   60. Create journal environment
   65. Start journaling
   66. Stop journaling

 Selection or command
 ===>


 F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
 F13=Information Assistant  F16=AS/400 main menu
```

Figure 6-17   The Movex – Database Setup menu

6.6.12 Installing additional languages

Movex core delivery contains U.S. English for all language components. This means that if any additional language components should be used, they must be installed using the Install Additional Languages (JINSLN) command. The Install additional languages command display is shown in Figure 6-18.

```
 JINSLN/E              Install additional languages              5/09/01 15:10:31
                                                                 SEIRDPAC


 Language............:
 Version.............: 12rs

 Path to language....: /Movex_12.r.s/languageS_yy/databaseS/db2_400




 Service pack library: MVXJSPAC    Library name

 Database library....: MVXJDTA     Library name

 Journal library.....: MVXJJRN     Library name, *NONE




 F3=End            F12=Cancel
```

Figure 6-18   The Install additional languages (JINSLN) display

6.6.13 Installing Movex Explorer

One difference from earlier versions of Movex is that the Movex Client application and view definitions, help text, language, and message files use Movex installation as the transport media for delivery. Intentia R&D has also made the application (.exe) release independent. This means that only view definitions (.mvx) need to be updated between releases.

Copy the files in the MEX folder to the files server or clients on which the installation can start. Refer to the Movex Explorer installation instructions on the Intentia Wire.

The Movex Explorer itself is also shipped as an executable file in the MEX folder of the distribution.

6.6.14 Installing Movex OUT

The same kind of management for Movex Explorer is used for the Movex OUT product. The points that were discussed in the previous section for Movex Explorer also apply to Movex OUT.

6.6.15 Setting up Movex.properties

In the Movex.properties files, all runtime settings are stored for a Movex Java installation. To make a comparison this file is as important to Movex Java as the library list is for Movex ThisGen. In other words, it is vital that the settings are correct according to the expected behavior.

The Movex.properties file is a text file located in the runtime environment folder in the IFS file system. You can edit it by using option 41 (Edit Movex.properties) on MVXAPP menu in the Movex utilities tools library. You can also edit it by using an ordinary text editor via Windows Explorer. If you choose this method, note that a time stamped backup copy is not made automatically.

6.6.16 Setting up the start program

When the Movex.properties file is set according to the specific environment, you must modify the start program. To simplify the job, Intentia delivers two templates (one for test and one for production) that you should use to set the library list and to call the Movex start program (/MvxStarter.class) with the parameters for batch, autostart jobs, log characteristics, etc.

You can find the start CL program in the source file STRPGM in the MVXCJVA utility library. Figure 6-19 shows an example of this program.

```
                      Work with Members Using PDM                 SEIRDWS1

 File . . . . . .   STRPGM
   Library . . . .    MVXJJVA              Position to . . . . .

 Type options, press Enter.
  2=Edit         3=Copy  4=Delete 5=Display      6=Print    7=Rename
  8=Display description  9=Save  13=Change text  14=Compile  15=Create
module...

 Opt  Member      Type       Text
      STRMVXTEMP  CLP        Start program for Movex 12.x - Template
      STRMVXTEST  CLP        Start program for Movex 12.x - Test




                                                                  Bottom
 Parameters or command
 ===>
 F3=Exit         F4=Prompt           F5=Refresh          F6=Create
 F9=Retrieve     F10=Command entry   F23=More options    F24=More keys
                                         (C) COPYRIGHT IBM CORP. 1981, 1999.
```

Figure 6-19 Contents of the source file STRPGM

Then you can start Movex Version 12 both in interactive mode and in batch mode
from the Start Movex (STRMVX) display (Figure 6-20). You can add this function
to the normal scheduling facility (WRKJOBSCHE) or other iSeries management.

```
                         Start Movex (STRMVX)

 Type choices, press Enter.

 Environment to start . . . . . .              Test, Prod, ..
 Type of start  . . . . . . . . .   *SBM        *SBM, *INT
 Auto start Batch and ASJ?  . . .   *YES        *YES, *NO









                                                             Bottom
  F3=Exit   F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
  F24=More keys
```
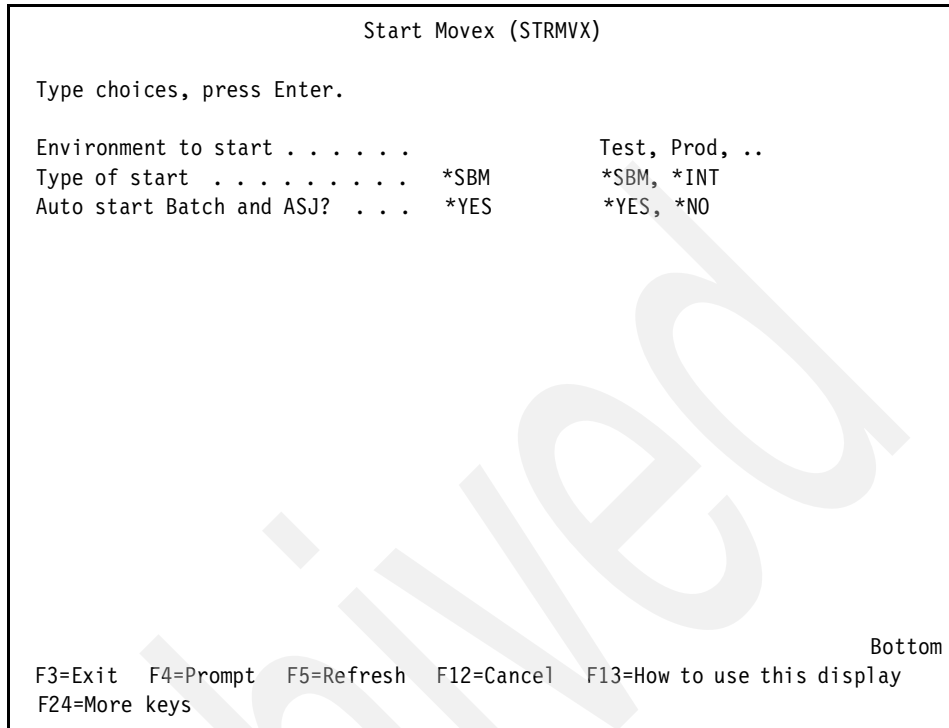
Figure 6-20   The Start Movex (STRMVX) display

6.7 Service packs

Service packs are provided in cases where correction of the software is required.
The delivery strategy is to publish service packs when needed and to make
consolidation levels (also called releases) on a periodic schedule according to
the delivery plan throughout the entire Movex version's life cycle.

The service pack components are placed in folders with names identical to the
service pack themselves. This makes it easy to add and remove specific
corrections seamlessly.

The naming of service packs is in the format *SP12rsSyddnn*, where:

► SP=service pack
► 12=version
► r=release
► s=shipment of the day
► S=Movex Standard

- ► y=year
- ► dd=week
- ► nn=day

Adding service packs is done through the Install application SP (JINSAP) and Install database SP (JINSDB) functions as illustrated on Figure 6-21.

```
 JINSAP/E                Install application SP              6/06/01 21:22:56
                                                             SEIRDPAC


  Install directory...: /Movex_12.r.s/SP12rsSydddn




  Device..............: TAPO1

  Label...............: MVX                    MVX, HUM, *SEARCH

  Run in batch........: *YES




 F3=End              F12=Cancel
```

Figure 6-21   The Install application SP (JINSAP) display

7

Work management

Work management is the method of managing OS/400 system resources to ensure the efficient running of application software such as Movex Java. This chapter looks at some techniques that you can apply to manage the Movex Java workload in an effective manner. You can use these techniques as a framework that is suitable for many installations. However, keep in mind that there is not one definitive solution, since the techniques may differ from installation to installation.

This chapter is intended as a practical guide rather than an in-depth discussion of the benefits of the different options. In many ways, Movex Java work management is similar to running other applications on an OS/400 system.

The OS/400 concepts are not discussed in detail. For further information, refer to *OS/400 Work Management,* SC41-5306.

7.1 OS/400 work management

This section discusses the topics related to work management of Movex Java tasks on the iSeries server. These include such tasks as OS/400 memory management, OS/400 shared pools, IBM-supplied subsystems, Movex Java-supplied subsystems, Movex Java runtime environment, and Java run priorities.

7.1.1 OS/400 memory management

On the iSeries server, all of the primary memory (main storage) can be assigned to logical units called *storage pools*. The iSeries server comes supplied with two pre-defined storage pools:

► The machine pool (*MACHINE)
► The base pool (*BASE)

The machine pool is used by the system, and the base pool contains all other main storage not used by the machine pool or any user-created storage pools. By default, all IBM-supplied subsystems run in the base pool. During the Movex Java installation phase, one additional shared memory pool (*SHRPOOL5) is created. All Movex Java subsystems are routed to this shared pool.

7.1.2 OS/400 shared pools

The OS/400 system has a number of shared pools available (60 excluding the delivered pools such as base and machine pool). All of these pools can be used in conjunction with the automatic performance adjuster. In a standard Movex Java installation, the automatic performance adjuster is turned on.

The available shared pools are *SHRPOOL1 through *SHRPOOL60 and some specially named entries such as *INTERACT (for the interactive pool), *MACHINE (for the machine pool), and *BASE (default for system jobs).

Use Operations Navigator or the Work with Shared Pool (WRKSHRPOOL) command to assign the memory size and the maximum number of jobs allowed. Remember that for a threaded application such as Movex Java, each thread is counted for one activity level.

Be sure to add a description that indicates what the pool is being used for, since this can be invaluable when another person tries to determine where subsystems are supposed to be allocated.

The runtime environment for Movex Java uses *SHRPOOL5 as the default memory pool. If this share pool is already defined and in use, you can easily change it to another available memory pool. Figure 7-1 shows the active memory pools, among them which is *SHRPOOL5. Figure 7-2 shows an example of Shared 5 Properties window that you would access to check *SHRPOOL5.

Pool	Description	Current Size (MB)	Current Threads	Maximum Eligible Threads	Total Faults
Machine	Used by internal machine functi...	595,95	57	No maximum	,1
Base	Default system pool	4 288,47	481	864	1,1
Interactive	Used for interactive work	437,58	1	150	,0
Spool	Used for printing	80,00	0	1	,0
Shared 5	Movex Application	2 598,00	39	54	,0

Figure 7-1   Active memory pools
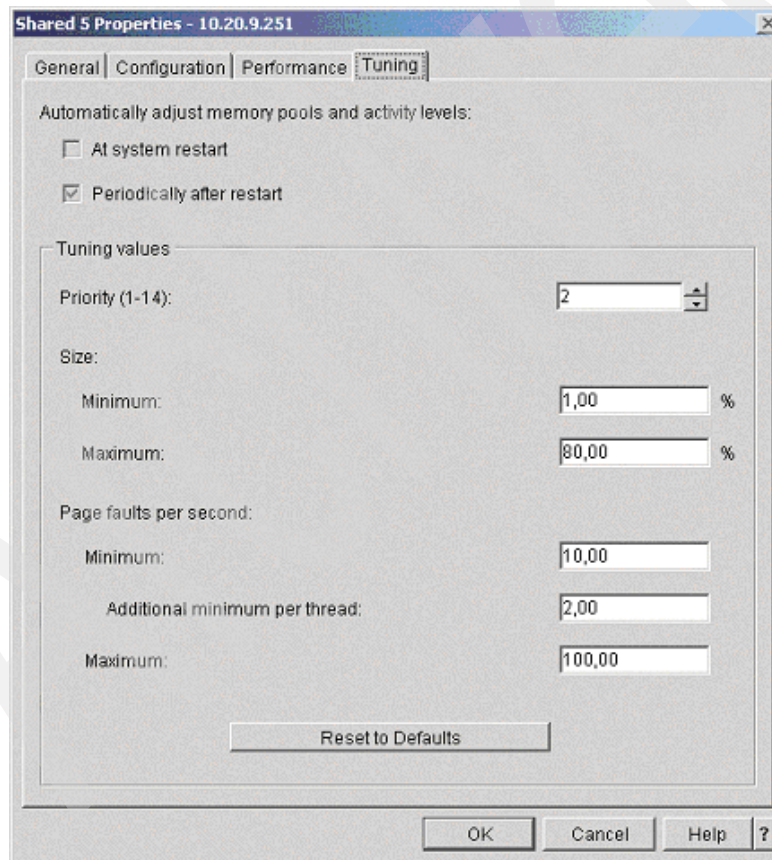
Figure 7-2   Shared pool properties

You can find more information about the amount of memory required, activity levels, and automatic performance adjustment in the following publications:

- ▶ *OS/400* Work *Management*, SC41-5306
- ▶ *iSeries Performance Capabilities Reference*, SC41-0607
- ▶ Chapter 12, "System sizing" on page 177, in this redbook

7.1.3 IBM-supplied subsystems

All jobs in the OS/400 system run in a subsystem. The subsystem controls the allocation of memory and the number of jobs that can run at any one time. It also prioritizes the different types of jobs.

Several subsystems are supplied by IBM, and some of these are used in a standard Movex Java installation. The IBM-supplied subsystems of interest in a Movex Java installation are described in the following section.

QCTL: The controlling subsystem

This is the subsystem where the system console runs. Tasks that require the system to be in a restricted state (such as the SAVSYS command) can only be run from the console. This subsystem is not used as the default, since the controlling subsystem is set to be the subsystem defined by the system value QCTLSBSD and the default for this is QBASE. IBM recommends that you change this value to QCTL. You can do this by using the Restart System Values window in Operations Navigator (Figure 7-3) or the Work with System Value (WRKSYSVAL) command.
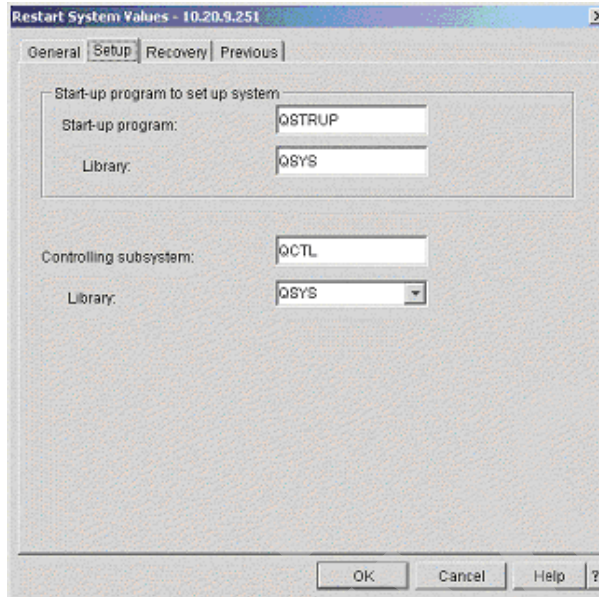
Figure 7-3   Startup program system value

QINTER: The interactive subsystem

The QINTER subsystem is the default subsystem where the interactive 5250 jobs run. This includes dumb (Twinax) terminals, PCs, and IBM Network Stations running 5250 emulation. Note that Movex Java does not support any 5250 clients. The only way to communicate with the application is with Movex Explorer, Movex Web Explorer, the e-business application, or through the Server View tool. The QINTER subsystem may still be used for some for OS/400 work management tasks.

QSPL: Spooling subsystem

This subsystem traditionally controlled print spooling. It is not used in a Movex Java installation. All output from Movex Java is managed with the Movex Output Management product. The native OS/400 printing facilities are not supported in Movex Java.

QBATCH: The batch subsystem

This subsystem is normally designed for batch processing. It is not used in a Movex Java installation, except from CL compilations and for creating internal Java program objects (CRTJVAPGM). All Movex Java-related batch jobs are managed internally in a separate Java virtual machine (JVM).

QSYSWRK: Subsystem monitor

The QSYSWRK subsystem contains jobs that support system functions started automatically at IPL and when the system comes out of restricted state. One of the most important Movex Java system jobs (QSQSRVR) runs in this subsystem. These QSQSRVR system jobs manage the internal SQL JDBC communication to the Movex database. Even when using the record level access (RLA) as the standard database access method in the Movex Java installation, some functionality still uses SQL to dynamically retrieve data from the database.

7.1.4 Movex Java-supplied subsystems

The Movex Java setup can consist of six or more subsystems. The only required subsystem is MVXJVA. The other ones are used mainly if a *multiple JVM setup* is configured. You can learn more about the multiple JVM setup approach 7.2.2, "Multiple JVM setup" on page 94.

MVXJVA: Supervisor subsystem

The MVXJVA subsystem is the subsystem where the first JVM is started (the supervisor). If the multiple JVM setup is not configured, this is the only Movex Java OS/400 subsystem used. All Movex Java subsystems are run in this OS/400 subsystem.

MVXJVAINT: Interactive subsystem

This OS/400 subsystem is used to serve interactive connections to Movex Java from the Movex Explorer client in a multiple JVM setup.

MVXJVABCH: Batch job control

This OS/400 subsystem is used for batch job type work in a multiple JVM setup.

MVXJVAMI: API calls

This OS/400 subsystem is used for API calls (MI programs) in a multiple JVM setup.

MVXJVAASJ: Autostart job subsystem

This subsystem is responsible for running autostart jobs in a Movex Java multiple JVM setup.

MVXJVACRT: Java program creation

This OS/400 subsystem is used for Movex Java service program compilations for example when applying new service packs.

Figure 7-4 shows the list of Movex subsystems.
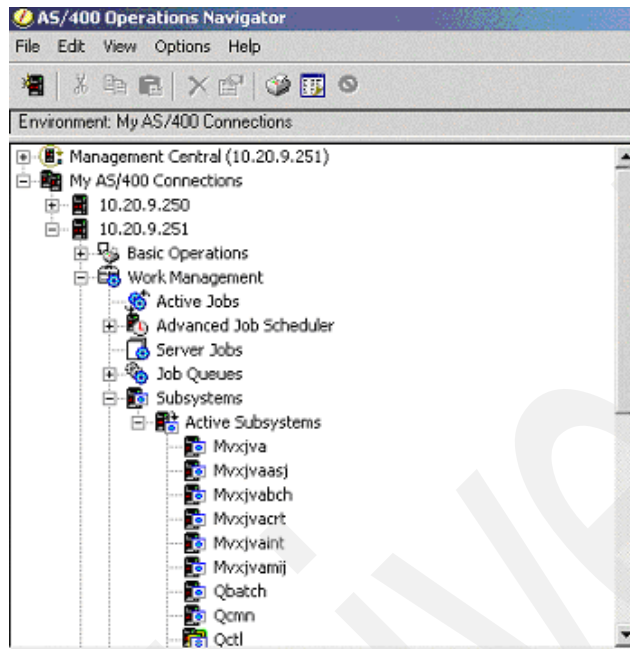
Figure 7-4   Active subsystems

7.1.5 Movex Java runtime environment

This section describes the Movex Java runtime environment, including the Movex job description and Movex class description.

Movex job description

A job description collects a specific set of job-related attributes these are used to control jobs entering the system. The same job description can be used by multiple jobs. During the Movex Java installation process, the following job descriptions are created:

► MVXJVABCH
► MVXJVAINT
► MVXJVACRT
► MVXJVAASJ
► MVXJVA
► MVXJVAMI

Movex class description

A class description contains the run attributes that control the run-time environment of a job. During the Movex Java installation process, the following class descriptions are created:

► MVXJVABCH
► MVXJVAINT
► MVXJVACRT
► MVXJVAASJ
► MVXJVA
► MVXJVAMI

The class descriptions are set in the corresponding job descriptions.

Movex job priorities

The job run priority is the highest priority at which any thread in the job may run. Each thread may have its own thread priority that is lower than the job priority. The run priorities are set in the corresponding class descriptions.

The following run priorities are shipped:

Run priority for class	Default value
MVXJVA	20
MVXJVAINT	25
MVXJVAASJ	30
MVXJVACRT	35
MVXJVAMI	35
MVXJVABCH	40

7.1.6 Java run priorities

The initial Java thread runs at the same OS/400 priority as the BCI job that created it, which runs at the same priority as the job that spawned it. If you start the JVM interactively with run priority 20, the first thread has the same priority. When the actual Java main method starts, the initial thread spawns a new thread whose run priority is lower by a relative amount, usually 5. This results in another thread with run priority of 26. Figure 7-5 shows the Work with Threads menu.

Java has 10 priorities that range from 1 (MIN_PRIORITY) to 10 (MAX_PRIORITY). The higher the integer value used in the java.lang.thread setPriority() method, the higher the run priority. The default Java priority of 5 is the same as NORM_PRIORITY.

When you convert Java priorities to OS/400 run priorities, you use this equation:

```
New OS/400 Thread Priority = (BCI job's run priority + (11 - Java priority))
or (20 + (11-5)
```

```
                          Work with Threads
                                                     System:    SEIRDNGT
  Job:   QJVACMDSRV      User:   MSRVADM        Number:   011369

  Type options, press Enter.
    3=Hold   4=End   5=Display attributes   6=Release   10=Display call stack
    11=Work with thread locks   14=Work with thread mutexes

                                  Total         Aux        Run
  Opt      Thread      Status      CPU          I/O      Priority
           00000018     JVAW       .578         1680        26
           00000215     TIMW       .031          109        26
           000001C8     TIMW       .253          175        26
           000001A4     TIMW       .896          219        26
           0000019B     TIMW      2.899         1214        26
           00000191     TIMW       .325          232        26
           00000118     TIMW       .044          121        26
           000000BA     TIMW       .083          140        26
           000000A3     JVAW     43.782        37358        20
           00000079     TIMW       .946         1671        26

  More...
   F3=Exit    F5=Refresh   F9=Command line   F12=Cancel   F16=Job menu F17=Top
   F18=Bottom
```

Figure 7-5   Work with Threads display

7.2 JVM setup

There are two ways to set up the Movex Java configuration:

► Single JVM setup
► Multiple JVM setup

As always, it is difficult to describe the optimum solution for each customer case.
This section describes both setup types.

7.2.1 Single JVM setup

By default, the Movex Java system is configured with one single JVM running in one OS/400 subsystem. This has some advantages, at least when it comes to work management, since this is the easiest and least complex way to handle the Movex Java environment. From a performance and tuning perspective, this is not so good since there are no possibilities to let the different job types, such as interactive jobs and autostart jobs, run in different memory pools and with different run priorities.

The single JVM approach is normally OK in reference to small installations and test environments. Multiple JVM implementation is a more sophisticated way in a production environment.

Figure 7-6 shows the Work with Movex Java subsystem.

Job Name	Detailed Status	Current User	Type
Mvxjva	Waiting for dequeue	Qsys	Subsystem
Mvxjva	Waiting for time interval	Msrvadm	Batch
Qjvacmdsrv	Waiting for Java program	Msrvadm	Batch immediate

Figure 7-6   Work with Movex Java subsystem

7.2.2 Multiple JVM setup

The multiple JVM setup is really designed for all types of complex environments. With the multiple JVM setup, you can choose to start the different JVMs in the same or in different OS/400 subsystems. The different OS/400 subsystems can run in one single OS/400 storage pool (*SHRPOOL5), or you can route the OS/400 subsystems in which the JVM runs into different OS/400 storage pools.

The division into different memory pools and OS/400 subsystems may be really important in complex environments with high loads on the machine. A benefit of routing the JVMs into different OS/400 subsystems and memory pools is the ability to control the memory allocation and the run priorities.

Figure 7-7 compares a single JVM setup with a multiple JVM setup.
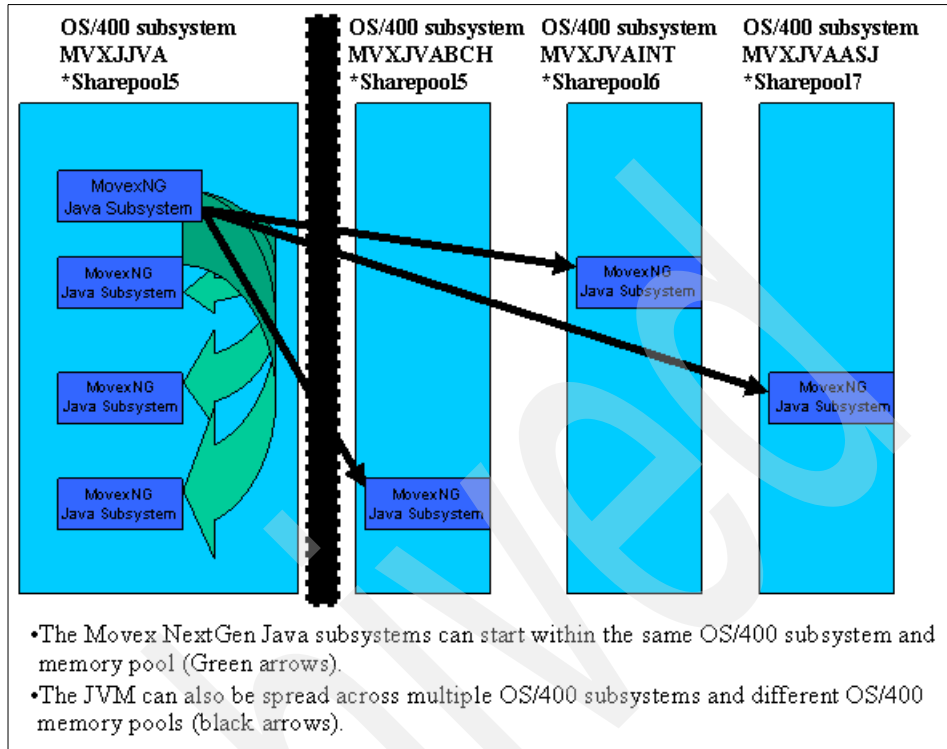
Figure 7-7   Single versus multi JVM configuration

Figure 7-8 shows the list of active Movex Java subsystems.

Figure 7-8   Active Movex Java subsystems

Jobs that would benefit from running in their own memory pools and OS/400 subsystems are:

► Interactive (MVXJVAINT)
► Batch (MVXJVABCH)
► Autojobs (MVXJVAASJ)

To change the memory pool for an OS/400 subsystem, you can use the Change Subsystem Description (CHGSBSD) command (Figure 7-9). Change the Storage size parameter to a suitable sharepool name.

```
                    Change Subsystem Description (CHGSBSD)

 Type choices, press Enter.

 Subsystem description  . . . . . > MVXJVAASJ     Name
   Library  . . . . . . . . . . . >   MVXJJVA     Name, *LIBL, *CURLIB
 Storage pools:
   Pool identifier  . . . . . . .   1             1-10, *SAME
   Storage size . . . . . . . . .   *SHRPOOLxx    Number, *BASE, *NOSTG...
   Activity level . . . . . . . .                 Number
                 + for more values
 Maximum jobs . . . . . . . . . .   *SAME         0-1000, *SAME, *NOMAX
 Text 'description' . . . . . . .   *SAME




                                                                     Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
 F13=How to use this display        F24=More keys
```

Figure 7-9   Change Subsystem Description display

7.2.3 Starting multiple JVMs

To start a multiple JVM environment and to have the JVMs to start in different OS/400 subsystems, you have to set the flag '/srvf:0' when you start the environment. This is normally done in the ordinary start scripts that are used. This flag makes the `mvx.os.MvxStarter` start the supervisor first in its own JVM. Then it calls for a command script, `startsubs.cmd`, that handles the subsequent start of the additional JVMs and subsystems.

Figure 7-10 shows an example of the start script.

```
IF          COND(&Type *EQ '*SBM') THEN(DO)
    IF          COND(&AutoStart *EQ '*YES') THEN(DO)
      RUNJVA      CLASS(mvx.os.MvxStarter)                        +
                  PARM(                                           +
                  '/DPSWVALID:AS400'                              +
                  '/SRVF:0'                                       +
                  )                                               +
                  CLASSPATH(*ENVVAR)                              +
                  OPTIMIZE(40)                                    +
                  PROP((os400.runtime.exec.mode SHELL)           +
                  (java.home                                      +
                   '/QIBM/ProdData/Java400/jdk13')               +
                    (java.version 1.3))                          +
                  GCHINL(200000)                                  +
                  GCHMAX(*NOMAX)
    ENDDO
```

Figure 7-10   Example of the start script

To find this command file, the ordinary system path is used, so the file has to be placed somewhere in that path. In normal installations, the root is added (that is env\'environment') for each environment to the system path by changing the current directory in the start script.

To facilitate an easy setup, the startsubs.cmd file is prepared for each platform and is delivered as a part of the environment folder of the distribution. To activate the feature, make sure that the file is copied to its proper location and then change the **start** command to send the right flag to the supervisor.

The class path settings must be inherited to the subsequent start of the separate JVMs because these are treated as separate jobs by the operating system. To facilitate this, **startsubs.cmd** must correspond to the starting CL program. This is done by naming the environment versions.

Figure 7-11 shows the JVM start flow.
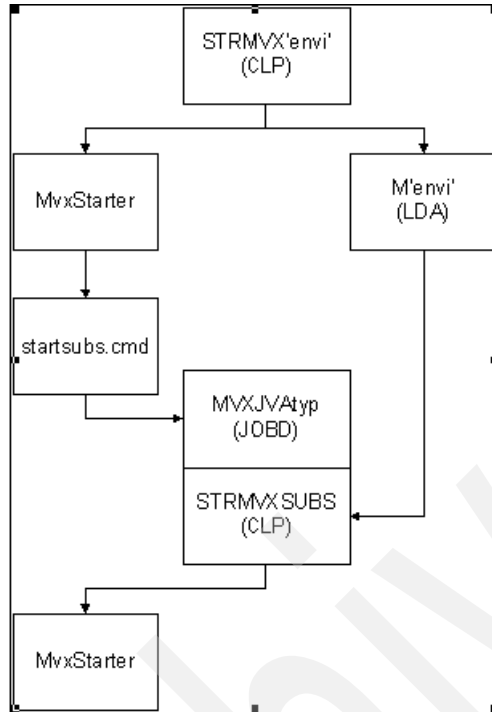
Figure 7-11   JVM start flow

The following actions take place when starting multiple JVMs:

1. The start CL program, STRMVX'envi', contains the setting of the proper path for each environment to start. The path is written to a Local Data Area (LDA) named M'envir' and is stored in the utility library MVXCJVA.

2. The start CL program calls mvx.os.MvxStarter that is started in its own JVM and runs the Movex Supervisor and Movex Server.

3. The mvx.os.MvxStarter then calls the startsubs.cmd file, which by its placement, corresponds to the version that is started.

4. Depending on the type of Movex Subsystem that is starting, the call for the CL program STRMVXSUBS is done by using the proper job description. The job description is then used to connect to the right OS/400 subsystem and to assign the right values for priorities, timeslice, and so on.

5. The CL program STRMVXSUBS retrieves the saved class path and then calls mvx.os.MvxStarter to start the proper Movex subsystem in the newly started JVM.

7.2.4 Starting multiple environments

To make it possible to start different versions (environments with different Movex.properties files), a technique with naming the starting CL programs is used. This means that the name of the CL program has to be STRMVX'envi', where *'envi'* may be any four alphanumerical characters that make sense.

The same name must be entered as the *version* in the command file startsubs.cmd. Note that the variable name *version* in this context does not mean version of Movex.

Figure 7-12 shows an example of an environment parameter script.

```
/* Set Environment root                         Need to be modified! */
  CHGVAR    &EnvRoot VALUE('/Movex_v12/environment/envi')
```

Figure 7-12   Example of environment parameter script

Depending upon what the environment root is in the start CL program, the corresponding startsubs.cmd command file must be placed in that catalog as shown in Figure 7-13.
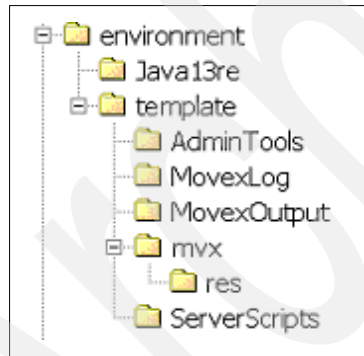
Figure 7-13   The environment folder

Figure 7-14 shows an example of the startsubs script.

```
# ================================================
#
#                 Start Movex Subsystems
#                 ---------------------
#
#
#         Copyright (C) Intentia R&D AB 2000.
#         All rights reserved.
#         Licensed material property of Intentia.
# ================================================
# Change log
# ----------
# Number Date    User         Description
#
#         001217 iehpe0       New function
#         010510 iehpe0       Changes to 11.5.0
# ================================================
#   The variable environment in this script must match the
#   environment name used in the CL-program that starts the
#   is used to start the environment (STRMVXenvi).
#
# ================================================

export environment="envi"

export QIBM_MULTI_THREADED=Y
if [ "$3" = "-strsa" ] ; then
        a="MVXJVAASJ"
fi
if [ "$3" = "-strsb" ] ; then
        a="MVXJVABCH"
fi
if [ "$3" = "-strsm" ] ; then
        a="MVXJVAMIJ"
fi
if [ "$3" = "-strsi" ] ; then
        a="MVXJVAINT"
fi

echo $1
echo $2
echo $3
echo $4
echo $a

system "SBMJOB CMD(CALL PGM(mvxjjva/STRMVXSUBS) PARM('"$1"' '"$3"' '"$1150"'))
JOBD(" mvxjjva/$a ") JOB("$a")"
```

Figure 7-14   Example of the startsubs script

If multiple environments are set up (Prod and Test), you must copy the
startsubs.cmd command file between the different Environment root catalogs.
You must also edit the file to correspond to the name of the starting CL program.
In the example above, the startsubs.cmd found in the template folder at
installation time is copied to the test folder when the environments are set up.

7.2.5 Initial setting of heap sizes

The garbage collector execution is based on the initial heap size parameter
GCHINL of the RUNJVA command. In a Movex Java installation, those
statements are located in the server start (CL) program named STRMVXxxxx as
illustrated in Figure 7-15.

```
IF          COND(&Type *EQ '*SBM') THEN(DO)
     IF          COND(&AutoStart *EQ '*YES') THEN(DO)
       RUNJVA      CLASS(mvx.os.MvxStarter)                        +
                   PARM(                                           +
                   '/DPSWVALID:AS400'                              +
                   '/SRVF:0'                                       +
                   )                                               +
                   CLASSPATH(*ENVVAR)                              +
                   OPTIMIZE(40)                                    +
                   PROP((os400.runtime.exec.mode SHELL)           +
                   (java.home                                      +
                    '/QIBM/ProdData/Java400/jdk13')               +
                     (java.version 1.3))                          +
                   GCHINL(200000)                                  +
                   GCHMAX(*NOMAX)
     ENDDO
```

Figure 7-15   RUNJVA instruction in the STRMVXxxxx program

Movex Java is shipped with the default initial heap size of 200 MB. The initial heap size parameter is a threshold that triggers a garbage collection cycle. It does not trigger a garbage collection cycle based on the absolute size of the heap, but on the amount the heap has grown since the last cycle (or since start up, if no cycle has yet been run).

If you specify an initial heap size of 200 MB, the first cycle runs after 200 MB of space is allocated. The garbage collector then frees some of that space for reuse. Then it waits until another 200 MB is allocated before running again.

If the garbage collector freed 70 MB in the first collection cycle, it does not run again until the heap size is at least 330 MB. The iSeries garbage collector does not do heap compaction. That 70 MB of heap is unused until it is needed again by the JVM.

> **Note:** The iSeries definition of initial heap size is different from other platforms. On other platforms, the initial heap size determines the initial amount of memory to allocate, and the maximum heap size has more influence when garbage collector runs. In Movex Java on iSeries, do not specify the maximum heap size (that is *nomax). The reason is that "normal" collect is run as an asynchronous thread in parallel with other threads. If the maximum heap size is reached, all other threads are stopped while collection takes place. The latter is of cause unacceptable from a performance viewpoint. This is different from other platforms. Cross-platform tuning tips may indicate to set the maximum heap size.

7.2.6 Garbage collection monitoring and settings

As stated previously, the initial heap size essentially determines the frequency of garbage collections. You can monitor the time interval between garbage collections by setting the OPTION parameter of the RUNJVA command to *VERBOSEBC.

7.2.7 Memory pool settings

In a standard Movex Java installation, the system value QPFRADJ is turned ON (3=Automatic adjustment). This means that the OS/400 handles memory allocation, activity levels, and so on. For most installations, this is sufficient. If for some reason you do not want to use the automatic performance adjuster, here are some basic tips to be aware of:

► Make sure the pool in which the Movex Java JVMs run in is large enough to hold the size of the total heap. Monitor the heap size during peak time and be sure the total heap size is smaller than the allocated amount of memory for the memory pool. If the memory pool is not large enough to hold the heap, you may encounter bad performance when the garbage collector starts to clean up objects that are paged out on disk.

► Be sure that the activity level on the share pool is set to a high value because, in a threaded application like Movex Java, each thread is counted for one activity level.

► If the Movex Java is the only application running in the machine, you can change the "minimum percentage of total main storage to allocate" for the interactive (*INTERACT) and spool (*SPOOL) pools to 1%. These may seem like small changes, but in a machine with 2 GB of memory and a default value of 5%, this translates to 100 MB of wasted memory for each pool.

Figure 7-16 shows the subsystem tuning parameters.
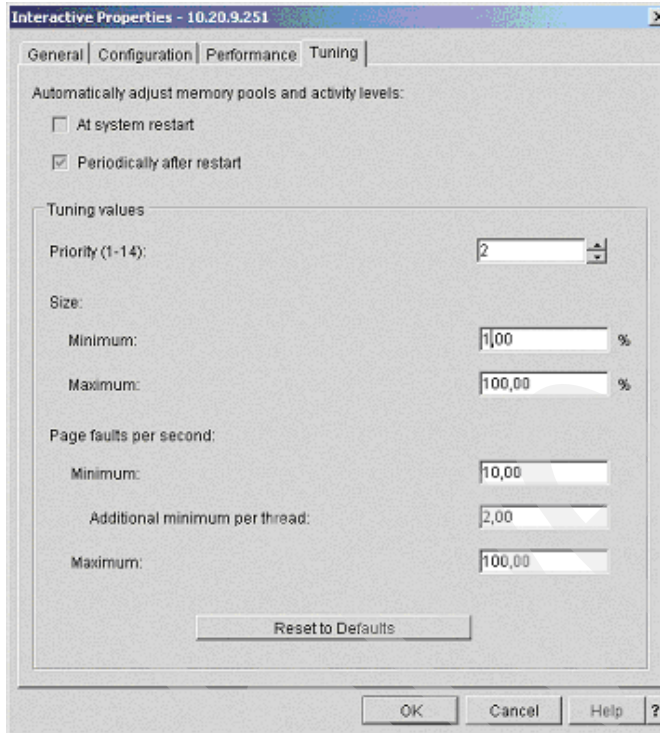
Figure 7-16   Subsystem tuning parameters

7.3 Server View

The Server View tool provides common work management functions for the Movex Java environment. Server View should be used for monitoring, troubleshooting, and performance analyzing of your Movex Java environment.

This section begins with an overview of the Server Views functions and then explains how to start Server View and work with Movex Java View. You can find more information in the Server View documentation on the Intentia Wire.

Server View offers the following functions:

► Monitor and control on the physical server and process level.

► Supervisor view that displays servers connected to the supervisor.

► Server View that displays subsystems connected to the server.

► Subsystem view that displays Movex jobs running on the subsystem.

- ► Profiler view that provides performance-related troubleshooting and optimization of Movex programs.

- ► Log view that displays the log file generated by Movex jobs and middleware.

- ► Counters view that contains information about performance and resource consumption of Movex subsystems.

- ► Find view that is used to find actual locations of Movex program class files and display version information of those.

- ► Run view that runs Movex batch programs.

7.3.1 Starting Server View

When the Server View port is written in the Movex.properties file, you can use Server View in Internet Explorer.

Complete these steps to start Server View:

1. Open Internet Explorer or Netscape Navigator.

2. Type the address to Server View:

   ```
   http://hostname:6666/
   ```

The Server View is now started, and you can work with the environment as illustrated in Figure 7-17.
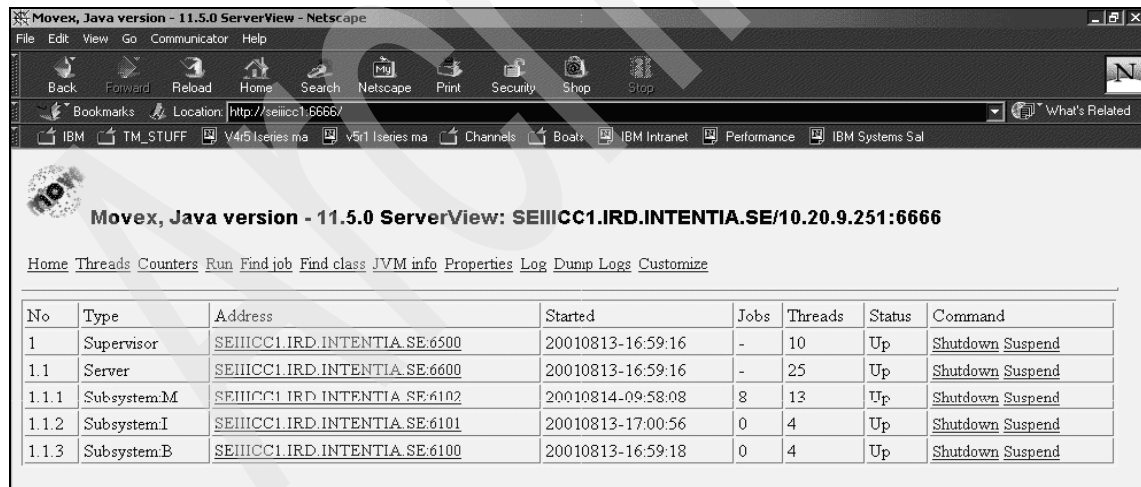
Figure 7-17   Server View main menu

7.3.2 Movex Java view

The home view is called the *Movex Java view*. It contains a list with links directly to the supervisor, server, and the subsystem view.

For each installation, there is one supervisor. It is possible to connect one or several servers to the supervisor and one or several subsystems to each server. In Server View, they are built up and numbered in a sort of tree structure.

Most Movex Java installations have several service ports, usually for test and live versions of the system. These can be managed and accessed by creating and using bookmarks or the Favorites feature of a Web browser.

Movex OUT and printing

This chapter outlines the Movex OUT solution and the technology behind it. Movex OUT simplifies and improves information management. Movex OUT automates the output from Movex, reduces the costs for output, and converts output into a powerful competitive tool.

Movex Output Management is a connection to fax, e-mail, Portable Document Format (PDF) publishing, and electronic archives. It offers total control of everything produced by the business application.

Movex Output Server is the engine for Movex Output Management. It refines raw output from the Movex Java application server and forwards the processed output to an appropriate output device. The Movex Output Server intelligently interprets and processes the configuration specifications on how to handle output information from the business application. These configuration specifications are set in Movex Output Tool, the user interface of Movex Output Server.

8.1 Movex OUT components

This section describes the different components in the Movex OUT solution:

- ► **Movex Dictionary** is the set of definitions used by the Movex Output Server for formatting and distributing the output.

- ► **Movex Output Server** is the real-time formatting and distribution engine. Runs on Microsoft Windows and on most UNIX platforms. By adding optional modules, you can use a wide variety of output formats and distribution channels.

- ► **Movex StreamIN** (structured flow) is Movex Output Server's interface module for receiving input as a continuous data stream in a predefined, non-formatted data structure (for example, tab-delimited).

- ► **Movex PageOUT** provides print-ready output in a variety of formats, including PostScript, PCL4, PCL5, and Canon CAPSL. In addition, Microsoft Windows native drivers are supported for other output needs.

- ► **Movex StreamOUT** supports output in unformatted, structured data formats using an easy-to-use designer tool. The StreamOUT output can be saved as text files further processing by other systems.

- ► **Movex Output Tool** is a Windows-based tool for output design and formatting, entering distribution rules, system configuration, and maintenance.

- ► **Movex Forms Design** is a Windows-based tool for creating electronic forms that can be used as overlays in Movex Output Tool.

- ► **Movex Output to Fax** connects output to Fax software like TOPCALL, Rightfax, and Zetafax for seamless Fax distribution.

- ► **Movex Output to e-mail** provides seamless e-mail integration. Output can be distributed in text format or as attachments depending on the solution chosen.

- ► **Movex Output to PDF** adds Adobe Acrobat PDF formatting capabilities to Movex Output Server, for further distribution, archiving, and platform-independent viewing.

- ► **Movex Output to ODBC** extends Movex Output Server's scripting capabilities by adding ODBC command support to access ODBC-compliant databases.

- ► **Movex Output to RePrint** adds, reprinting, backup, basic archive, and search and retrieval functionality to Movex Output Server using a Microsoft SQL Server database on Windows platforms.

- ► **Movex RemotePrint** adds printer boosting to WAN connections using ISDN, Point-to-Point, or leased lines with bandwidth problems. Acts as a local server by handling the graphics locally and is attached by network to central a

MOVEX OUT server. There are no attachable module options, and it can only connect to two printers.

8.2 Movex Out technology

A stream file represents output from Movex Java. This file contains all output data and information about how the output is to be distributed. Upon completion of the Movex printout function, the stream file is transferred to the Movex Output Server using a socket interface. The rest of the job is handled by the Movex Output Server, which formats and directs the output. Figure 8-1 illustrates the Movex OUT design.
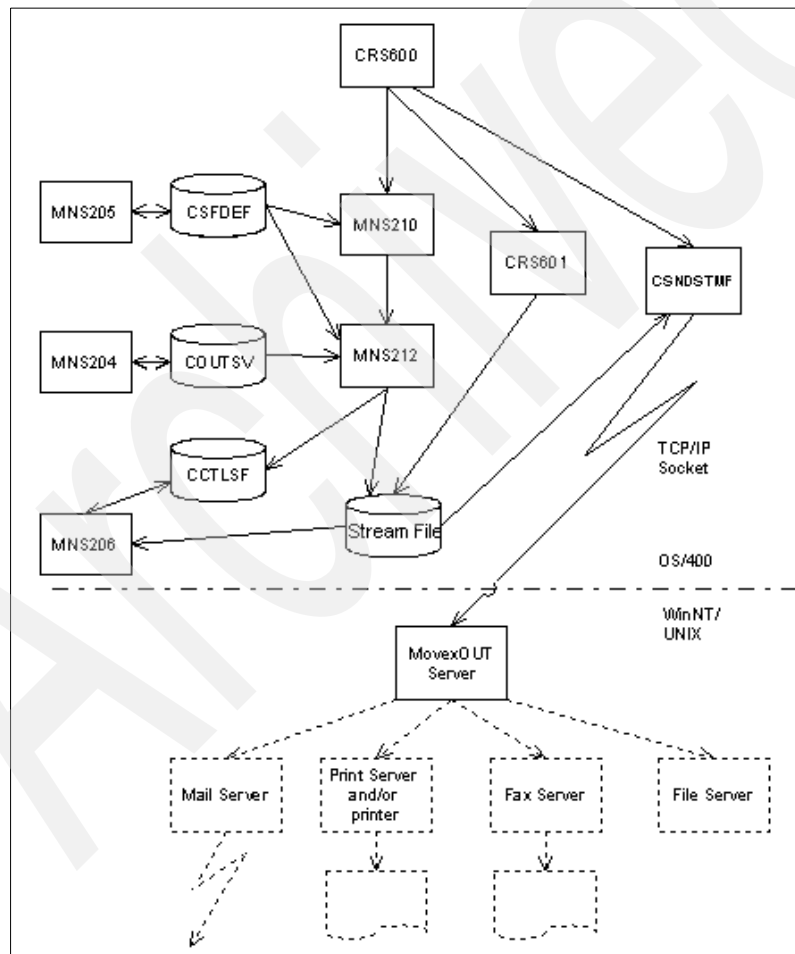
Figure 8-1   Movex OUT design

8.2.1 User's perspective

The user selects what to produce. This depends on how Movex has been set up and prompted on how the output is to be distributed. Configuration issues reside in the program 'Output Definition. Open' (MNS205) and to some extent on 'Output Server Definition. Open' (MNS204).

Output tracking and monitoring is done by using the function 'Output. Open' (MNS206). You can override the settings in 'Output Definition. Open' (MNS205) by 'Partner Reference. Open' (CRS945) and 'Partner Reference – Media. Open' (CRS949).

8.2.2 Customer's view

Customizing output without having programming skills is made possible by using the graphical Movex Output Tool.

Reuse of modifications when upgrading Movex to a new version is easy. Since the interface between Movex and Movex OUT consists of a structured file, modifications (assuming no fields are deleted or renamed) work transparently with different versions of Movex.

8.2.3 Language handling

Constants (labels) are retrieved from the same language files used by Movex Explorer. Using industry solution-specific files can be configured. The language, in which the output constants are to be presented, is controlled by the user settings in Movex and is passed to Movex OUT in the stream file.

8.2.4 Modification directories

Intentia recommends that you store modified project and exporting files in a directory separated from the standard files. To make this possible, a keyword (-scandux) is used in the argument file. Enter multiple -scandux lines, and a classpath type of mechanism is created.

8.2.5 Agent control

Movex controls all directing of output. Technically, keywords are sent from Movex via the stream file telling the output server where to direct the output. This is accomplished by using a set of standard queues and an agent enabled feature in the definition files.

8.3 Movex Java printing features

The printing features of Movex Java include:

► **Wide Distribution Capabilities**

Movex Output Server can deliver the processed data to a large range of output devices, such as file, PCL printers and PostScript printers, and in such formats as PDF files, e-mail, EDI, and fax.

► **Platform Independent**

Movex Output Server is built for a variety of platforms, such as most UNIX dialects and Windows versions.

► **Novell Compatible**

Movex Output Server can run in a Novell environment (Bindery or NDS).

► **Storage Options**

Movex Output Server supports the option to store documents in a Lotus Notes database.

8.4 Hardware requirements for Windows

The hardware requirements for running the Movex Output Server on Windows are:

► **Computer/processor**

System using an Intel Pentium or higher processor.

► **Memory**

16 MB of memory (RAM) required, 64 MB of memory recommended.

► **Hard disk**

25 MB of available hard-disk space.

► **Operating system**

Microsoft Windows NT and 2000.

► **Peripheral/miscellaneous**

The following networks are supported using native protocols:

– Microsoft Windows NT Server
– Microsoft LAN Manager
– Novell NetWare
– TCP/IP-based networks
– IBM LAN Server

8.5 Setup

The following entries in Movex.properties are related to output management:

- ▶ **mvx.app.common.MvxLang.parentDir** specifies where to store output files on the integrated file system for example:

    ```
    mvx.app.common.MvxLang.parentDir=/Movex_v12/environment/template/
    MovexOutput
    ```

- ▶ **runtime.mvxout.streamservecodepage** specifies which codepage to uncode output data to, for example:

    ```
    runtime.mvxout.streamservecodepage=UTF8
    ```

8.5.1 Setting up Movex Output Server

Movex Output Server is configured using the three files listed in Table 8-1.

Table 8-1   Movex Output Server configuration files

File	Description
mvxarg.arg.	An argument file that determines log levels, where to scan for .dux files, etc.
movex.dua	The platform file used in run time by the server. It contains the setup for all input and output queues used by the server.
quealias.	Contains the mapping information used for translating Movex printer names to network names.

8.6 The mvxarg.arg argument file

This section explains the mvxarg.arg file. Figure 8-2 shows the contents of the mvxarg.arg file.

Figure 8-2   Argument file

Table 8-2 explains the keywords of the argument file.
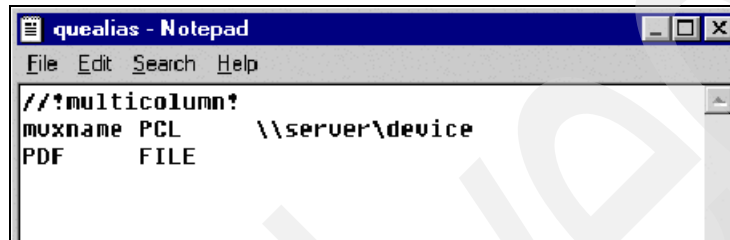
Table 8-2   Argument file keywords.

Row	Explanation
-demo	Required if running an unlicensed version of the server.
-ll	Log level. Values 0 to 10 are valid, where 10 is the highest level of logging.
-rmlog	The name of the produced log file. In this case, it resides in the server's working directory.
-wsin	A structured log file for current input.
-sprog	Allocates memory for scripts, for example, the total number of bytes in the memory area.
-prn	Specifies the path for Overlay files in relation to the working directory.
Drivers	These statements point out the locations of the driver files to be used, which in this case, is a subdirectory to the servers working directory.
-mvxlanginit	Controls the order in which constants from Movex language files are used.
Movex.dua	The name of the platform file to be used.
zer.cml	Invokes a global script used for initial zero suppressing of numeric data.
-scandux	Points out where to look for the .dux files to be loaded. Multiple -scandux statements are allowed to create a library list function. The server loads all events not previously loaded found in each directory stated in a -scandux statement.

You can learn more about these parameters and others of the argument file in the Movex OUT reference manual located on the Intentia Wire.

8.7 The Queue Alias file (quealias)

The Queue Alias file is used to map Movex printer names to the network device name. The device type (PCL, PostScript, or other) of each printer is also defined here. The Queue Alias file is shown in Figure 8-3.

```
quealias - Notepad
File  Edit  Search  Help
//?multicolumn?
muxname  PCL       \\server\device
PDF      FILE
```
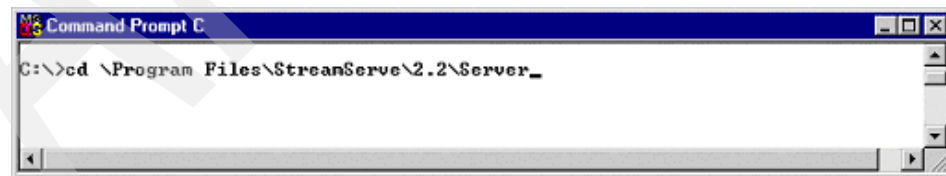
Figure 8-3   Quealias example

8.8 Starting the Movex Output server

This section explains how to start the Movex Output server to run as a window. How to start the server as a service is described in the Movex OUT reference manual located on the Intentia Wire.

Follow these steps to start the Movex Output Server to run as a window:
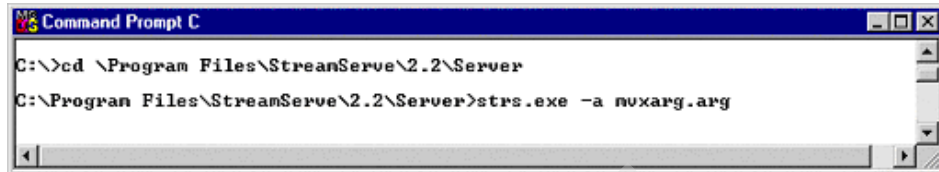
1. Open a Command Prompt window.

2. Change the directory to the work directory for the server as shown in Figure 8-4.

```
Command Prompt C
C:\>cd \Program Files\StreamServe\2.2\Server_
```

Figure 8-4   Server window (Part 1 of 3)
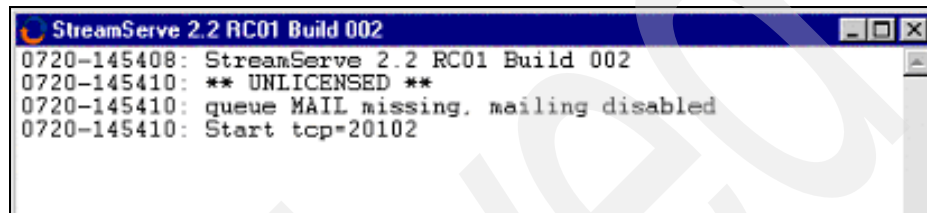
3. Specify `strs.exe -a` and the argument file parameter as shown in Figure 8-5.

Figure 8-5   Server window (Part 2 of 3)

The server window that is started is shown in Figure 8-6.

Figure 8-6   Server window (Part 3 of 3)

The server now listens for input from Movex via a TCP/IP socket with port
number 20102.

9

Security

This chapter explains the different security aspects when Movex Java is set up to run on the iSeries server.

117

9.1 Movex security model

Movex security is set up to cover all aspects of security, when it comes to installing and using Movex Java. Figure 9-1 illustrates the Movex security model.
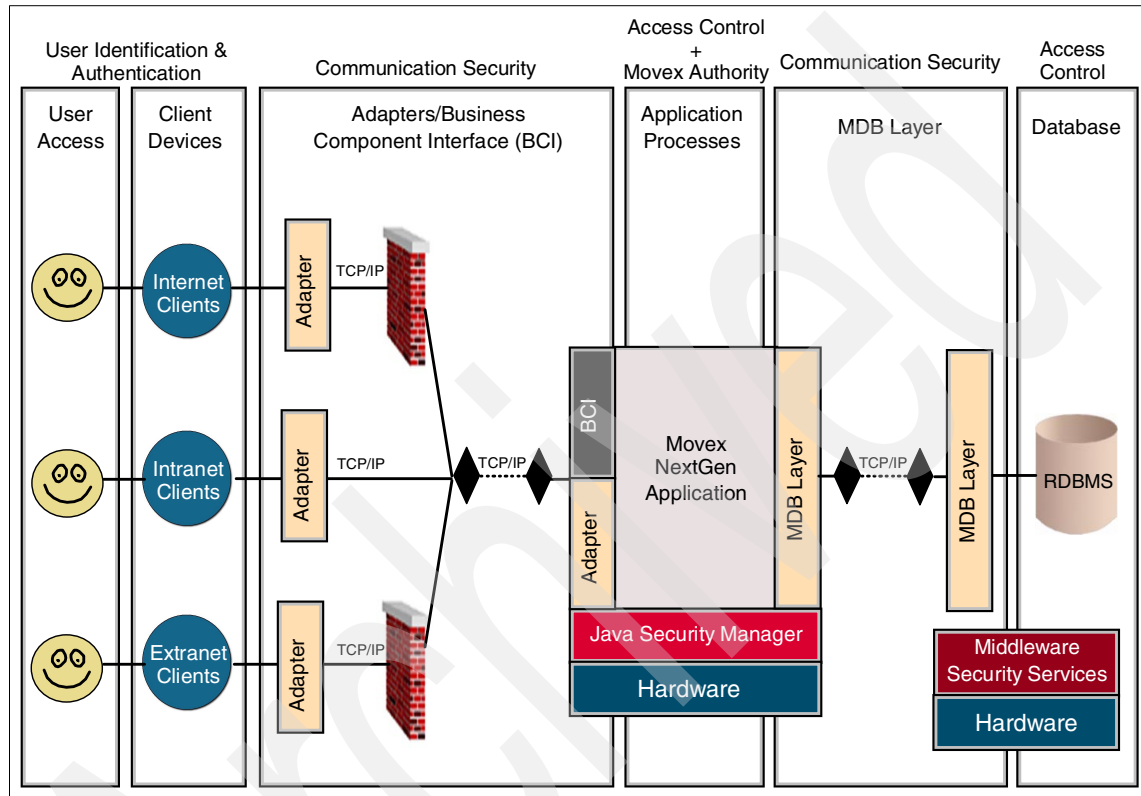
Figure 9-1   Movex security model

The different areas that are covered by the security model are:

► User identification and authentication
► Communication security
► Access control
► Movex authority

9.2 User identification and authentication

Password validation, which is a part of User identification and authentication, needs to be configured when an installation of Movex Java is performed. This is different from earlier generations of the Movex application that relied on the platform that used to handle these issues.

This section offers some basic hints on setting up an installation where all users who need to access an application are fully authenticated.

9.2.1 Password validation

When Movex Java is started for the first time after installation, the default setting is that password validation is not done. This is not the recommended setting for a fully live installation, but it is the smoothest way to get the installation up and running from the start.

Before you can start to set up password validation, you need to determine where it should be done. The natural answer is to do it on the application server, but is that really the best place?

Imagine this scenario. The Movex Explorer is a Windows 32-bit application. Because of that, there is in most cases, a Windows domain in place that handles the user identification and authentication for the different clients that are used. If most of the users in that domain also gain access to the Movex application, it could be easier for all parts that Movex uses the password validation on Windows, even if the application server platform is something else. On the other hand, if not all or most of the users of the Windows domain should be authorized to use Movex, it may be easier to use password validation on the application server platform instead.

> **Note:** Users defined as valid users in the operating system of the application server are only used for password validation. There are no other links between the user profile on the server and the user definition in Movex.

To activate password validation, you need to perform two steps:

1. Set the settings in the configuration file, Movex.properties. This is the place where you specify the password validation to take place.

2. Change the start program for the application server so that the application server starts with the password validation activated.

When you perform these two steps, the passwords are validated for the users who sign on to Movex through the client interface, Movex Explorer.

Notes:

► The settings in the following sections do not affect whether the user signing on via Movex Explorer has to enter a password. Movex Explorer always requires a password to be entered, while the following settings affect whether and where the entered password should be validated.

► For recovery purposes, where you need to temporarily disable the password validation, it is enough to start the application server without password validation switched on.

9.2.2 Movex.properties

The entries in the Movex.properties file that specify where the password validation should take place are presented in the following list:

Property	Description
dir.logon.nt.domainController1	Specifies the name of the Windows domain controller that should be used for password validation if the password validation flag in the start program specifies that Windows NT should be used.
dir.logon.nt.domainController2	Specifies the name of an alternative Windows domain controller that may be used to validate the passwords.
dir.logon.nt.group	Name of a Windows group that a user signing on must be a member of to have their password validated.
dir.logon.as400.server	The IP address of the iSeries server where password validation should be done. If the password validation flag is set to iSeries in the start program, this property must be set.

9.2.3 Starting Movex

By sending the following flag with the start request for the Movex application server, subsequent connection attempts from the Movex Explorer clients require the passwords entered by the users to be validated according to the setting of this flag:

```
/dPSWVALID:<DUMMY|AS400|NT>
```

The following values are included in the flag:

DUMMY   No validation takes place.

AS400   The iSeries set in Movex.properties is used for password validation.

NT   The Windows domain controllers listed in Movex.properties are used for validation.

9.2.4 Movex user definition

This section describes the basics of the Movex user definition.

'User. Open' (MNS150)

The users that should be able to use the Movex system must be defined in the Movex program 'User.Open' (MNS150). Figure 9-2 shows 'User.Open', panel B.
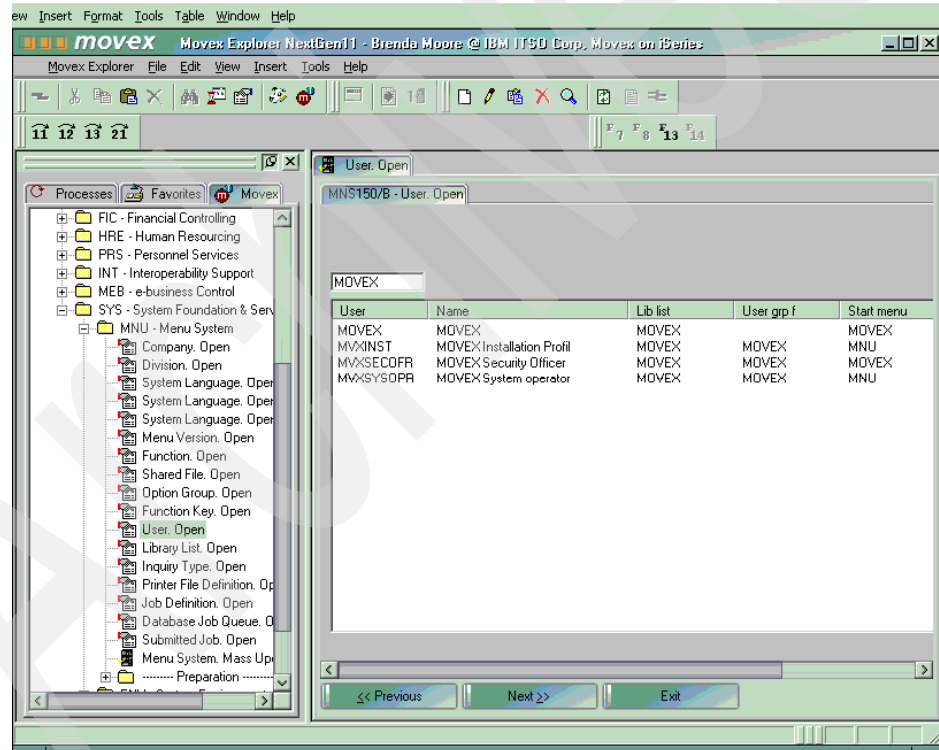
Figure 9-2   'User. Open', panel B

Because Movex Java is still synchronized with the previous generation of the application, some of fields in this program are not valid for a Movex Java installation.

For more information about this program, refer to 9.5, "Movex authority system" on page 146.

Delivered user profiles

At delivery time, some general Movex users are already defined in 'User. Open' (MNS150). These are internal profiles, and as such, they do not necessarily need to be defined for password validation. An exception may be the user MVXSECOFR that has manager authorities to the Movex functions and, thereby, overrides all security checks inside the Movex application. Figure 9-3 shows the delivered user profiles.
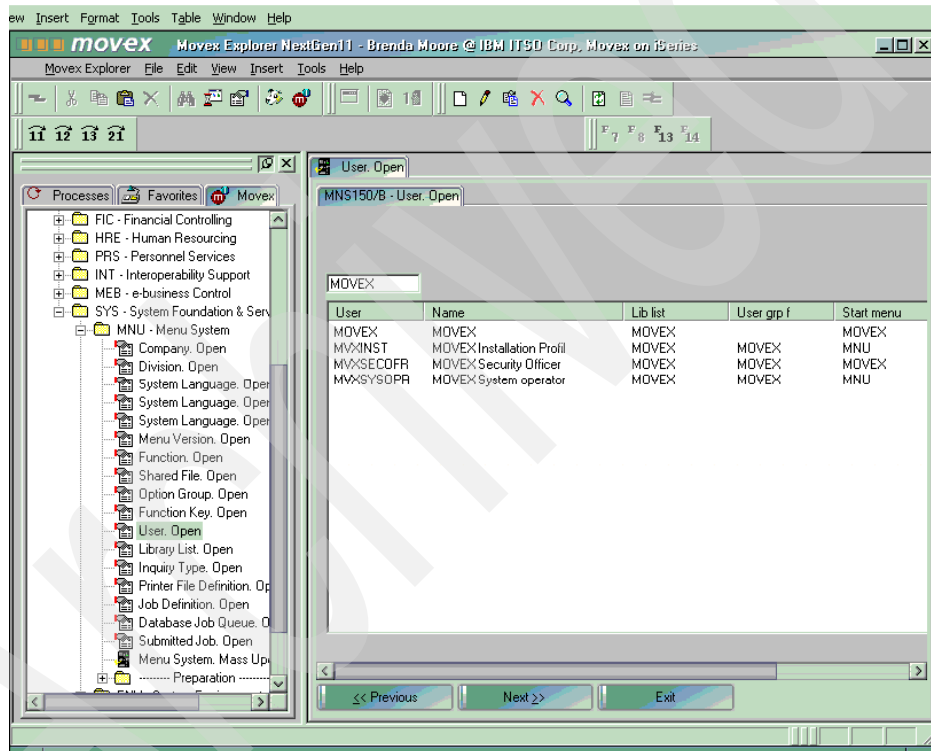
Figure 9-3   Delivered user profiles

The different delivered user profiles include:

MOVEX   General user of the system, used for such setup purposes as the autostart jobs.

MVXSECOFR   Movex Security Officer has manager rights to all functions and thereby overrides all security settings.

MVXSYSOPR Movex System Operator is the receiver of internal diagnostic messages from the application.

9.3 Communication security

Because of the layered architecture of the Movex Java application, you must give extra consideration to communication security. In the case of using an iSeries server as the installation platform for Movex Java, it is a little easier, because the application and the database reside in the same machine. Still there are some areas that need to be handled.

9.3.1 Port allocation schema

Table 9-1 presents all properties in the Movex.properties file that defines the ports that should be used. Note that these are the default values that will be used if nothing else is stated.

Table 9-1   All properties of the Movex.properties file

Property	Type	Description
boot.suervisor.port=6500	I	Startup port for Movex Supervisor.
boot.batchdispatcher.port=6800	E	Port that specifies where the batch dispatcher is to be found. This port is used for the MI program connection, among other things.
boot.servers.port=6600	I	Startup port for a Movex server.
boot.subs.baseport=6100	I	Base port for a Movex subsystem instance.
boot.subs.usrport=200	E	Port increment relative to a subsystem's base port that will be used by a subsystem.
boot.kbdserver.port=6060	E	Port for connecting a Telnet interface session to a running JVM.
boot.wm.html.port=6666	E	Port for connecting the Server View interface.
boot.medispatcher.port=9973	E	Port for connecting a Movex Explorer interface session.

Property	Type	Description
boot.meproxy.baseport=7000	I	Base port for connecting a Movex Explorer single port interface session.
boot.meproxy.range=1000	I	Range of ports that are available for use of Movex Explorer single port interface sessions, relative to the base port.
mvx.app.pgm.EDI.port=7409	I	Port for connecting to the EDI server.
boot.dbdispatcher.port=6801	I	Port for connecting the Server View to the dbdispatcher of a remote database server.
runtime.standAloneDbsPort=9993	I	Port for connecting the Movex database running on a remote JVM.

To run multiple environments in the same physical server, use the same port numbers for all environments, but add x0 000, where *x* equals 1 to 5 for each port number configured. This gives you the ability to run up to six parallel environments in the same system. If you have more environments, you must develop a more sophisticated approach.

9.3.2 Firewalls

To run communications through firewalls, the used ports must be known so a proper configuration of the firewall may be made. Of interest in this case is which ports are used for external connection of clients and other programs.

MI programs

MI program is the Movex term for a kind of API program that is used for connecting to other applications, such as e-business solutions to the Movex Java installation. Because these programs must be reached from outside the application server, you need to keep the following points in mind.

Basics

There are three settings in the Movex.properties file in regard to MI programs. The most important setting is the MI program entry point, which is the port used for clients to connect to MI programs in Movex Java. This is the boot.batchdispatcher.port, for which the default is set to 6800.

The others are boot.subs.baseport, determining on which ports the subsystems will listen, and the boot.subs.usrport, which actually is an increment (relative the owning subsystem) for the port resolvers used by clients to connect to MI programs.

For example, the boot.batchdispatcher.port is set to 6800. Movex Java is configured to have a maximum of 200 subsystems. The boot.subs.baseport=6100 makes subsystems occupying ports 6100-6299. An appropriate boot.subs.usrport setting is boot.subs.usrport=200, which makes ports 6300-6499 occupied by the port resolvers. If you have 201 subsystems running with these settings, there is a potential port collision on port 6500.

Externally accessible ports

The ports needed to be accessible from outside Movex Java for MI programs bbp = boot.batchdispatcher.port, bsb = boot.subs.baseport, and bsu = boot.subs.usrport are (above example parameters within parenthesis):

```
bbp (6800)
bsb + bsu -> bsb + 2 x bsu - 1 (6300 -> 6499)
```

Single port Movex Explorer

Movex Explorer always connects to the application server through the port defined in the property boot.medispatcher.port for which the default is 9973. The application server then dynamically advises a port to use for further communication. This dynamically advised port cannot always be predicted, and is, therefore, difficult to define for use through a firewall.

Instead a special communications driver may be used, named *Movex Explorer Single port driver*. This solution, which requires more resources on the server, works by the help of a proxy server built-in to the application server. This means that all communication between the client and the application server uses the external port defined by the property boot.medispatcher.port. The proxy server internally handles the communication through ports defined in the properties boot.meproxy.baseport and boot.meproxy.range.

Telnet interface

The application server has a built-in Telnet server that may be used to control the operations. If you plan to use this facility through a firewall, the port defined by the property boot.kbdserver.port must be allowed.

Server View interface

The normal tool for surveillance of the Movex Java server is HTML based and named Server View. By using this tool, you can do most of the administrative tasks needed for your installation. If this tool is to be used through a firewall, the port defined by the property boot.wm.html.port must be allowed.

Additional security for Server View

There are two properties that give additional security for the Server View:

boot.wm.html.requireLogon	Specifies whether the user attempting to use the Server View must enter a valid password.
boot.wq.addr	Lists the client IP addresses that are allowed to connect to the Server View port.

9.4 Access control

When you install Movex Java, you must configure the access control security so that no one gains unauthorized access to the application itself or to the database.

This section offers some basic hints about how to set up the access control. That is, you can have full control over the different components of the installation from a management point of view. In practice, the names used for users and groups in this section may be changed without affecting the functionality of the installation. Intentia recommends that you use its naming pattern since it is proven to work.

9.4.1 Access control setup considerations

You must set up access control in Movex Java on a platform-dependent basis. This is because it relies upon mechanisms found in the underlying operating system in use on the specific platform. Still you can see some common patterns. The approach is to use the same rules for naming as much as possible to make the different installations easier to understand.

The basic principle used is Mandatory Access Control (MAC), which means that the single user cannot change the rights afterwards. Instead all maintenance of the access control has to be set by one of the appointed management users.

Also an ordinary user of Movex Java does not have any access rights to neither the application nor the database. All functional access to ordinary users is granted through the ordinary use of the application.

All platforms in use for the moment – OS/400, Windows, and Solaris – offer basically the same access operations (read, write, and execute rights) with some minor differences. This means that the same level of security may be reached regardless of the choice of platform.

There are two different access control models in use, depending upon in which environment the installation is done. The main difference between these two models is whether the application and the database reside in the same server (Centralized model) or they are placed on different servers (Decentralized model). It is also important to know whether the database management system (DBMS) shares the authority model with the underlying operating system.

9.4.2 OS/400 platform overview

System security is an integrated function in OS/400. It is implemented at the instruction level and controls all software functions on the iSeries server. Users are identified and authenticated by a single security mechanism, at the system level, for all functions and environments available on an iSeries server. This includes program execution, database, applications, and all objects on an iSeries server. It also means that they are all under the same security control.

In OS/400 terminology, an authority is the permission to access an object. The object owner and the security officer (or other *ALLOBJ users) can grant or revoke authority to an object. It is important to understand the difference between authority to an object and authority to the data in the object. Such operations as moving, renaming, saving, or deleting apply to the object accordingly. It is possible to have authority for these operations without having access to the data stored in the object. Likewise, you can have full access (read, write, update, delete, or execute) to the data in an object without having full authority to manipulate the entire object.

9.4.3 Scenario description

On the OS/400, the centralized access control model is used. This means that the same user, MSrvAdm, is used as the owner of both the application and the database. Also the group MSrvAdms is used as the primary group to grant manager rights to the application and database for other users if needed. Figure 9-4 illustrates the scenario.
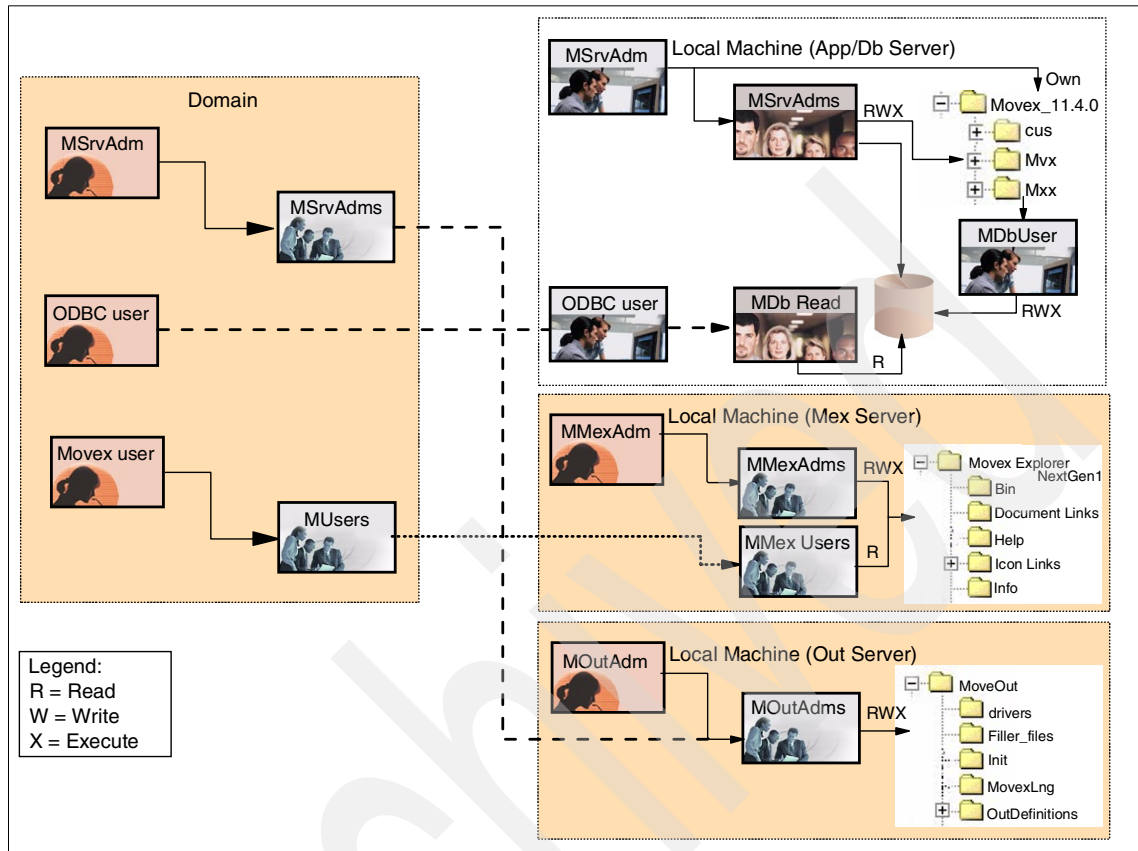
Figure 9-4   Access control scenario on OS/400

The main reason for selecting this setup for the OS/400 is the fact that the creator of objects in the integrated file system (IFS) where the application resides automatically becomes the owner of the objects. The IFS also supports the method of authority inheritance of the group settings. By creating the root directory and setting the authority right for it, all components installed in that directory automatically inherit the right authority settings.

This means that as long as the user profile MSrvAdm is used when performing all installation and maintenance, everything is created with the right authority. If another user, that either has the *ALLOBJ special right or belongs to the manager group MSrvAdms, is used, that user becomes the owner of the objects created. This is normally not a problem, since the authority is handled by inheritance from the overlying folder. To avoid possible problems with this, there are tools in the Utility library, MVXCJVA, that you may use to set the authorities correctly.

Another reason for the centralized model to be used on OS/400 is the fact that the DBMS (DB2 UDB for iSeries) is a fully integrated part of the operating system. This makes it possible to use exactly the same access control model for both the application and the database.

9.4.4 Users and groups

The users and groups used on the OS/400 are created automatically through the first phase of the installation. Tools are also available in the Utility library, MVXCJVA, to check the users and groups later.

Table 9-2 shows the different users and groups that are recommended for use, by platform.

Table 9-2 Recommended users and groups

	Type	OS/400	Windows	Solaris
MAppAdm	User		Y	Y
MDbAdm	User		Y	
MDbUsr	User	Y	Y	Y
MSrvAdm	User	Y	(Y)	
MAppAdms	Group		Y	Y
MDbAdms	Group		Y	
MDbReads	Group	Y	Y	Y
MSrvAdms	Group	Y	(Y)	
Y = Used (Y) = Optional				

Users

This section describes the users that are used for the access control. They are all created as described here during the LODRUN phase of an installation.

MSrvAdm: Movex Server Administrator

The user MSrvAdm is, in the centralized access control model, the owner of both the application and the database. The user can also be used in the decentralized access control model as a general management user, on the Windows domain level, that has manager rights through the two manager groups – MAppAdms and MDbAdms (Figure 9-5).
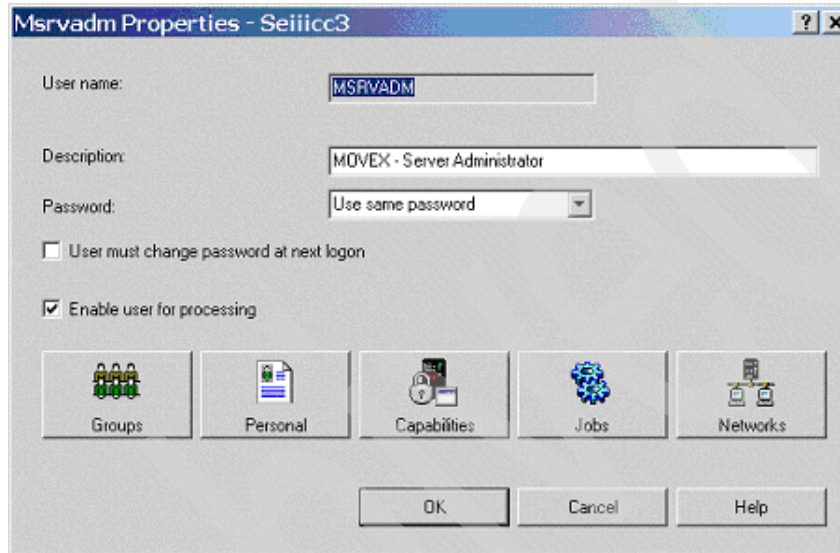
Figure 9-5   Msrvadm Properties

MSrvAdm is the user that normally should be used to perform installation and maintenance of the Movex Java on the OS/400 platform. If this user is used, no further actions are required to obtain the right authorities for the application or the database.

As shown in Figure 9-6, the user MSrvAdm belongs to the group MSrvAdms. This group is also set as the user primary group. All objects created by the user should be assigned All Access rights for the group.
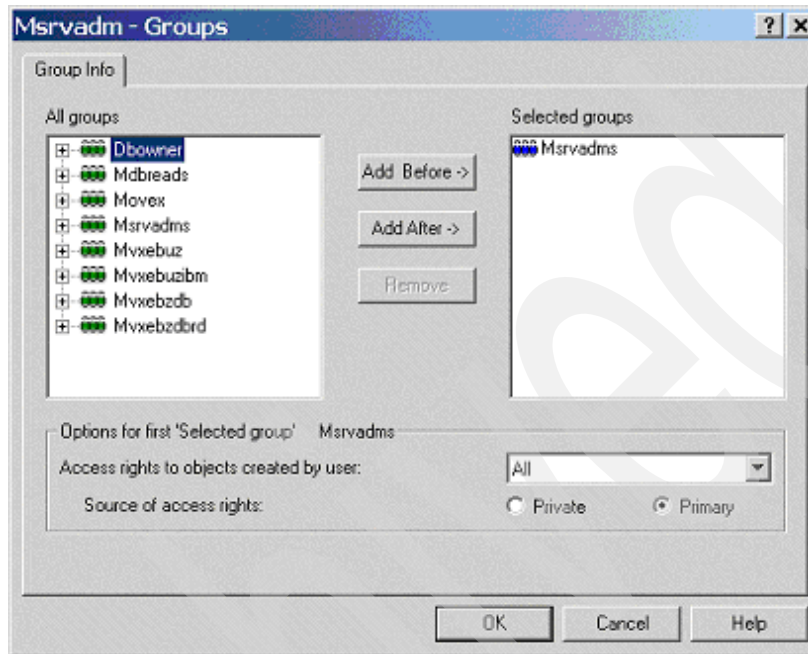
Figure 9-6   Msrvadm - Groups

Note: If an older version of Movex is installed on the same OS/400, the owner group of that installation, MOVEX, should not be assigned to the user MSrvAdm. This is because that group is defining a different access control model.

The user MSrvAdm is assigned the system privileges in Figure 9-7 as a minimum. If desired, you can also assign the system privilege Security Administration, but that is not necessary.
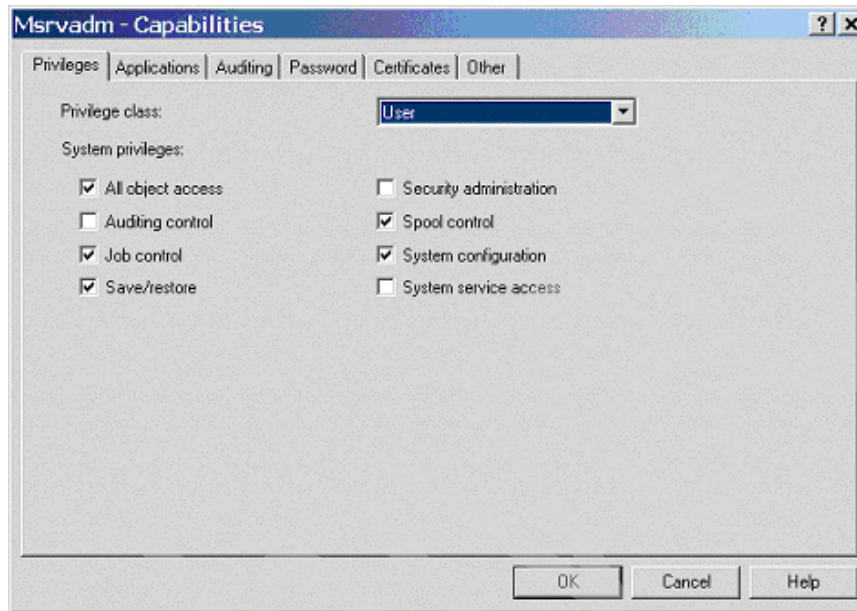
Figure 9-7   MSrvAdm - Capabilities

The user MSrvAdm has the Utility library, MVXCJVA, assigned as Current library (Figure 9-8).
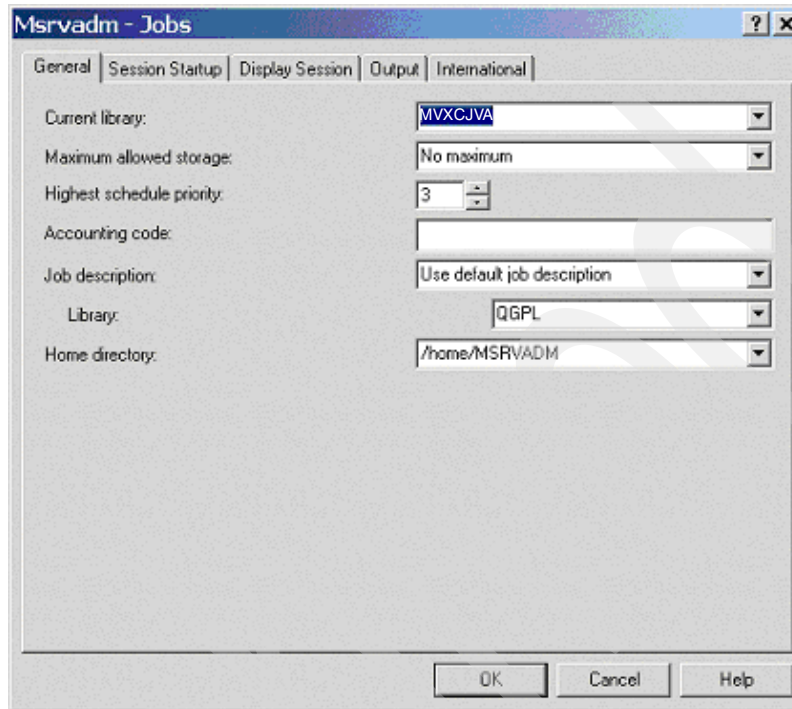
Figure 9-8   Msrvadm - Jobs: General page

The Initial menu for the user MSrvAdm is MVXSTART in the Utility library,
MVXCJVA (Figure 9-9).
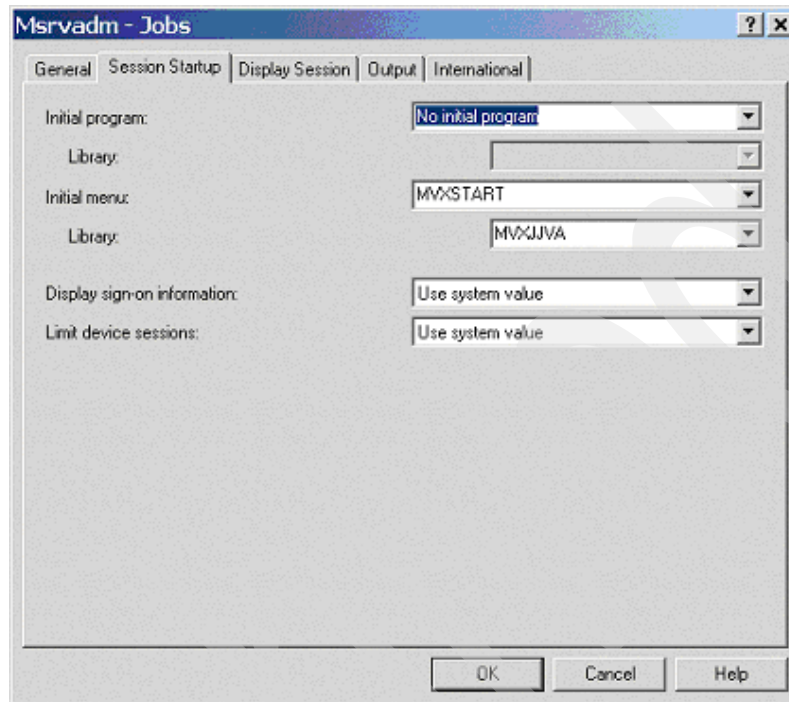
Figure 9-9   Msrvadm - Jobs: Session Startup page

MDbUsr: Movex Database User

The user MDbUsr is used, in both the centralized and the decentralized access
control model, to connect to the database from the application. The name of this
user and the password in use must be set up in the Movex.properties file to give
the application server the ability to connect to the database. The properties are:

dir.con.user   The name of the user profile that is used to connect to the
database.

dir.con.password   The password of the user profile used to connect to the
database.

If possible, according to the security policies in place for the installation, allow
this user to have a never expiring password. Otherwise routines must be in place
to change this manually on a regular basis.

The user MDbUsr is an internal user used to connect to the database from the application (Figure 9-10).
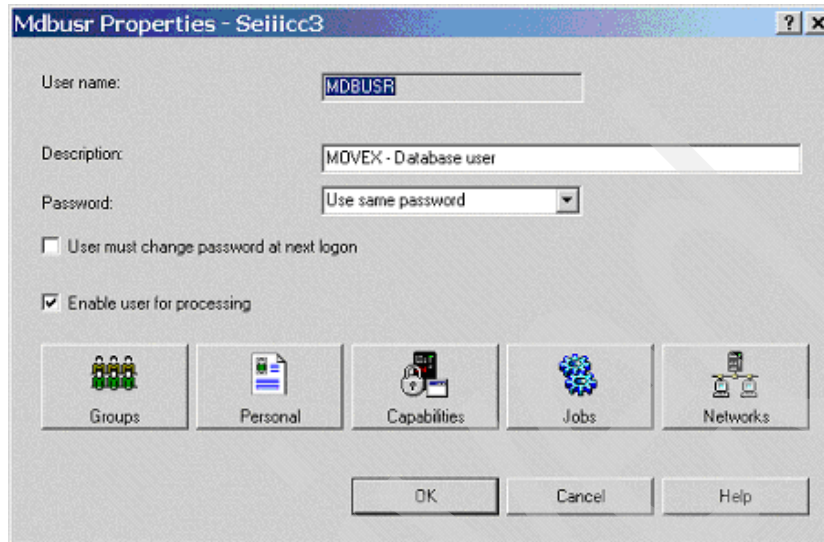
Figure 9-10   Mdbusr Properties

The database user MDbUsr does not belong to any groups (Figure 9-11).
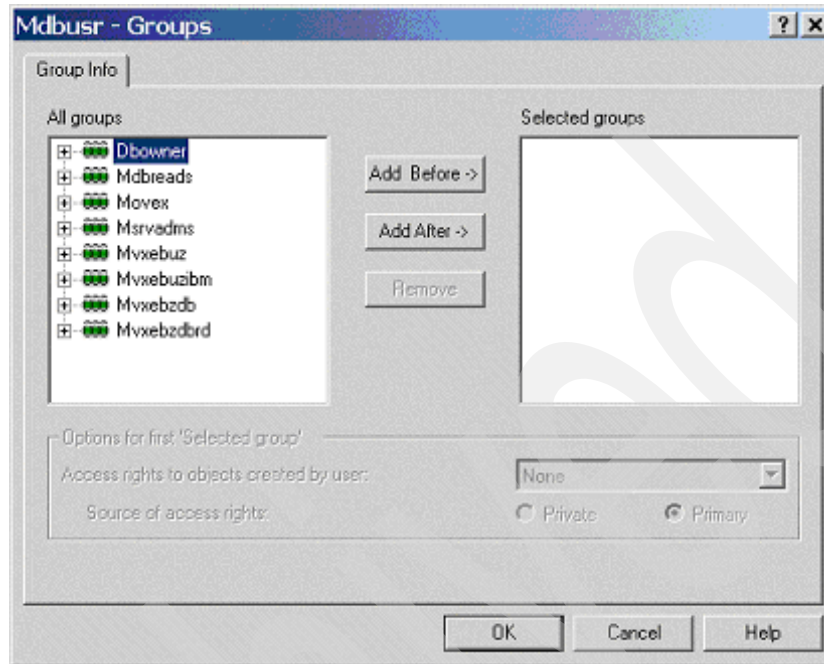
Figure 9-11   Mdbusr - Groups

The user MDbUsr has no special system privileges (Figure 9-12).
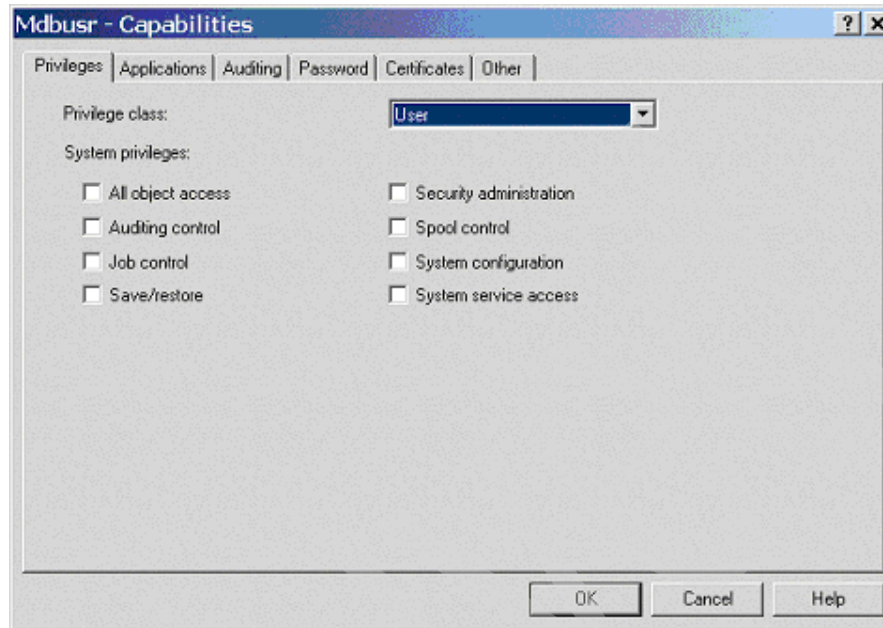
Figure 9-12   Mdbusr - Capabilities: Privileges page

Under Password expires, select **Never** for the user MDbUsr to avoid unforeseen stops in the processing (Figure 9-13). This is only if the security policy of the installation allows users to have this setting.
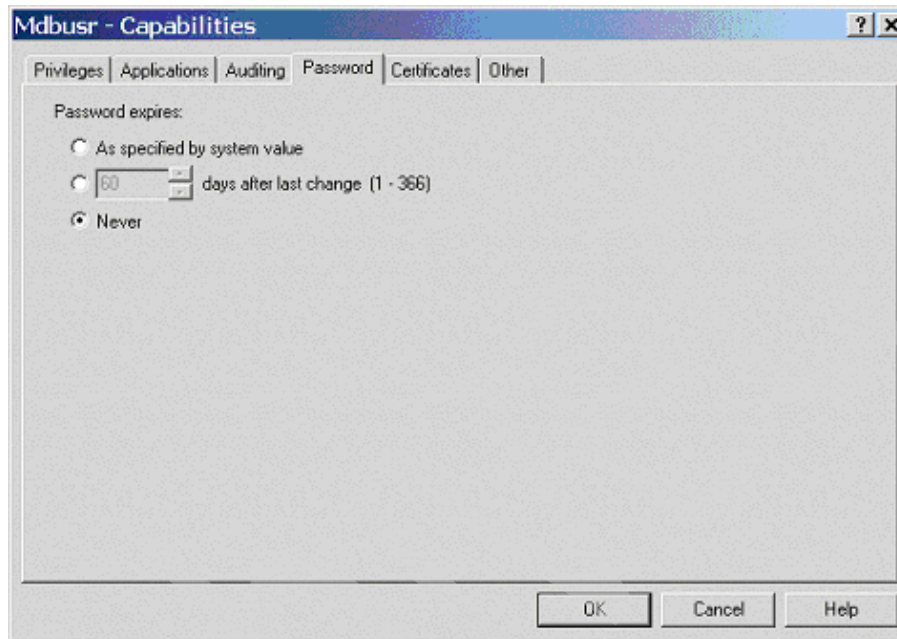
Figure 9-13   Mdbusr - Capabilities: Password page

Groups

This section describes the groups that are used for access control.

MSrvAdms: Movex Server Administrators group

The group MSrvAdms is used to give manager rights to both the application and the database in the centralized access control model. This group could also be used in the decentralized access control model, on Windows, to gather users on the domain level that should have manager rights through the two local groups MAppAdms and MDbAdms.

The user MSrvAdm is the only default user that belongs to the group MSrvAdms (Figure 9-14).
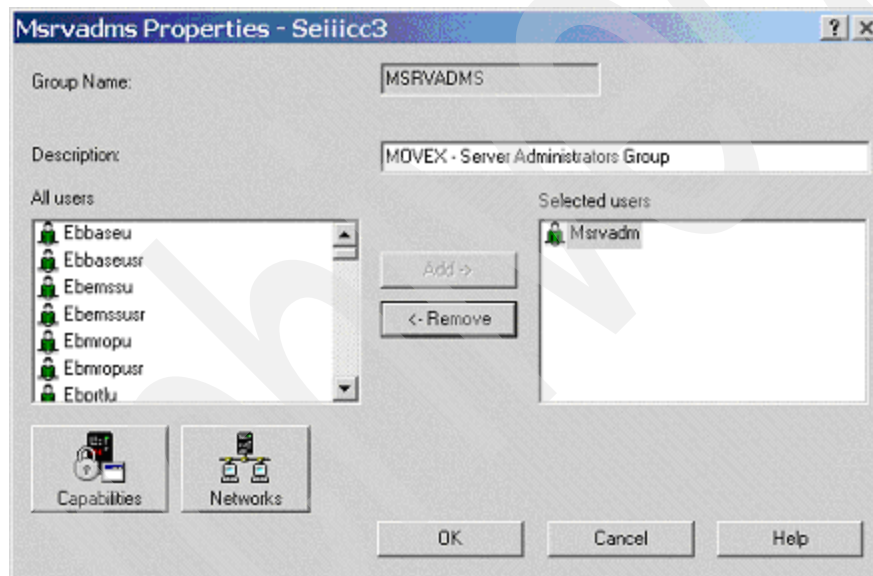
Figure 9-14   Msrvadms Properties

The manager group MSrvAdms does not have any special system privileges (Figure 9-15).
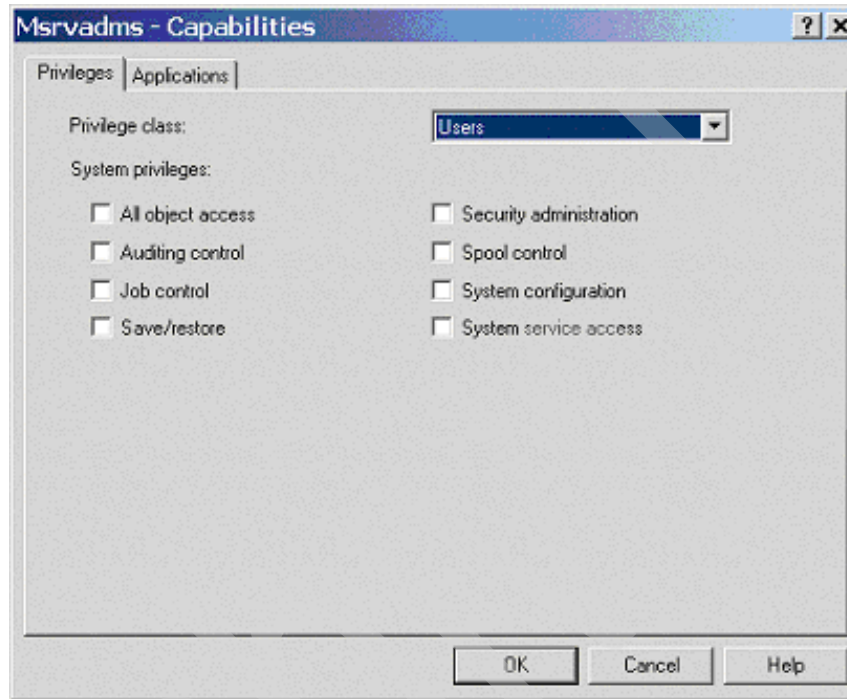
Figure 9-15   MSrvAdms - Capabilities

Note: Ordinary users that should be assigned manager rights through the group MSrvAdms must have this group set as primary group, in the same way as user MSrvAdm. This means that, if an older version of Movex is installed on the same iSeries server, the same user profile cannot be assigned manager rights to both versions. This is due to a conflict in the assignment of groups between the two different access control models that are in place for these versions.

MDbReads: Movex Database Access group
The group MDbReads is used to give external users read rights to the database in both the centralized and the decentralized access control model. The database readers group, MDbReads, does not belong to any groups (Figure 9-16).
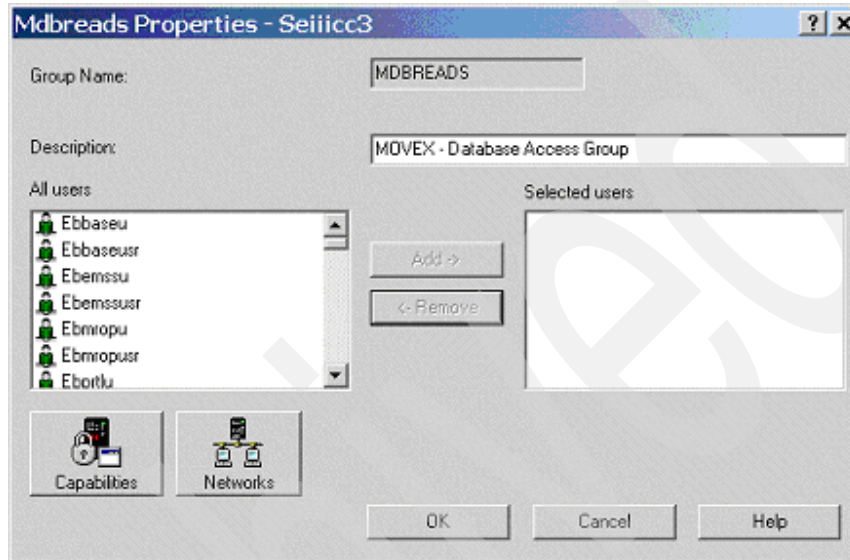
Figure 9-16   Mdbreads Properties

The group does not have any special system privileges (Figure 9-17).
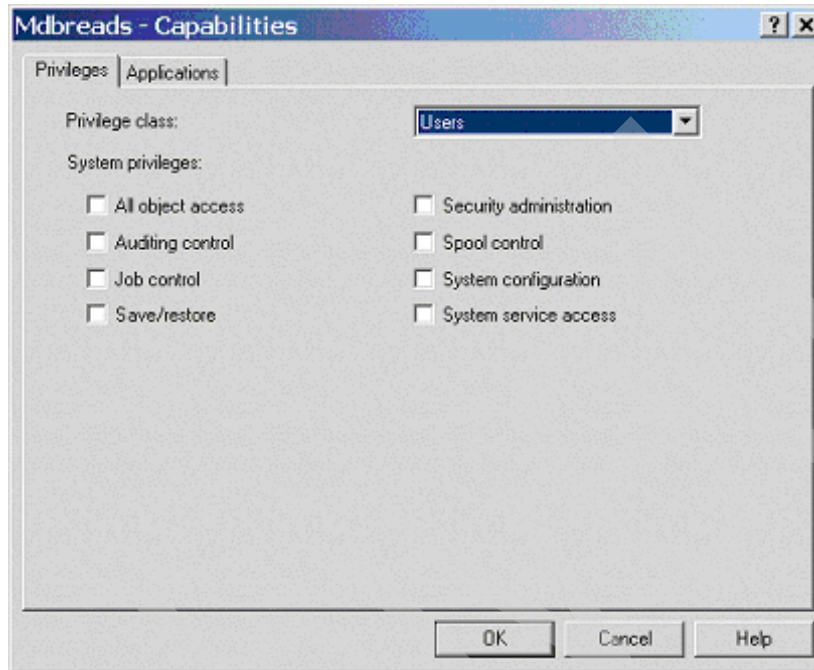
Figure 9-17   Mdbreads - Capabilities: Privileges page

9.4.5 Authority settings: Application

By setting the right authorities to the root folder of the installation, that authority is inherited by the lower level components when the installation is done. These settings are done automatically when the function to create the root folder, found in the Utility library MVXCJVA, is used.

The owner of the file structure is the MSrvAdm user, and the primary group is MSrvAdms. Both the owner and the primary group have full authorities to the structure, while the system default group Public is excluded from the structure (Figure 9-18).
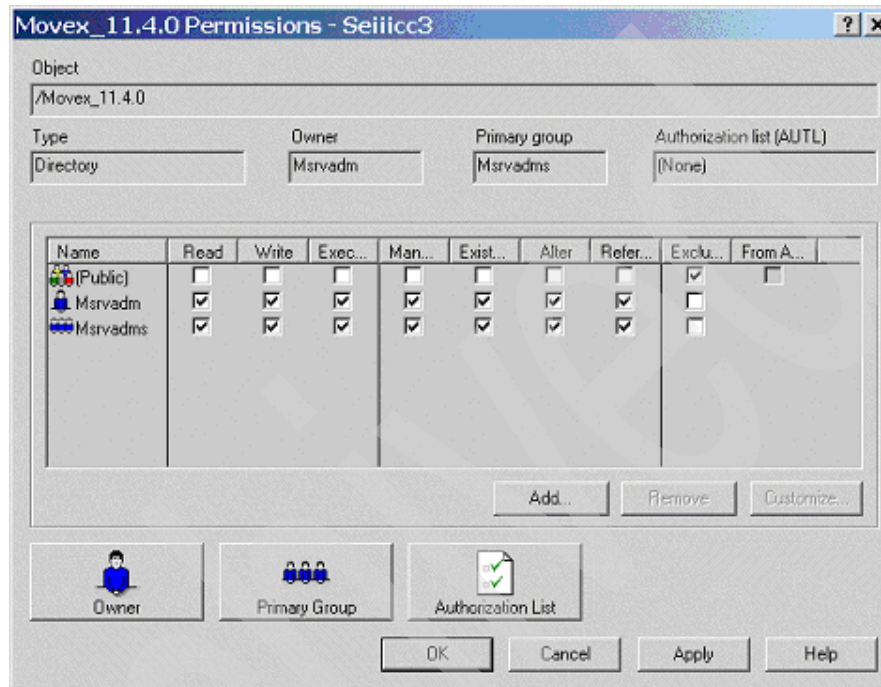
Figure 9-18   Application authority: Root folder

Also note that the MDbUsr user and the MDbReads group do not need any authority to the application.

By the inheritance mechanism, the assigned authorities go all the way down to the lowest level of the structure (Figure 9-19).
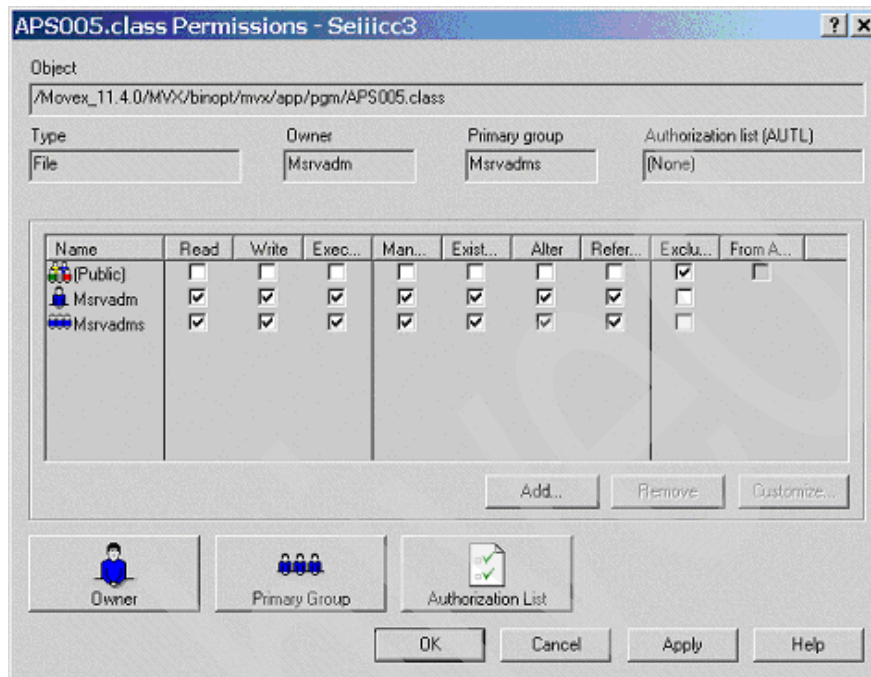
Figure 9-19   Application authority: Single class

9.4.6 Authority settings: Database

For the tables in the database, the settings looks a little more complicated. The owner, MSrvAdm, and the primary group, MSrvAdms, have All authority while the Movex database user, MDbUsr, has Change authority. The Movex database access group, MDbReads, has only Use authority because that group should be used to give external users the ability to read data in the database but not to update the data (Figure 9-20).
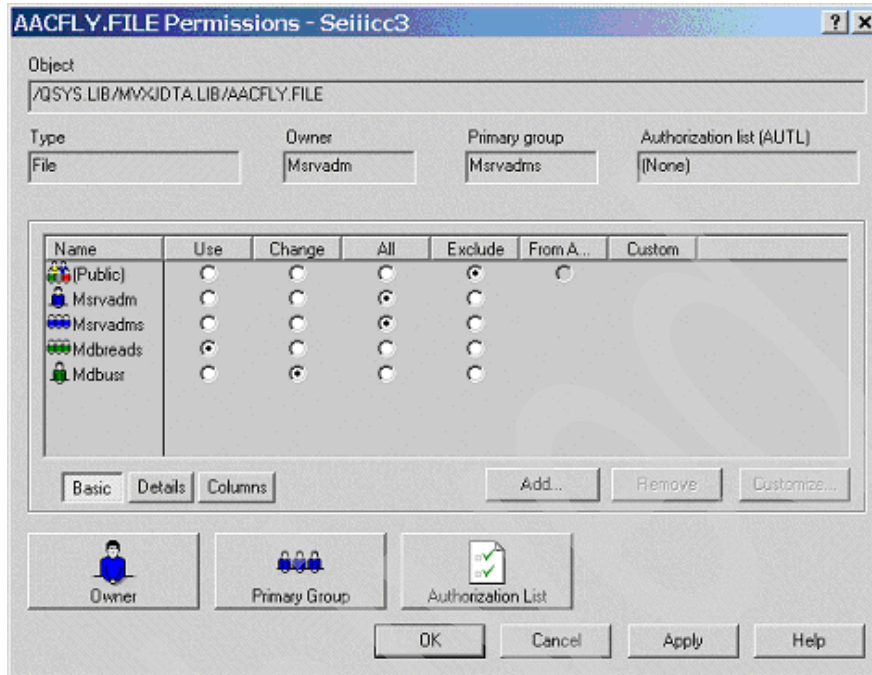
Figure 9-20   Database authority

These settings work for the major part of the tables in the database, but for tables created dynamically in runtime, the Change authority for MDbUsr is not enough. This is the fact for the tables that constitute the datasets, for example. To manage these tables, the user MDbUsr needs All authority. When these tables are created this happens automatically, because MDbUsr becomes the owner of the files. If the utility to set authority for the database found in the utility library, MVXCJVA, is used, it takes this into consideration.

Figure 9-21 shows the same information as Figure 9-20, but in the traditional 5250 way. The information shown may vary somewhat in how the groups are represented. This is because of the fact that users and groups in the OS/400 security context basically are the same thing.

```
                        Edit Object Authority

Object . . . . . . . :   MITMAS          Owner  . . . . . . . :   MSRVADM
  Library  . . . . . :    MVXCDTA        Primary group  . . . :   MSRVADMS
Object type  . . . . :    *FILE

Type changes to current authorities, press Enter.

  Object secured by authorization list  . . . . . . . . . . . .    *NONE


                         Object
User        Group       Authority
MSRVADM                 *ALL
*GROUP      MDBREADS    *USE
MDBUSR                  *CHANGE
*GROUP      MSRVADMS    *ALL
*PUBLIC                 *EXCLUDE




                                                                      Bottom
 F3=Exit   F5=Refresh   F6=Add new users   F10=Grant with reference object
 F11=Display detail object authorities     F12=Cancel   F17=Top   F18=Bottom
 (C) COPYRIGHT IBM CORP. 1980, 2000.
```

Figure 9-21   Edit Object Authority display

9.5 Movex authority system

This section describes the built-in authority system of the Movex application. The major functionality is the same in the Movex Java generation of the application as in previous generations, with some minor differences.

The Movex authority system is a programmed authority that operates at two different levels of the functions. It can operate inside a function or outside a function on a general level.

The programmed authority is built into some functional areas. It works inside single programs, taking into account the business data in Movex. For example, in the General Ledger, authority to certain accounts can be restricted to certain users. Or in Purchasing, the authority to accept purchase requisitions can be authorized at different monetary values. Functional-related security is not a concern here.

The general function authority is an overriding layer of Movex that applies to all functions. It is independent of the functional area. The Movex authority system is also independent of the underlying OS/400 security mechanisms. They are used to define access control security, which is discussed in 9.4, "Access control" on page 126. This section deals with general function authority.

Figure 9-22 shows an overview of Movex authority.
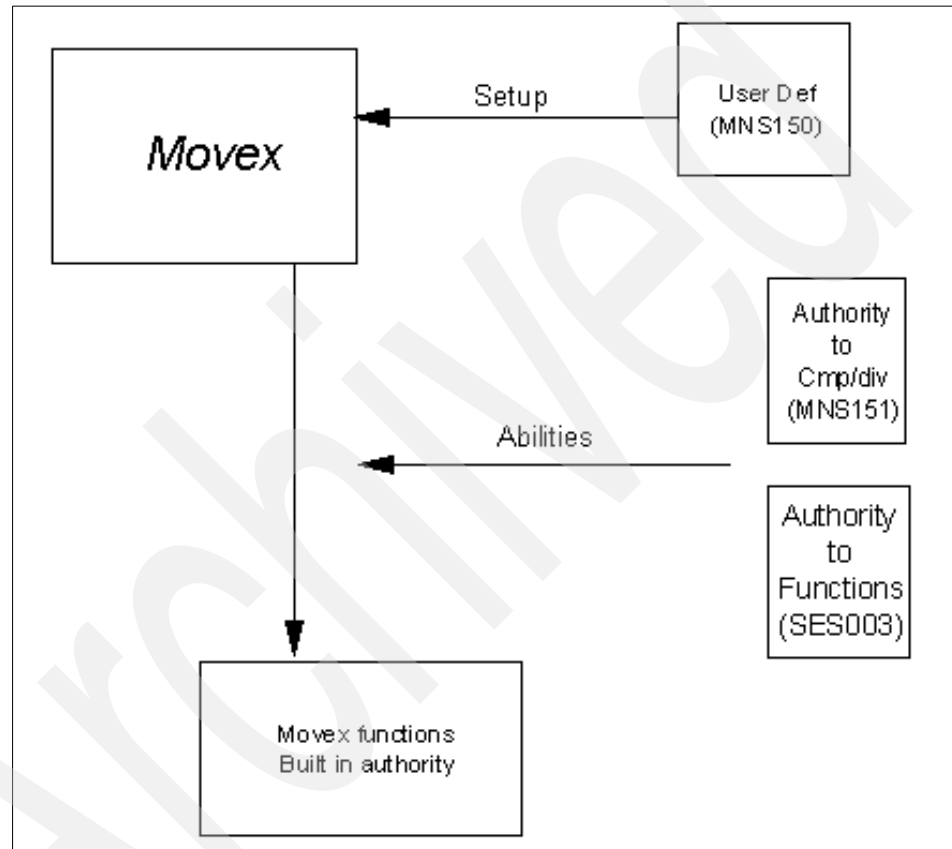
Figure 9-22   Movex authority overview

Many different components constitute the general function authority system of Movex. The most central one is the Movex user definition, which is handled within the program 'User. Open' (MNS150). With this definition as the basis, more detailed information is maintained to define to which companies and divisions a specific user is authorized, and to specify what authority the user has to specific

functions within a specific company and division. On top of this, the definitions may be made using different grouping levels to ease maintenance. Intentia recommends that you create a plan for the security structure before you start entering the definitions.

Another part of the authority mechanisms, that is often forgotten, is the ability to create unique menus for a user or a group of users and then lock these users to these menus. This is normally the easiest way to get started with the setup of the authority system.

9.5.1 Movex user definition

This section describes the programs and methods used for user definition in Movex.

'User. Open' (MNS150)

To use the Movex system, each user must be defined in the program 'User. Open' (MNS150). This program contains all the environmental information that is assigned to each user to set up the correct environment when the user starts a new session. For some of the values entered, the user then has the ability to override the settings within the session.

The information to enter in 'User.Open' (MNS150) is for the moment larger than necessary for Movex Java, because this program is still synchronized with older generations of the Movex system. This may change in a future version of Movex.

The first detail panel defines the user to the system. It includes the start values the user should have for such things as company and division, language, date format, and others (Figure 9-23).

Figure 9-23   User. Open' (MNS150), panel E

On the second panel of the program (Figure 9-24), only the following fields are of interest for Movex Java:

Field	Description
Start menu	The name of the start menu assigned to a user must correspond to a menu file (.tre file) that must exist in the Movex Explorer installation. If that is the case, this menu is shown to the user.
Menu lock	In Movex Java, only Menu lock = 0 has meaning. This setting means that the user does not have access to the Movex menu tab. At the same time, if no start menu is stated, or the start menu stated does not exist as a menu file, the user is only presented the Process and Favorites menu tabs. Menu lock = 0 also disables the Run command in Movex Explorer for this user.
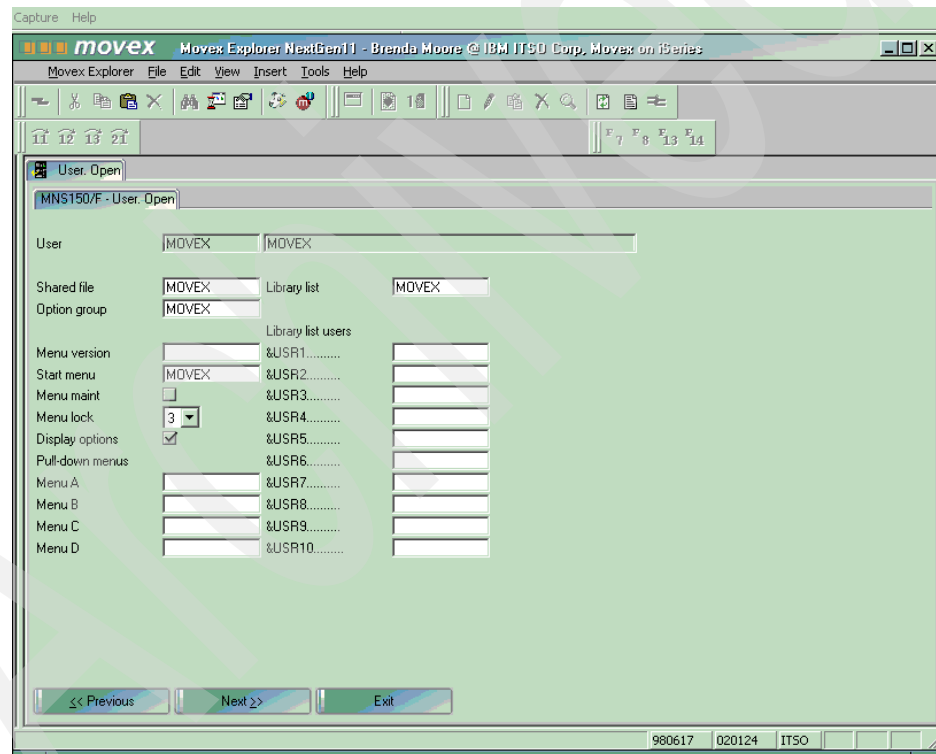
Figure 9-24   'User. Open' (MNS150), panel F

The information on this panel is not used in Movex Java (Figure 9-25).
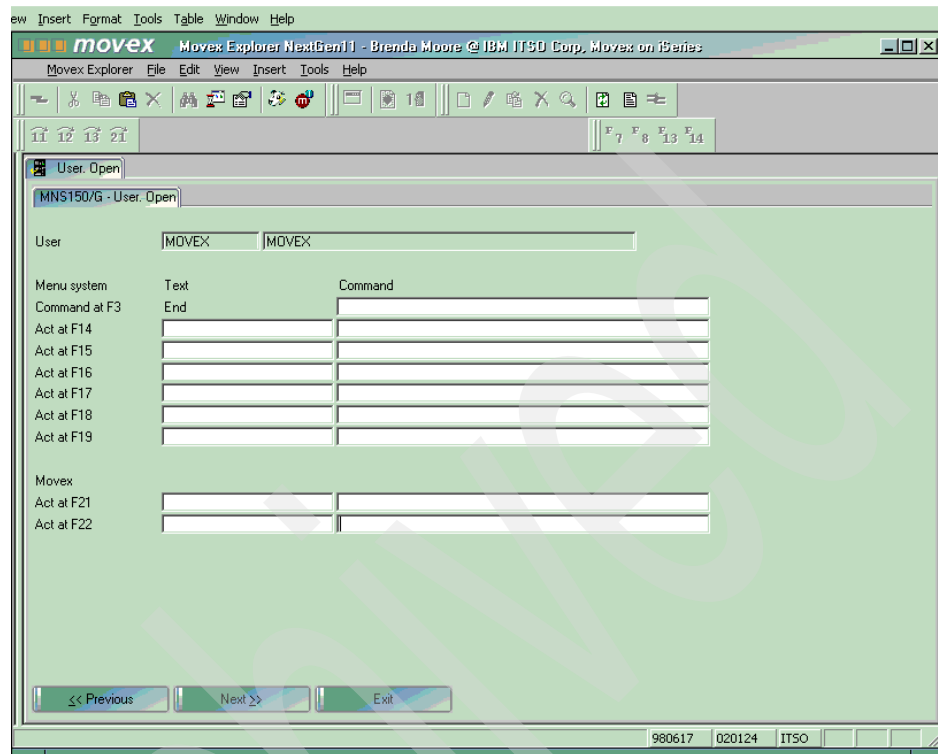
Figure 9-25 'User. Open' (MNS150), panel G

Grouping users

Users may be grouped to ease maintenance of the authority settings. One user may belong to no group or to one group. This information is entered in the field 'User grp func' on panel E in 'User. Open' (MNS150). The group name entered must first be defined in the same program by entering a user that on the E panel is specified as 'User type' *GRPPRF.

'Authorization. Specify for Company' (MNS151)

To gain access to the Movex application, the users must not only be defined within 'User. Open' (MNS150), but they must also be authorized to use the companies and divisions in question. This authority is specified in the function 'Authorization. Specify for Company' (MNS151) that may be reached by selecting 'User ID' under Open related on the overview panel of the program 'User. Open' (MNS150). Select the proper record in the overview panel and right-click to find the Open related menu.

All valid combinations of company and division per user are displayed in the panel in Figure 9-26.
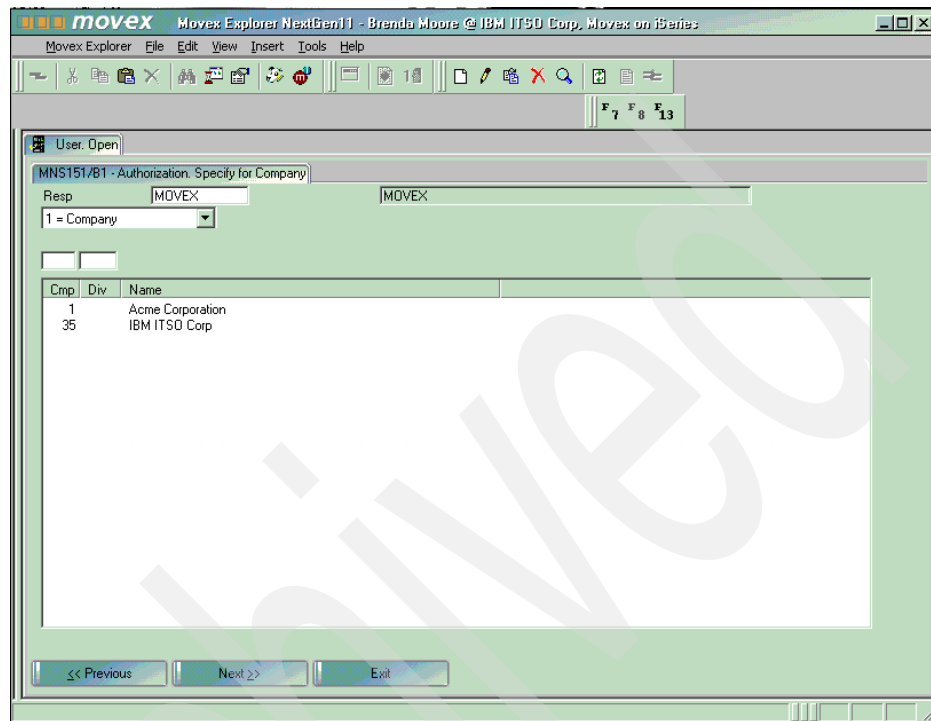
Figure 9-26   'Authorization. Specify for Company' (MNS151), panel B

There is also the ability to assign a user as authorized to all existing companies and divisions by selecting 'Upd user id' under Open related on the overview panel of 'User. Open' (MNS150).

Notes:

► This mass update is only done for existing companies and divisions. If a new company or division is entered afterwards, all users that should be authorized to that combination must be updated.

► This mass update function should be used with care. The case may be that not all users should be authorized to all the existing companies or divisions.

Information specified in 'User. Open' (MNS150) may be overridden by information specified in 'Authorization. Specify for Company' (MNS151) for the specific combination of user, company, and division. This could be helpful if one division is operating under a different date format or another language, for example (Figure 9-27).

Figure 9-27 'Authorization. Specify for Company' (MNS151), panel E

Tip: Be aware of the difference between creating a new user in 'User. Open' (MNS150) by using New record or Copy record. If New record is used, the new user is automatically authorized to all existing companies and divisions. If Copy record is used, the new user inherits the authorization from the user that was copied. This means that it is convenient to have some template users, with the right authority set, entered in this program and then use Copy record to add new users to the system.

9.5.2 Movex general function authority

Movex general function authority works by making an association between a function and a user, and specifying whether the combination is allowed. The general function authority is maintained by 'Function. Connect Authority' (SES003). You can simplify maintenance of the general function authority by using function and user groups.

Restricting a user's access to a specific menu by setting a specially designed start menu in 'User. Open' (MNS150) does not implicitly restrict access to other functions in the system. For greater power, combine the Start menu information with Menu lock = 0, which disables the Run command in Movex Explorer. If not, you must define all authority settings in 'Function. Connect Authority' (SES003).

Note: The user MVXSECOFR bypasses all Movex security. This can be useful if you by mistake "lock yourself out" of 'Function. Connect Authority' (SES003).

In 'Function. Connect Authority' (SES003), you see a list of the entries already made. At delivery time, there are no entries in this list (Figure 9-28).
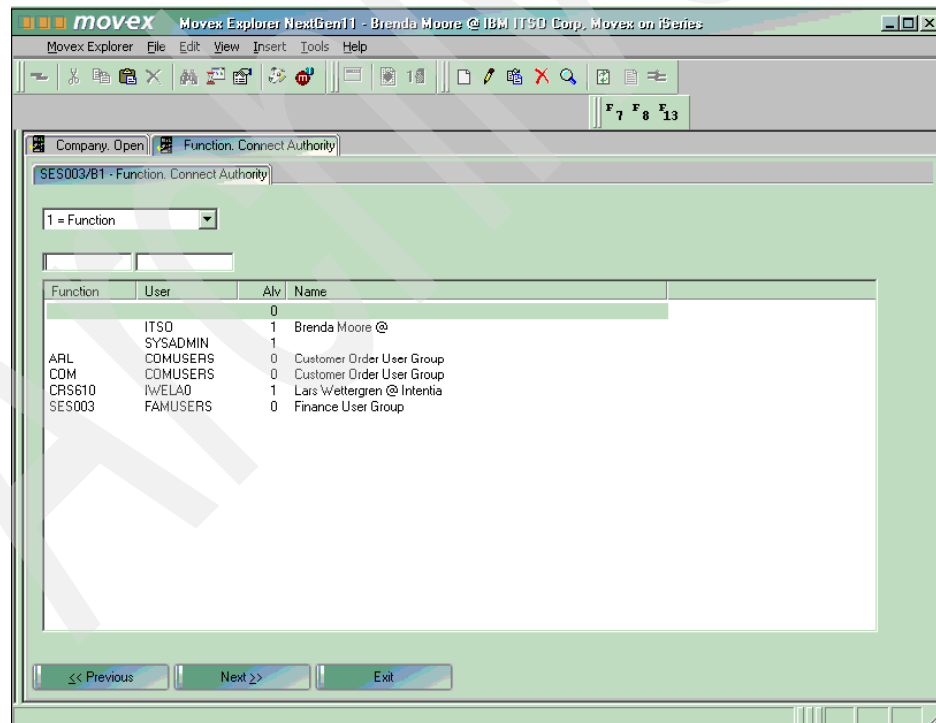
Figure 9-28   Function. Connect Authority' (SES003), panel B

> **Note:** The entries can either disallow a function or allow a function. A value of 1 in the Alv column means that the function is allowed. A value of 0 indicates that it is not allowed for this specific user.

When working at a central level (division Blank), you can view individual divisions' entries using the Division field. This does not apply when working at the division level.

On the E panel (Figure 9-29), you define the specific authority to each combination of function and user by setting the Funct permitted field to 1 or 0:

1   The user is permitted to use the function.
0   The user is not permitted to use the function.
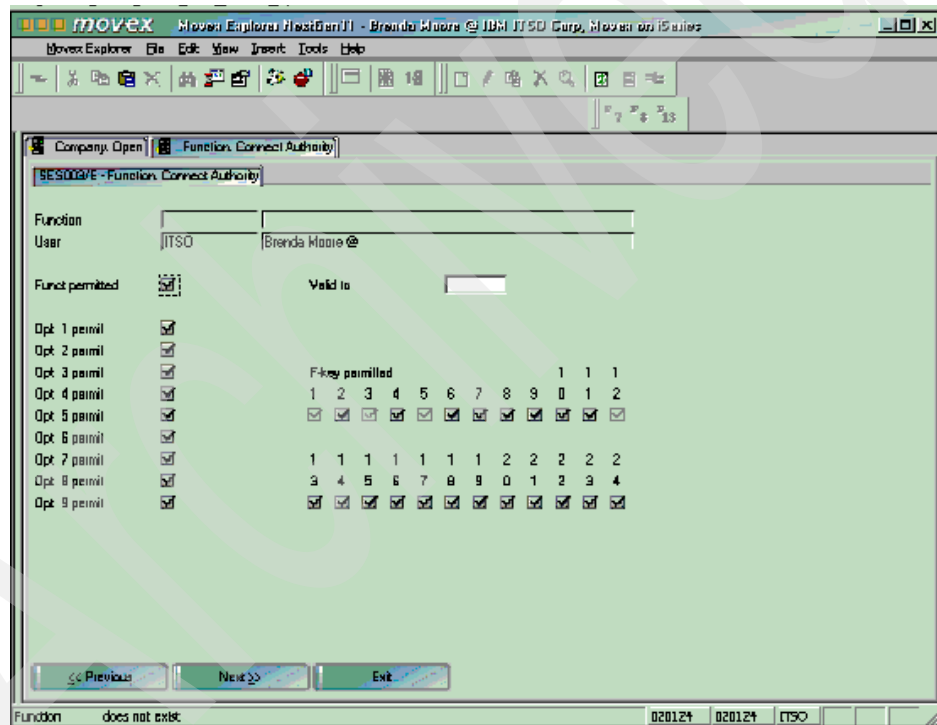
Figure 9-29   'Function. Connect Authority' (SES003), panel E

If you set the Funct permitted field to 1, you can set the standard options (1 to 9) to be allowed or disallowed, or use function keys F1 or F24.

The Valid to date field gives the entry a limited time span. After the date has passed, Movex ignores the entry.

Relationship between company and division

When you are maintaining general function authority, your entries apply only to the company or division in which you are currently working. To maintain another company or division's authorities, you must first switch to that company or division.

Still there is a referential link between a company and its division in the Movex authority system. Entries made at the company level (division blank) also apply to any divisions in that company that have no security entries of their own. This can be thought of as the divisions that have no need for their own security, and therefore, adhere to company policy.

In this example, as illustrated in Figure 9-30, Company 1 central has some security entries in 'Function. Connect Authority' (SES003). Division A in company 1 has no security entries of its own, and therefore, uses the security entries of its parent company. That is, it adheres to company policy.
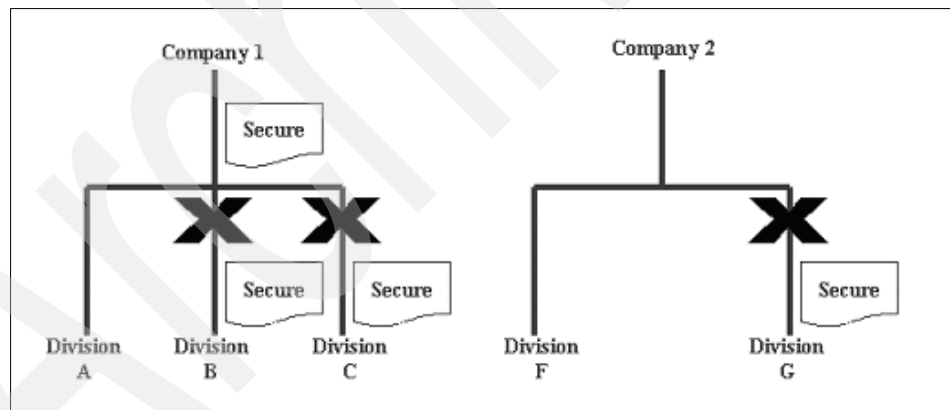
Figure 9-30   Relationship between company and division security

Divisions B and C in company 1 have entries of their own, so the reference back to the parent company is broken. They do not refer to company-level entries at all, even for users or functions for which they do not have entries. If a division has any security of its own, even a single record in 'Function. Connect Authority' (SES003), it breaks the reference to company security.

Company 2 Central and division F in company 2 do not have security. Division G in company 2 has security entries of its own. These divisional restrictions are not seen at the company level. The security in the division does not mean Company 2 Central is secured.

Movex authority search priorities

Since it is possible to mix individual entries and group entries, it may be of interest to know how Movex resolves requests from users to run functions. The principle is that specific entries are always searched before generic ones are searched.

Table 9-3 shows the possible combinations of user and function, individual and group, and the order in which Movex searches for authority records. Whenever a match is found, the search stops.

Table 9-3   Authority search priorities

	User	**Function**
Level 1	Individual user ID	Individual function ID
Level 2	Individual user ID	Function group
Level 3	User group	Individual function ID
Level 4	User group	Function group
Level 5	'blank'	Individual function ID
Level 6	'blank'	Function group
Level 7	Individual user ID	'blank'
Level 8	User group	'blank'
Level 9	'blank'	'blank'

The 'blank' user or function denotes no entry, which means a global group.

Backup and recovery

This chapter covers the basic concepts of Movex Java backup and recovery on an iSeries server.

159

10.1 iSeries backup and recovery overview

Backup and recovery options for Movex Java on an iSeries are managed by using *native OS/400 facilities*. OS/400 provides many facilities to save and restore information on the iSeries server. You can select these from menu options or use them in user-written programs, depending on the backup requirements and your operational environment.

Backup and Recovery Media Services/400 (BRMS/400) provides automated backup and archival operations, and tape management services. For additional information on BRMS/400, refer to *Backup Recovery and Media Services for iSeries*, SC41-5345.

This section discusses some considerations for backup and recovery and includes some example backup programs for reference. This section does not develop a full backup and recovery strategy. Nor does it cover all save and restore scenarios, because each one is unique to the save strategy or reason for restore. For more information, see *Backup and Recovery*, SC41-5304.

10.2 Backup types

The best and easiest way to ensure that your backup is complete is to make a *cold backup*. That means that you end Movex Java, and there is no other activity on your system during the backup.

You can use the save-while-active functions. However, to be sure that the backup is consistent, you still have to end Movex Java for a short while when a checkpoint is created in the database. Then you can start Movex Java again.

If you do not end Movex Java while you create the checkpoint, you will experience problems in determining where you should start in a recovery situation.

You can also use high availability functions to make your downtime even shorter. Then you will have a second iSeries server with replicated or mirrored data where you can make your backup when convenient.

10.2.1 Cold backup

You can perform a cold backup in many different ways. This section discusses the manual approach. An alternative is to use the automated approach, using a CL program.

To perform a cold backup manually, follow these steps:

1. Verify that no users or batch jobs are active in Movex Java. End Movex Java from the Server View or directly from the WRKACTJOB panel. Wait until Movex Java has ended.

2. Save the Movex Java libraries to tape with the SAVLIB command:

   ```
   SAVLIB LIB(MVXxDTA) DEV(TAPxx) ENDOPT(*LEAVE) ACCPTH(*YES)
   ```

3. Save the integrated file server (IFS) tree where the Java objects reside with the SAV command:

   ```
   SAV DEV('QSYS.LIB/TAPxx.DEVD') OBJ(('/object-path-name')) ENDOPT(*LEAVE)
   ```

4. Restart Movex Java with the start program.

10.2.2 Save while active

The save-while-active (SWA) function on the iSeries server is an option on several of the save commands. It allows you to use the system during all or part of the backup process, reducing the planned outage time while backups are being taken. It allows you to modify objects most of the time they are being saved. In contrast, other save functions provided on the system allow no access to the objects as they are being saved or only allow the objects to be read as they are being saved.

There are two ways in which SWA can be used. Each is discussed in the following sections.

True save-while-active (advanced) method

This method involves actually saving the objects as they are updated. Using this method, the time-relationship between objects is not maintained. For example, object A in library A may be saved first, and then some time later, object Z in library Z is saved.

During the time between A and Z being saved, object Z is updated, losing the integrity between the two objects. To recover from such a save and maintain the integrity of the database, it is necessary that, at a minimum, journaling is running when the save is taken and preferably commitment control is set.

In the event of restoring from the backup, the objects would initially be restored *exactly* as they had been *saved* (including the time differences between objects A and Z being saved). Then you must back out the updates to a known point of integrity, using the journal. Clearly this can be a difficult and time consuming task. If commitment control is included in the application design, it can be easier so that changes can be backed out to a commitment boundary.

Save after checkpoint (recommended) method

The recommended way to use SWA is to hold the system so that no updates are done to the database until a checkpoint is established. Then, once the checkpoint is reached, you can safely allow users back on the system to continue making updates to the database while it is being saved to tape.

The system performs the save-while-active function by maintaining an image of the object being saved as it existed at one time. As the object is changed by an application during the save, the system maintains an original copy of the pages of the objects that are being changed.

In virtual storage, a page is a fixed-length block that has a virtual address and is transferred between main storage and auxiliary storage. It can be viewed as if the system is maintaining two images of an object as it is being saved:

► An image that contains the updates to the object that normal system activity works with

► A second image is an image of the object from a single point in time, that only the save-while-active job is using to save the object to the media

The system does not maintain two complete images of the object being saved. It only maintains two images for the pages of the objects that are being changed as the save is being performed.

Follow these steps to perform a save after checkpoint backup manually:

1. Verify that no users or batch jobs are active in Movex Java. End Movex Java from the Server View or from directly from the WRKACTJOB panel. Wait until Movex Java has ended.

2. Save the Movex environment in the IFS tree with the SAV command:

   ```
   SAV DEV('QSYS.LIB/TAPxx.DEVD') OBJ(('/env/xxxx')) ENDOPT(*LEAVE)
   ```

3. Save the database by running the SAVLIB command with SAW options:

   ```
   SAVLIB LIB(MVXxDTAxx) DEV(TAPxx) ENDOPT(*UNLOAD) SAVACT(*SYNCLIB)
   SAVACTWAIT (*NOMAX) SAVACTMSGQ(LIBxx/MSGQxx) ACCPTH(*YES)
   ```

4. When the checkpoint is reached, you receive a message to the message queue you defined in the SAVLIB command. Then you can start the Movex Java environment again.

10.3 Backup schedule

When you decide the backup routines in a project, follow the technical Implex recommendations. This activity takes place in the Configuration phase of Implex. See the Implex *Operational Handbook*, which is available from the Intentia Wire.

Consider these recommendations:

► **Daily backup**: For changes only, SAVCHGOBJ. A typical setting for libraries containing changed data for a Movex Java version installation is MVX*v*DTA (where *v* is the version, for example, A, B, etc.).

► **Weekly backup**: For comprehensive backup except for SAVSYS, operating system, and IBM installed software

► **Quarterly backup**: For a full system backup

Table 10-1 shows what you need to save in a backup.

Table 10-1   Save contents

Object	Frequency
MVXJDTAxxx – standard DB HUMJDTAxxx – HUM DB MMAJDTAxxx – market DB MVXJJRxxx – journal library MRPSIMYY – MRP simulation	Daily
MVXJJVA	At least weekly (and before and after new SP installation)
Environment objects in IFS (refer to Figure 10-1)	Daily
IFS objects (refer to Figure 10-1)	At least weekly (and before and after new SP installation)
J = Movex Java version, MA = Market, YY = simulation version 01 to 20	

Figure 10-1 shows an example of Movex objects in the IFS.
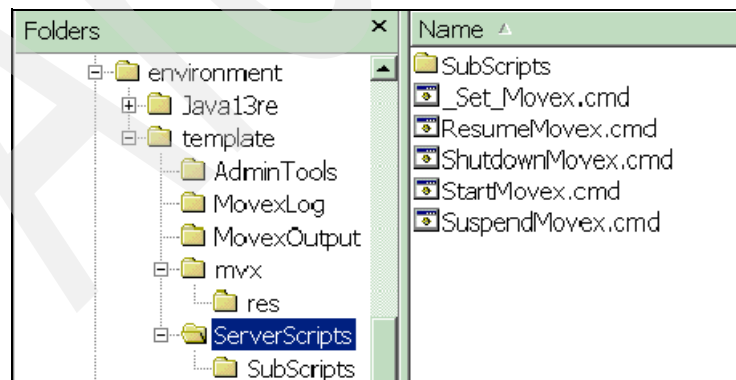
Figure 10-1   Movex objects in IFS -12.4.0

10.4 Recovery of objects

For the system to automatically re-establish your journaling environment, restore the objects in this sequence:

1. Journals
2. Based-on physical files
3. Dependent logical files
4. Journal receivers

When these objects are in the same library, the system restores them in the correct sequence. When these objects are in different libraries, you must restore them in the correct sequence, or you must manually re-establish your journaling environment after the restore operation.

If the entire system is not lost, you may need to simply restore an individual library or file that was inadvertently deleted or has become damaged or corrupt in some way. You can accomplish this by using the RSTLIB, RSTOBJ, or RST command for IFS objects. Sometimes it is easier to delete the object that you want to replace before you overlay it.

Since it is impossible to predict the different scenarios, refer to *Backup and Recovery,* SC41-5304.

10.5 Recovery of journaled objects using journaled changes

You can recover from many types of damage to journaled objects by using journaled changes. For example, an object is damaged and becomes unusable, an error in an application program caused records to be improperly updated, or incorrect data was used to update an object.

In each of these instances, simply restoring a saved version of the object may result in the loss of a significant amount of data. If you use the Apply Journaled Changes (APYJRNCHG) command to apply journaled changes, significantly less data may be lost.

10.6 Recovery after abnormal system end

If the system abnormally ends while you are journaling objects, the system performs the following actions:

1. Brings all journals, journal receivers, and objects that you are journaling to a usable and predictable condition during the IPL. This includes any access

paths that are being journaled and that are in use at the time the system abnormally ended.

2. Checks all recently recorded entries in the journal receivers that were attached to a journal.

3. Places an entry in the journal to indicate that an abnormal system end occurred. When the system completes the IPL, all entries are available for processing.

4. Checks that the journal receivers attached to journals can be used for normal processing of the journal entries. If some of the objects you are journaling cannot be synchronized with the journal, the system sends a message (CPF3172) to the history log (QHST) that identifies the journals that could not be synchronized.

   If a journal or a journal receiver is damaged, the system sends a message to the history log identifying the damage that occurred. CPF3171 indicates that the journal is damaged, and the message CPF3173 or CPF3174 indicates that the journal receiver is damaged.

5. Recovers each object that was in use at the time the system ended abnormally, using the normal system recovery procedures for objects. In addition, if an object being journaled was opened for output, update, or delete operations, the system performs the following functions so changes to that object are not lost:

   a. Ensures that the changes appear in the object. Changes that do not appear in the journal receiver are not in the object.

   b. Places an entry in the journal receiver that indicates whether the object was synchronized with the journal. For database files, if the file cannot be synchronized with the journal, the system places a message (CPF3175) in the history log identifying the failure, and you must correct the problem.

      For other journaled objects, the system places message CPF700C in the history log identifying the failure, and you must correct the problem. A synchronization failure can occur if the data portion of the object is damaged, a journal receiver required to perform the synchronization is damaged, or the journal is inoperable.

10.7 Procedure for abnormal system end recovery

Follow these steps after an abnormal system end:

1. Perform a manual IPL.

2. Check the history log to determine if there are any damaged object, objects that are not synchronized, or any damaged journals or journal receivers.

3. If necessary, recover the damaged journals or journal receivers as described in "Recovering when a journal is damaged" and "Recovering when a journal receiver is damaged" in *Backup and Recovery*, SC41-5304.

4. If there is a damaged object:

   a. Delete the object.

   b. Restore the object from the latest saved version.

   c. Allocate the object so no one else can access it.

   d. Restore the needed journal receivers, if they are not online.

   e. Use the Apply Journal Changes (APYJRNCHG) command to apply the changes to the object.

   f. Deallocate the object.

5. If an object cannot be synchronized, use the information in the history log and in the journal to determine why the object cannot be synchronized and how to proceed with recovery. For example, you may need to use the DFU or a user-written program to bring a database file to a usable condition.

6. Determine which applications or programs were active, and determine where to restart the applications from the information in the history log and in the journal.

If a journaled access path is in use during an abnormal system end, that access path does not appear on the Edit Rebuild Access Path display. If the maintenance for the access path is immediate or delayed, the system automatically recovers the access path during IPL.

A status message appears for each access path whose maintenance is immediate or delayed as it is being recovered during an IPL. The system places a message (CPF3123) in the system history log for each access path that is recovered through the journal during the IPL. This message appears for access paths that are explicitly journaled and for access paths that are protected by SMAPP.

10.8 Recovering when a journal is damaged

If a journal becomes damaged, the system sends message CPF8135 to the system operator and to the job log. Use the Work with Journal (WRKJRN) command to help you recover a damaged journal. Select option 6 (Recover damaged journal) on the Work with Journals display to recover a damaged journal.

For a description of the Work with Journals display, see "Work with Journal (WRKJRN) command options" in *Backup and Recovery*, SC41-5304, or the WRKJRN command in the online command help. To view the help, type `WRKJRN` on a command line, and press F1. Use the Work with Journals (WRKJRN) command to recover a damaged journal if you are only journaling physical files and access paths to this journal. The WRKJRN command performs all the following steps, except for saving the physical files and logical files. The WRKJRN command associates the receivers with the recovered journals without having to delete and restore the receivers.

Complete the following steps to recover a damaged journal without using the WRKJRN command:

1. Use the ENDJRNAP command (for the following sequence) to end journaling for:
   a. All access paths associated with the journal
   b. All physical files associated with the journal
   c. All IFS objects
   d. All other object types

2. Use the DLTJRN command to delete the damaged journal.

3. Run the Create Journal Receiver (CRTJRNRCV) command. Create a journal with the same name and in the same library as the damaged journal or restore the journal from a previously saved version.

4. Enter the Start Journal Physical File (STRJRNPF) command for the physical files that were journaled.

5. Use the Start Journal Access Path (STRJRNAP) command for access paths that were journaled.

6. Type the Start Journal (STRJRN) command to journal IFS objects.

7. Use the Start Journal Object (STRJRNOBJ) command to journal other object types.

   Note: You can restore your journaling environment by deleting and restoring all the objects that were being journaled. Objects that were journaled, at the time of their save, automatically begin journaling at restore time if the journal is online.

8. Save the journaled objects to allow for later recovery.

9. Associate the old journal receivers with the new journal. Follow these steps:
   a. Type `WRKJRN` and press Enter.
   b. On the prompt display, enter the name of the journal.

c. From the Work with Journal display, select option 9 (Associate receivers).

d. Press F12 to cancel the display.

e. Type the following command:

```
WRKJRNA JRN(library-name /journal-name)
```

Press Enter.

f. From the Work with Journal Attributes display, press F15 to display the receiver directory. If the journal receivers are not reassociated correctly, perform the following steps. These steps are usually required only if you have journal receivers that were created before V3R1.

i. Save the journal receiver that was attached to the damaged journal.

ii. Delete it and restore it and any previously attached journal receiver you need. You must delete and then restore the receiver after the journal is restored or recreated to associate the journal receiver with the journal. The journal receiver must be restored newest to oldest.

iii. Use the WRKJRNA command to display the receiver directory again.

Each time a journal is restored, a new receiver chain is started. This happens because the last journal receiver on the chain that existed prior to the restore process did not have the newly created receivers as its next receivers.

10.9 Recovering when a journal receiver is damaged

If a journal receiver becomes damaged, the system sends message CPF8136 or message CPF8137 to the system operator and the job log. To recover from a damaged receiver, choose one of the following options:

► If the damaged receiver is currently attached to a journal, use the Change Journal (CHGJRN) command to attach a new receiver and detach the damaged receiver.

► If the journal receiver is not currently attached to a journal, delete the journal receiver by using the Delete Journal Receiver (DLTJRNRCV) command and restore a previously saved copy.

► If the journal receiver was never attached to a journal, delete the receiver and create it again or restore it.

If the journal receiver is partially damaged, all journal entries, except those in the damaged portion of the journal receiver, can be viewed using the Display Journal (DSPJRN) command. Using this list, you can determine how you need to recover your objects. Applying or removing journal changes cannot be done with a partially damaged journal receiver.

IBM recommends that you use the Work with Journals (WRKJRN) command to recover a damaged journal receiver.

10.10 High availability solutions

In today's business environments, customers expect systems to be available almost 24 hours-a-day, seven days-a-week. One hour of server downtime can potentially cost a business hundreds of thousands of dollars. Achieving high availability requires you to implement technologies that sharply reduce the number and duration of unplanned outages. The iSeries server is the industry leader in reliability, with a single system delivering an average of 99.9+% availability.

High availability solutions on platforms other than an iSeries often use a shared disk pool. The rest of the disks are replicated or mirrored.

The iSeries high availability solution differs in that there are no shared disk pools. In addition to the high degree of disk protection afforded by RAID-5 and mirroring, information on the iSeries server can be asynchronously replicated onto another iSeries using journaling. This system can be physically connected offsite by high-speed T-1 or T-3 communication lines. This way, the building in which the primary system is located can theoretically be destroyed by a fire or natural disaster. Then the alternate site can keep critical business operations functioning (typically in a degraded performance mode) until the original site can be restored.

10.11 iSeries high availability solution providers

There are several high availability solution providers for the iSeries server. They are provided by Independent Software Vendors (ISVs). The main vendors and their respective Web sites are:

► Lakeview Technology:

http://www.lakeviewtech.com

► Vision Solutions:

http://www.visionsolutions.com

► Data Mirror Corporation:

http://www.datamirror.com

11

Movex Java Utilities

This chapter describes briefly the most important utilities that are available for you to use in Movex Java. These utilities include:

- ▶ Directory Compare
- ▶ Foundation Check
- ▶ Copy Data
- ▶ Log View
- ▶ Update Data

171

11.1 Directory Compare

The Directory Compare tool compares two directory structures to list all differences when it comes to content. It is especially helpful in that you have the ability to ignore file extensions. This makes it possible to compare a source directory structure to a binary directory structure.

This tool runs independently from the application server.
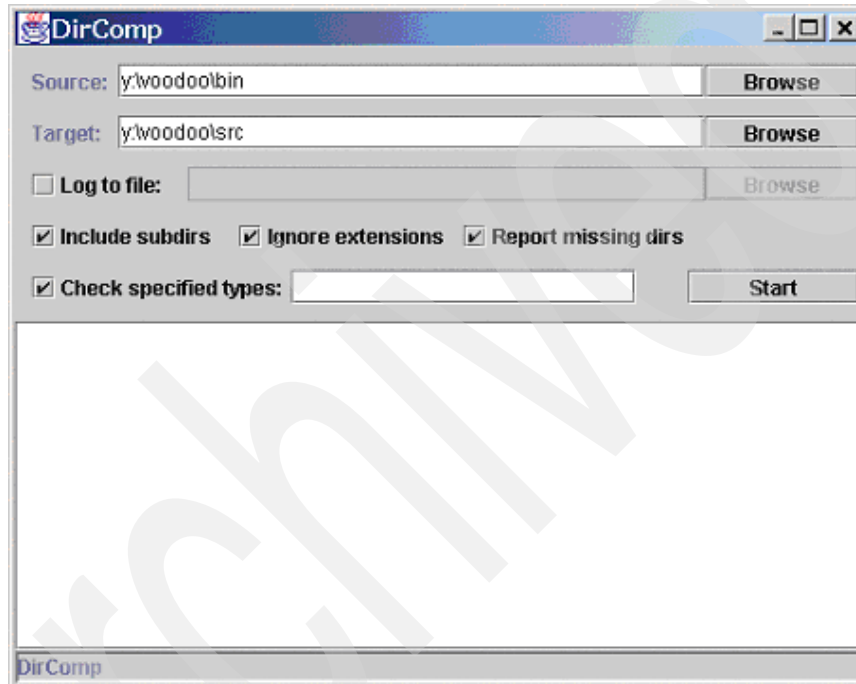
Figure 11-1   The Directory Compare user interface

11.2 Foundation Check

Foundation Check is described in the Quick Guide, which is available from the Intentia Wire. This tool requires connection to a running application server in order to function.
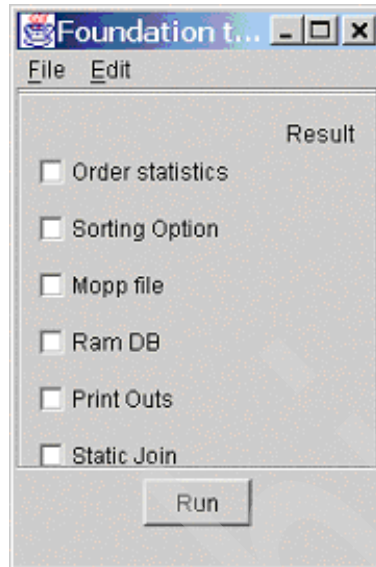
Figure 11-2   The Foundation Check user interface

11.3 Copy Data

This tool may be used to copy data within a database schema, between different schemas in the same database, or between different databases. The copy may be done by selecting individual companies or divisions that should be copied.

This tool runs independently from the application server, but requires an ODBC data source to be set up for the database server in question.
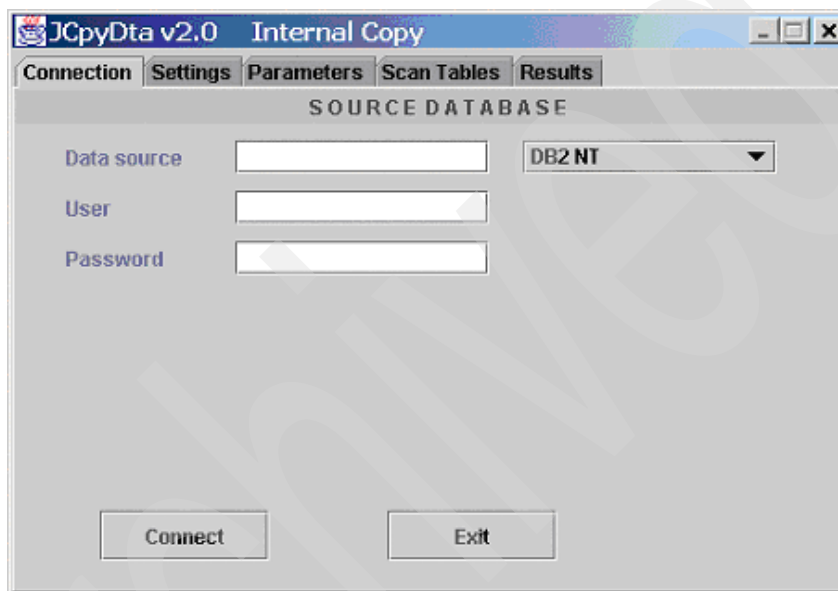
Figure 11-3   The Copy Data user interface

11.4 Log View

The Log View tool is suitable when dump logs are to be read offline from the Server View. This tool runs independently from the application server.
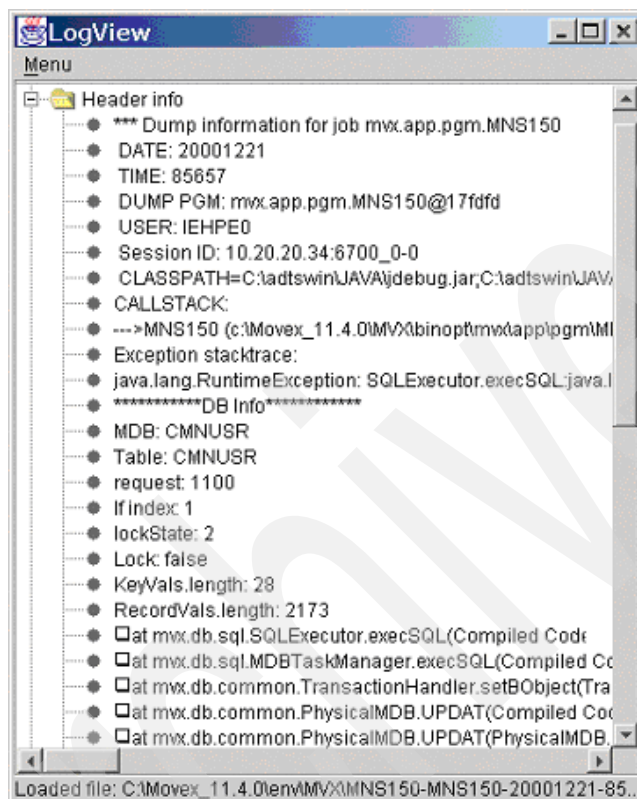
Figure 11-4   The Log View user interface

11.5 Update Data

The Update Data tool can help you manipulate the data in a specific table in the database. This tool requires a connection to a running application server to function.
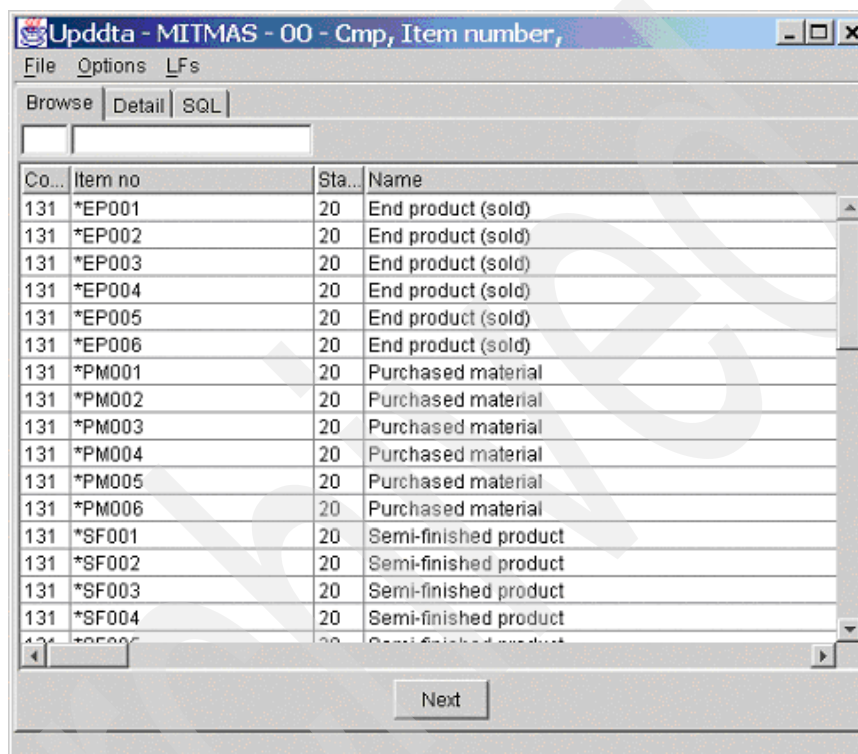
Figure 11-5   The Update Data user interface

12

System sizing

This chapter covers the basic concepts of the Movex sizing methodology.

12.1 Defining workload

The Movex Java workload can be defined both in terms of:

► Number of users concurrently logged on to the application
► Business transaction volume

Users are divided into a few main categories depending on the volume of business transactions they are assumed to complete within one working hour. A business transaction is defined as a logical unit of work from a user perspective (entering a journal entry, raising a purchase order, placing a customer order, etc.).

12.2 Definition of users

The activity level of users having an account in Movex Java varies. User activity has a major impact on the required CPUs. Main storage requirements are related to the total number of users that can be connected to the application at the same time:

► **Authorized - Global number of users defined in Movex authorization database**

The number of authorized users represents the maximum number of users defined in Movex that can access the application at the same time.

► **Concurrent - Global number of users connected to Movex application**

The actual number of connected (logged on) users represents the maximum number of users that are concurrently accessing the systems during a working day. They have interaction with the application, even if it is with different weight and workload. This number is, in most cases, smaller than the number of authorized users.

► **Low (Infrequent) - Occasional users**

An infrequent user is logged on to the system and consumes system resources. Typically this type of user definition applies to executives and controllers. These users access the system from time to time. If they are working during the peak hours, they have the least impact on the memory resource.

Infrequent users are assumed to produce one business transaction per hour.

► **Medium (Active) - Regular users**

An active user is logged on to the system and consumes system resources. Typically this type of user accesses the system permanently and is working with the system on a regular basis. This user definition applies to accountants, clerks, or office personnel.

Active users are assumed to produce four business transaction per hour.

► **High (Hyperactive) - Intensive users**

A hyperactive user is logged on to the system and consumes system resources. Typically this type of user accesses the system permanently and works intensively with the system on a regular basis. This type of user definition applies to the telesales environment and data entry work.

Hyperactive users are assumed to produce 20 business transaction per hour.

12.3 Calculating activity from a number of authorized users

In the case where there is no information on the number of users associated to each category in the previous section, the following assumptions are relevant:

► Authorized N/A
► Concurrent 80% of authorized
► Low 25% of logged on
► Medium 65% of logged on
► High 10% of logged on
► Total N/A

12.4 Calculating transaction volumes

Transaction volumes are calculated by adding the following transaction types and, from those numbers, assuming a certain peak load.

Transaction types

There are three transaction types:

► **Online transactions**: The transaction workload is created by users who interactively connect to the Movex Java application server. The transactions are calculated based on the user types/activity described earlier.

► **Batch transactions**: These transactions are generated in batch by online transaction workload. Batch transaction workload is generated during the normal OLTP processing and it refers to ad-hoc batch and reporting.

- **Scheduled batch transactions**: These transactions are generated out of the normal OLTP processing. It refers to special scheduled activities as payroll runs, General Ledger imports, etc.

Peak workload

Peak hours are the hours where the maximum number of concurrent (logged on) Movex users impose the maximum load to the Movex Java application. It can also be when large amounts of batch transactions are being processed.

12.5 Sizing methodology

All sizings (that is hardware requirement estimates) are derived from the expected number of transactions to be produced in a specific case. The expected peak and daily load expressed in transaction numbers are calculated based on such data as:

- The expected number of users
- Background workload
- Peak transaction rate and time combined with experience based assumptions on user activity

Next, those numbers are used to calculate minimum hardware requirements for the load in question.

Generally, processor requirement calculations are based on benchmarks using a mix of typical Movex Java business transactions. During benchmark testing, the amount of CPU time (note, not clock time) consumed to process a fixed number of transactions is determined for a specific processor type. When scaling up (comparing throughput for different iSeries configurations), parallels are drawn from earlier benchmarks to calculate the capability for the configuration in question.

This "cost" combined with the expected number of transactions per time unit is then used to pinpoint how much CPU power is needed. Calculation of main storage and disk arm requirements is done in a similar way. Finally, this is mapped to the current list of iSeries models.

12.6 The Quick Sizer

To provide user number-based sizing, a tool for internal use is provided. You can obtain instructions on how to use this tool by sending an empty mail message to:

`mailto: IRD-Movex-Sizing-Request@Intentia.com`

Figure 12-1 illustrates a user sizing request.
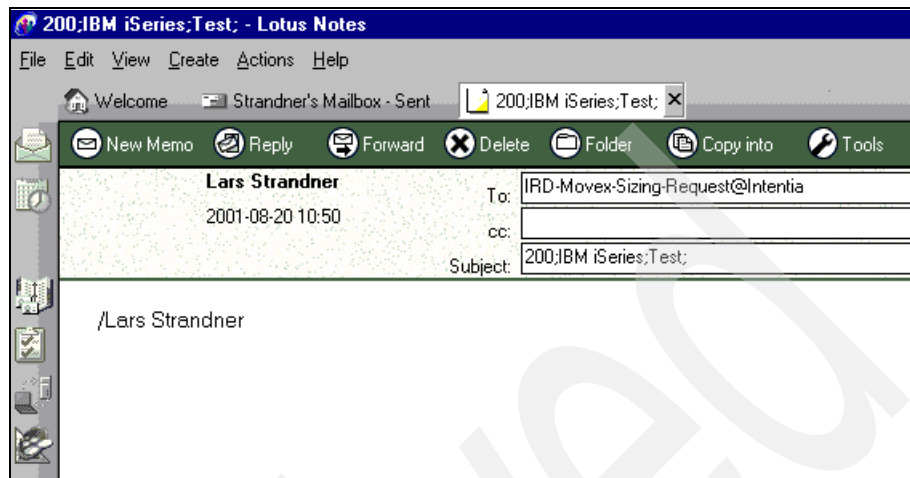
Figure 12-1   A 200 user sizing request

Figure 12-2 shows the reply from Quick Sizer.
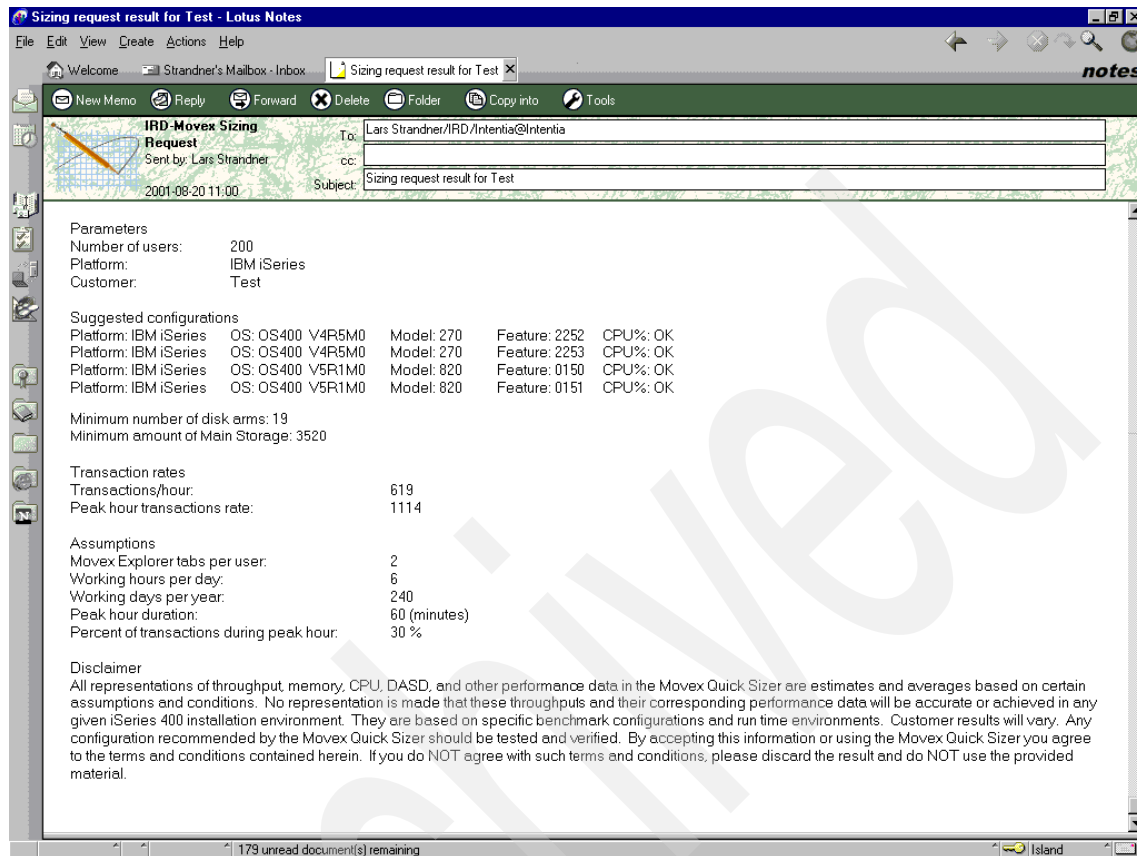
Figure 12-2   Reply from Quick Sizer

Glossary

access path   Describes the order in which records are to be retrieved and presented to the application program. Records in a physical or logical file can be retrieved using an arrival sequence access path or a keyed sequence access path. For logical files, you can also select and omit records based on the value of one or more fields in each record.

Admin client   An ordinary PC that is setup to serve the application server with graphic capabilities for administrative purposes. Most of the installation and maintenance issues are done from this client.

American National Standard Code for Information Interchange (ASCII)   The code developed by the American National Standards Institute for information exchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters, plus one parity bit.

American National Standards Institute (ANSI)   A voluntary, nonprofit organization of U.S. business and industry groups formed in 1918 for the development of trade and communication standards. ANSI is the American representative of the International Standards Organization and has developed recommendations for the use of programming languages including FORTRAN, C, and COBOL.

ANSI   See *American National Standards Institute*.

API   See *application programming interface*.

application   Represents a major functional management area in an enterprise, in the Component Repository. For each Application, a set of Components Groups is connected.

application component   Represents the basic building block for configuration of a user-specific application, in the Component Repository. The behavior of a component is exposed by its methods. Components can be configured to meet specific customer requirements, for example, operations to be carried out, choice of workflow, and data fields to expose. An Application Component consists of a Business Component, Documentation Component, and Education Component. A Business Component conforms to the notion of a business object, a representation of a real world artifact, for example, customer, item, warehouse, budget, order etc. Business Components, in turn, often use finer-grained components to contribute to their responsibility.

application programming interface (API)   A set of rules for program interaction with the purpose that programs using the API can communicate with other programs using the same API. It is an interface between the application program and the underlying system, for example functions for network management. An API consists of a number of formalized calls that the application program can use to reach the system functions.

application server   See *Movex application server*.

ASCII   See *American National Standard Code for Information Interchange*.

ASP   See *auxiliary storage pool*.

authority   In OS/400 terminology, the permission to access an object.

autostart jobs   Batch jobs doing repetitive work or one-time initialization work that is associated with a particular subsystem. The autostart jobs associated with a subsystem are automatically started each time the subsystem is started.

auxiliary storage pool (ASP)   One or more storage units defined from the disk units or disk unit subsystems that make up auxiliary storage. ASPs provide a means of placing certain objects on specific disk units to prevent the loss of data due to disk media failures on other disk units.

Backup and Recovery Media Services/400 (BRMS/400)   Provides automated backup and archival operations, and tape management services.

BRMS/400   See *Backup and Recovery Media Services/400*.

bytecode   Intermediate code that is generated by the Java compiler. The code must be interpreted or translated to run on a specific platform or processor.

bytecode interpreter   Generally included in JVM. The bytecode interpreter is responsible for reading the bytecode and for carrying out the operations that they specify. The bytecode is interpreted on the fly as the Java application steps from one instruction to the next. As new versions of the JDK are introduced, the overall performance of the bytecode interpreter improves, because of the new algorithms that are being developed.

bytecode verifier   Generally included in JVM. The bytecode verifier performs extensive checks before running a Java program to ensure that the Java bytecode has not been altered and that it still conform to Java security specifications. This involves such checks as type matching. For example, when an arithmetic operation code is encountered, the bytecode verifier checks that all the operands involved in the operation are of the integer type. If the bytecode does not pass the verifier checks, the JVM throws a runtime exception and the program is terminated. It is especially important to perform extensive checks in an open network environment where someone could run an applet from an unknown source.

C++   An object-oriented programming language developed at AT&T Bell Laboratories during the early 1980s. C++ is a hybrid language whose object-oriented features were grafted upon an existing language. It is the most widely spread object-oriented programming language. C++ is available on most hardware platforms UNIX, DOS, OS/2, Windows, and VMS.

CCSID   Coded character set identifier. A way to tell with what encoding data is stored in a database table.

class loader   Generally included in JVM. The class loader is capable of dynamically locating and loading the various classes that the application uses. This is a powerful feature, because it allows programmers to develop applications or applets that consist of many classes that can be provided by several different vendors. As the acceptance of Java grows in the entire industry, many companies are developing standard, ready-to-use software components or Java beans that greatly simplify the job of application developers.

COM   See *Component Object Model*.

component group   Consists of a group of components, in the component repository. A component group is used to refine navigation within Movex. A component group generally defines and describes a set of components.

Component Object Model (COM)   From Microsoft, COM defines how objects interact within an application or between applications. A component object conforms to this model, implementing and using the interfaces that want to access.

component repository   The Movex internal organization and architecture. The component repository serves as the basis for the mapping of Movex application components to business processes in the Enterprise Process Manager (EPM).

database management system (DBMS)   A system that has a set of tables describing the data it manages. The DBMS controls the access to the data stored within it. The DBMS also has transaction management and data recovery functions to protect data integrity.

DBMS   See *database management system*.

EBCDIC   See *Extended Binary Coded Decimal Interchange Code*.

encapsulation   A technique in which data is packed together with its corresponding procedures. In object-oriented technology, the mechanism for encapsulation is the object. It is the process of hiding all of the details of an object that do not contribute to its essential characteristics. Typically, the structure of an object is hidden, as well as the implementation of its methods. The terms encapsulation and information hiding are usually interchangeable. Encapsulation is one of the fundamental elements in the object model. Encapsulation also involves methods that make it more abstract than data hiding and information hiding.

Enterprise Process Manager (EPM)   A fully integrated process-mapping tool, which manages process design and application configuration. The tool is used to design new processes or redesign current ones. It is also possible to map responsibilities to the processes. Enterprise Process Manager consists of the Enterprise Reference Model (ERM).

Enterprise Reference Model (ERM)   A repository of business processes used as a reference when improving current processes and designing new ones. See also *Enterprise Process Manager*.

EPM   See *Enterprise Process Manager*.

ERM   See *Enterprise Process Model*.

Extended Binary Coded Decimal Interchange Code (EBCDIC)   A character presentation (not equal to ASCII) used in IBM mainframe computers. An EBCDIC character is encoded using one byte of storage. The EBCDIC character set is an IBM standard that is most commonly found on mainframe computers.

garbage collector   Generally included in JVM. The garbage collector provides fully automated memory allocation and de-allocation, which is unlike C++, where the programmer is responsible for allocating memory to store new objects and freeing unused memory when objects are being discarded. The garbage collector solves one of the main problems found in many C++ applications, which is known as memory leaks. This is one of the most difficult bugs to deal with when developing C++ applications, and often C++ applications fail with an "out of memory" error because of poor memory management. In Java, the JVM allocates the memory needed when a new object is created, while a background task, running in a separate thread, continuously scans the memory and de-allocates space that objects (without an active reference in any of the running classes) occupy. The garbage collector is key to both the performance and the reliability of Java programs.

hard link   In a file system, an actual path to an existing object. A hard link is established by creating a directory entry. A hard link cannot cross file systems. Contrast with *symbolic link*.

history files   Used when performing Movex archiving from transaction files on data that is considered by business functions to be used for historical data or statistics reasons. The transfer to the history files is often a task done on a scheduled basis and initiated by a user or a weekly, monthly, or other routine.

Implex   The Intentia implementation method developed specifically for the implementation and management of all activities within an implementation project of the Enterprise Application Movex.

index   See *logical file*.

inheritance   A mechanism where classes can use the methods and variables defined in all classes above on their branch of the class hierarchy. A relationship among classes, where one class shares the structure or behavior defined in one (single inheritance) or more (multiple inheritance) other classes. Inheritance defines a kind of hierarchy among classes in which a subclass inherits from one or more superclasses; a subclass typically augments or redefines the existing structure and behavior of its superclass. A new class is specified by enlarging an existing class from which the new class is inheriting properties. When regarding a class, it is not possible to determine any differences between attributes and methods from the current class and from the classes from which the current class has inherited. Inheritance is used to divide specifications in a natural way to obtain well-defined and well-structured classes. Inheritance between classes is used to describe that a class is a specialization of another class. General properties can be described in a class and then inherited by subclasses. Programming by inheritance allows you to define new objects by describing how they differ from existing objects. Inheritance allows the formal definition of commonality to be expressed for a set of objects. Inheritance is the ability to create a new class based on an existing class by defining the differences from the existing class. Inheritance is valid among classes, not among objects. A hierarchy is created when inheritance is used. The possibility to create polymorphism, through dynamic binding, is connected to the inheritance mechanism.

inlining   Includes a piece of code in another sequence rather than performing a branch or a call to a subroutine and dynamic dead code elimination.

Internet   Worldwide constellation of servers, applications, and information available to a desktop client through a phone line or other type of remote access.

JAR (Java ARchive) file   A file format that is used for aggregating many files into one.

Java   An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated.

Java applet   A small computer program that is written in Java and runs inside of a Java-compatible browser or appletviewer.

Java application   See *Java applet*.

Java ARchive file   See *JAR file*.

Java bean   In Java, a portable, platform-independent reusable component model.

Java Database Connectivity (JDBC)   An API for Java for connecting to and using databases. The standard way to access Java databases, as set by Sun Microsystems. Allows you to use any JDBC driver database.

Java Development Kit (JDK)   Software that Sun Microsystems distributes for Java developers. This software includes the Java interpreter, Java classes, and Java development tools. The development tools include a compiler, debugger, disassembler, appletviewer, stub file generator, and documentation generator.

Java Native Interface (JNI)   Generally included in JVM. You can view the JNI as the "glue" code that allows a Java program to start a method that is written in a language other than Java. Usually, the supported languages are C or C++. This interface allows for interoperation between Java applications and legacy applications. However, by using native methods you lose portability. By construction, native methods are written to a specific execution environment and are platform-dependent. As soon as a Java application uses code that is written in a different language, the entire application becomes platform-dependent. If portability is important to you, do not use JNI.

Java transformer A component of the iSeries implementation of Java. The Java transformer preprocesses Java bytecode that is produced by any Java compiler on any platform and contained in a class file, JAR file, or ZIP file to prepare them to run using the OS/400 JVM. The Java transformer creates an optimized Java program object that is persistent and is associated with the class file, JAR file, or ZIP file.

Java virtual machine (JVM) The part of the Java Runtime Environment (JRE) that is responsible for interpreting Java bytecodes.

JDBC See *Java Database Connectivity*.

JDK See *Java Development Kit*.

JNI See *Java Native Interface*.

Just-In-Time (JIT) compiler Often tightly integrated with the bytecode interpreter. It performs additional tasks such as setting aside, in memory, the real instructions that correspond to the bytecode. Any further reference to a bytecode that was executed once results in the execution of the corresponding real machine instruction that already exists. JIT compilers also perform code optimization functions to further improve performance. Such functions as inlining are becoming common. In fact, sophisticated compiler optimizing techniques are being implemented in JIT compilers.

JVM See *Java virtual machine*.

LDA See *local data area*.

local data area (LDA) A method to temporarily store session-unique runtime data used through out in Movex.

logical file Provides a different view of the physical data. It allows data to be retrieved in a sequence other than that specified in the physical file. It also allows field (or column) subsetting, record selection, joining multiple database files, and so on. Logical files in Movex are used to access the records stored in physical files. These files contain all key information. Besides the keys (sorting), select or omit definitions are also partly entered. Because of this additional selection, only certain records can be selected or omitted. This allows for optimal processing of online inquiries, among other things.

master table Contains general data that is used in most parts of all business functions. Examples are Customer master, Items master, Supplier master, etc.

MI program The Movex term for a kind of API program that is used for connection of other applications, such as e-business solutions to the Movex Java installation. These programs must be reached from the outside the application server.

Movex Database Connection Optimizer A database connection technology where the database access is designed as encapsulated container objects for persistent data rather than as in other technologies, where persistent data is treated as entity beans. Movex Database Connection Optimizer, where all the transaction handling is managed in the database (single or distributed), releases the middleware from coordination problems in the runtime clusters of each JVM.

Movex Explorer The user interface available for Movex version 10 and later versions. The container window in Movex Explorer contains the tree view, and possibly the list view or detail view. The window is divided into a left and a right pane. The left pane holds the tree view, while the right pane holds the windows containing different functionality in Movex.

Movex Intelligent Object Reuse   A proprietary intelligent tuning algorithm for all types of object reuse units in the runtime version Movex Java technology and architecture. All object instances in the runtime version of Movex Java, are reused in the architecture and middleware.

Movex Java Super Dispatcher   An advanced dispatcher that can be used to dedicate special types of workload to a specific Java virtual machine or server.

object-oriented programming (OOP)   A method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a class hierarchy united via inheritance relationships. In such programs, classes are generally viewed as static, where objects typically have a much more dynamic nature, which is encouraged by the existence of dynamic binding and polymorphism. Implementation of a design in an object-oriented programming language, where the objects defined in the analysis and design are directly translated into program code.

OOP   See *object-oriented programming*.

physical file   Consists of records (rows) with a predefined layout of data fields (columns). Physical files can have a keyed index included in its definition that allows the retrieval of information in a pre-defined sequence. Physical files in Movex only contain data and never key definitions. Because of this, access to physical data is always sequential. Data is processed via a logical file with a corresponding access path. This concept has been chosen to make changes and extensions of unique key definitions simpler.

polymorphism   The ability to hide different implementations behind a common interface, simplifying the communications among objects. For example, defining a unique print method for each kind of document in the system would allow any document to be printed by sending the message print, without concern for how that method was actually carried out for a given document. A concept in type theory, according to which a name (such as a variable declaration) may denote objects of many different classes that are related by some common superclass. Any object denoted by this name can respond to some common set of operations in different ways. Objects with different internal representation can communicate in the same way. It allows a common protocol to be used when communicating with objects from the same type of hierarchy. The ability to put objects of a common base type in a single array or collection, and then use the common protocol defined in the base class to communicate with the individual object. (The term polymorphism is borrowed from biology and means "many forms".) Polymorphism requires inheritance. It simplifies a programmer's task by generalizing communication syntax, which lets you treat objects of a different type in a similar fashion. That is, polymorphism uses inheritance to express commonality in the form of a communication protocol for sending messages to objects of similar (but not exactly matching) type. It is the ability of similar objects to respond to the same message in different ways. The term describes that a method can be performed on objects of different types but with different implementations for each type.

portability   Allows the same application to run on different operating systems and hardware platforms.

RDBMS   Acronym for relational database management system.

reflection   A function found in a JVM. Reflection in Java refers to the ability of a Java class to reflect upon itself (to "look inside itself"). The reflection technique allows a Java program to inspect and manipulate any Java class. This is the technique that the Java beans "introspection" mechanism uses to determine the properties, events, and methods that a bean supports. You can use reflection to query and set the values of fields, start methods, or create new objects. Java does not allow methods to pass directly as data values, but the reflection technique makes it possible for methods that pass by name to start indirectly.

reuse   The process of locating, understanding, and incorporating existing knowledge, design, and components into a new system. Should occur at all levels of system development analysis, design, implementation, testing, documentation, and user training.

scalability   Allows software, architecture, network, or hardware growth that will support software as it grows in size or resource requirements. The ability to reach higher levels of performance by adding microprocessors.

serialization   The ability to write the complete state of an object (including any object to which it refers) to an output stream, and then to recreate that object at a later time by reading its serialized state from an input stream. This technique is used as the basis for transferring objects through cut-and-paste and between a client and a server or vice versa for remote method invocation (RMI). It can also be used by Java beans to provide preinitialized serialized objects rather than a simple class file. This technique is also used as an easy way to save users preferences and application states. Serialization includes information about the class version. Obviously, an early version of a class may not be able to de-serialize a serialized instance that was created by a newer version of the same class.

Server View tool   Provides common work management functions for Movex Java environment. Server View should be used monitor, troubleshoot, and performance analyze the Movex Java environment.

SQL   See *Structured Query Language*.

Structured Query Language (SQL)   The standardized query language used in all relational databases.

subclass   A class that is a special case of another case. For example, Fox is a special case of Mammal. A class that inherits from one or more classes (which are called its immediate superclasses).

superclass   A class that is higher in the class hierarchy than another class. For example, Mammal is a superclass of Fox. The class from which another class inherits (which is called its immediate subclass), and identifies the user or terminal for which you accessed this window.

system file   Contains data that has a connection to the runtime environment or usage of system like functions such as users, access control lists, authority, etc. Usage characteristics are similar to master tables, but with distention, since there is no direct connection to functional entities.

tables   See *physical file*.

transaction file   Used to store data in or between business functions. Data in these files is used and change intensive.

Unicode   A character standard (ISO/IEC 10646) that will substitute ASCII. Unicode is a character encoding that uses a uniform 16-bit code for each language. This allows the representation of every character in every written language of the world that is likely to be used in computer communication, including the logographic symbols of Chinese, Japanese, and Korean. Unicode was developed by a consortium of computer industry companies and is documented in the book The Unicode Standard by Addison-Wesley. Unicode is used in Windows NT. A Unicode character is encoded using two bytes of storage. Unicode is not available in Windows 95 browser. Communicates with the Web server for Movex data.

view See *logical file*.

work files   Used for temporary storage for data that are being processed in order to feed a specific business function. Generally work files should be empty when the system in backup or offline mode.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering the following publication, see "How to get IBM Redbooks" on page 192.

- ► *Intentia Movex ThisGen Implementation for AS/400,* SG24-5403

Other resources

These publications are also relevant as further information sources:

- ► *OS/400 Work Management,* SC41-5306
- ► *Backup and Recovery,* SC41-5304
- ► *Backup Recovery and Media Services for iSeries,* SC41-5345
- ► *Operational Handbook,* Intentia's Implex document

Referenced Web sites

The following Web sites are also relevant as further information sources:

- ► Intentia Web site

  http://www.intentia.com

- ► Intentia Wire: You can find the most current Movex Java information on the Intentia Wire. This is an Intentia intranet site that is available only to Intentia representatives and requires a user ID and password. If you do not have access to the Intentia Wire, contact your local Intentia office.

  http://www.intentia.com

- ► Lakeview Technology:

  http://www.lakeviewtech.com

- ▶ Vision Solutions:

  http://www.visionsolutions.com

- ▶ Data Mirror Corporation:

  http://www.datamirror.com

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

   ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

Symbols

*BASE   86
*INTERACT   86
*MACHINE   86
*SHRPOOL1   86
*SHRPOOL5   86
*SHRPOOL60   86
*VERBOSEBC   102
'Authorization. Specify for Company' (MNS151)
151
'Function. Connect Authority' (SES003)   154
'User. Open' (MNS150)   148

Numerics

5722-JV1   42
5722-PT1   55
5722-SS1   42
5722-ST1   55
5722-TC1   42
5722-WSD   55
5722-XE1   55
5722-XW1   55

A

Abstract Windowing Toolkit (AWT)   42
access control   126
access path   28
activity from a number of authorized users   179
Add Environment Variable (ADDENVVAR) command   50
American National Standard Code for Information Interchange (ASCII)   28
application   7
application architecture   13
application component   7
Application menu MVXAPP   78
application tier   17
architecture of Movex   13
ASCII (American National Standard Code for Information Interchange)   28
authorized user activity   179
automatic garbage collection   45

AWT (Abstract Windowing Toolkit)   42

B

backup   160
batch subsystem   89
binopt (Business Logic)   22
boot.meproxy.range   125
BQM (Business Quality Messaging)   20
BRMS/400   160, 184
Business Components Layer   20
Business Logic (binopt)   22
business performance measurement (BPM)   5
Business Quality Messaging (BQM)   20
bytecode interpreter   38, 40
bytecode verifier   38–39

C

Call Level Interface (CLI)   46
CCSID   33, 65
Change Java Program (CHGJVAPGM) command   47–48
Change Prestart Job Entry (CHGPJE) command   73
Change Printer File (CHGPRTF) command   71
class description for Movex   92
class loader   38–39
CLI (Call Level Interface)   46
client/server implementation   12
coded character set identifier   65
cold backup   160
collector garbage collection thread   45
communication security   123
companies   10
Component Configurator   3
component group   7
configurability   14
Control   19
controlling subsystem   88
Core Library APIs   42
Core Services   19
Create Java Program (CRTJVAPGM) command   44, 47
CRS945   110

193

system sizing   177

T
Technology Independent Machine Interface (TIMI)
42
Telnet interface   125
TIMI (Technology Independent Machine Interface)
42
transaction file   32
transaction volume   179
true save-while-active (advanced) method   161

U
Unicode   33
Unicode Worldwide Character Standard   28, 33
user authentication   119
user identification   119
users, definitions of   178

W
work file   32
work management   85
Work with Relational Database Directory Entries
(WRKRDBDIRE) command   63
Work with Shared Pool (WRKSHRPOOL) command
86
Work with System Values (WRKSYSVAL) com-
mand   65
workload   178

IBM

Intentia Movex Java on the IBM @server iSeries Server

(0.2"spine)
0.17"<->0.473"
90<->249 pages

Intentia Movex Java on the IBM *e*server iSeries Server

An Implementation Guide

Overview of Movex Java on the iSeries server

Movex Java on iSeries installation and configuration

Operational tips and techniques

The Intentia Movex Java solution improves Intentia customers' business processes. It offers of an optimum set of knowledge, tools, methods, and functionality for a successful configuration and implementation of Movex. This IBM Redbook provides a detailed guide that explains specific tasks associated with implementing Movex Java on the IBM *e*server iSeries server. It is based on a collection of knowledge gathered by the architects and developers behind Movex Java, and by the Intentia professionals who have implemented Movex Java at customer sites.

This redbook is designed to assist Movex Java customers, Movex Java consultants, business partners, and IBM technical and service representatives. It targets these professionals who are directly involved with implementing a total business solution consisting of the Movex Java solution, the iSeries, the DB2 for iSeries database, and supplemental solution products.

SG24-6545-00   ISBN 0738424617