

IBM VisualAge[®] for Java[™], Version 3.5



Team Programming

Note!

Before using this information and the product it supports, be sure to read the general information under **Notices**.

Edition notice

This edition applies to Version 3.5 of IBM VisualAge for Java and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 1998, 2000. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Team development concepts 1

Team development - overview	1
Team client/server configuration	3
The repository server (EMSRV)	4
Editions and versioning.	5
Scratch editions	7
Baselines, releasing, and reloading	8
Version control for project resource files	9
Ownership and team roles - overview	11
Class developer	12
Class owner	12
Package owner	13
Project owner.	14
Repository administrator	15
EMSRV user (All server operating systems except OS/2)	16
Workspace owner	17
Repository user list.	18
Package groups	19

Chapter 2. Team development scenarios 21

Team Development Scenarios - overview	21
Team development - basic class development pattern	21
Team development scenario - single package, single developer	22
Team development scenario - single package, multiple developers	24
Team development scenario - multiple packages, multiple developers	28
Team development scenario - project wrap-up and delivery	31
Sample life cycle of an application.	32

Chapter 3. Working in a team environment 37

Logging in to the server	37
Connecting to a shared repository	37
Changing workspace owner	39
Releasing a program element or resource file	39
Creating a scratch edition.	41
Sharing resource files	42
Creating resource folders	43
Adding resource files to the project from the file system	44
Replacing a resource file with the released version	45
Building a team baseline	45
Managing editions of program elements.	46
Finding unreleased editions in the workspace	47
Finding unversioned editions in the workspace	48
Viewing a class or interface's developer	48
Viewing a program element's owner	49
Changing a program element's owner	49
Adding members to a package group.	50

Changing the owner of a resource file or folder	51
Viewing information about your repository connection.	52
Changing repositories	53
Working at a standalone workstation	55
Recovering from a server failure	56

Chapter 4. Team administration 57

TCP/IP network considerations in team development	57
Server considerations in team development.	57
Server files and directories	58
Server security	60
Number and placement of shared repositories.	61
Team and project organization	62
Setting up a team server - overview	64
Installing EMSRV as a service in the Windows registry.	65
Removing EMSRV from the Windows registry.	66
Authorizing the EMSRV user (Windows)	67
Starting the repository server on Windows	69
Starting the repository server on OS/2	72
Starting the repository server on AIX, HP-UX, Solaris, or Linux.	73
Starting the repository server on NetWare	74
Stopping a client connection.	75
Stopping the repository server	76
Displaying server connections	77
Displaying server statistics	77
Displaying active EMSRV settings	78
Displaying EMSRV messages	78
Changing the EMSRV working directory	79
Setting the server disk threshold	80
Setting EMSRV message logging options	80
Changing EMSRV settings (NetWare).	81
Adding users to the repository user list	82
Enabling password validation - overview	83
Enabling native password validation	84
Enabling password validation with the passwd.dat file	85
Providing a standard workspace	86
Creating a repository	87
Backing up a shared repository.	88
Dividing a repository	90
Optimizing server performance.	91
EMSRV startup parameters	92
EMSRV and TCP/IP	94
The EMADMIN utility - overview.	95
The EMADMIN bench command	96
The EMADMIN copy command	96
The EMADMIN list command	98
The EMADMIN opts command.	99
The EMADMIN stat command	100
The EMADMIN stop command	101

Notices	103	Trademarks and service marks	109
Programming interface information	107		

Chapter 1. Team development concepts

Team development - overview

VisualAge for Java, Enterprise Edition, provides a collaborative team development environment based on a shared repository. Change control works at the object level; it is based on object ownership.

Shared repository on a server

In the team development environment, all source code is stored in a shared repository on a server. Team members connect from their clients to the shared repository. Once connected, they can perform the following tasks:

- Find program elements in the shared repository, including those developed or owned by other team members
- Bring various editions of those program elements into their workspaces
- Create, change, test, and version program elements
- Release editions of program elements *that they own* for other team members to use as a common baseline
- Selectively replace editions of program elements in their workspace with other editions released by other members of the team

Change control based on ownership

In the VisualAge for Java team development environment, every project, package, and class has one person who is responsible for the quality of that program element, and who has the most authority over it. That person is the program element's owner.

For example, any number of developers may work on the same class, each in their own time stamped editions, but it is the owner of the class who compares their editions, merges their work to create a single version, and then releases the new version into its containing package. (Releasing updates the containing package with the new version of the class and indicates to other team members that this is a good baseline to work from.)

Similarly, packages and projects can only be versioned or reopened by their owners. Owners control the mainstream of development for their program elements.

For links to more information on program element ownership and team roles, see the related concepts listed at the end of this document.

Comparison with other source configuration management (SCM) systems

VisualAge for Java, Enterprise Edition, is different from other team development environments in the following ways:

- Team developers do not reserve or "check out" program elements, so program elements are always available to everyone on the team.
- There is no need to "check in" a program element after changing it. Incremental changes are immediately saved in the shared repository.
- Anyone on the team can access and modify any program element for development, testing, and debugging purposes, regardless of who owns the program element. This facilitates code reuse and collaborative development.

- Change is managed at the object level rather than at the file level. This facilitates parallel development of classes by more than one developer.
- Program element owners approve changes by releasing them into the team baseline. There is an emphasis on roles and responsibilities assumed by the team, rather than on file locking performed by the software.

VisualAge for Java's team features are optimized for object-oriented development in fast-moving, iterative, prototyping development environments. They are flexible and offer a high level of programmer productivity, while also providing stability.

For a more detailed discussion of setting up the team development environment, see the IBM redbook, *VisualAge for Java Enterprise Team Support* (SG24-5245-00). For information on VisualAge for Java books, select the **Library** link at <http://www.software.ibm.com/ad/vajava/>.

Using External Version Control

VisualAge for Java also offers an interface to other software configuration management (SCM) programs, including:

- ClearCase for Windows NT[®], from Rational Software Corporation
- PVCS Version Manager, from INTERSOLV, Inc.
- VisualAge TeamConnection[™], from IBM Corporation
- Visual SourceSafe, from Microsoft[®]

The interface to external version control systems is accessible from a menu, which you can use to add classes to source control, check classes in and out of the SCM repository, and import the most recently checked-in version of a class from the SCM repository.

If you select to install the interface when you install VisualAge for Java, you can use the **External Version Control** menu for convenient access from the VisualAge for Java IDE to an existing SCM tool. *There is no relationship* between external SCM functions and the version control provided by VisualAge for Java. If you use another SCM tool to manage program elements developed in VisualAge for Java, you will have to correlate the names, contents, and versions of program elements in the two systems.

RELATED CONCEPTS

Team client/server configuration
 Repository
 Editions and versioning
 Ownership and team roles - overview
 Baselines, releasing, and reloading
 Team development scenarios - overview
 Sample life cycle of an application

RELATED TASKS

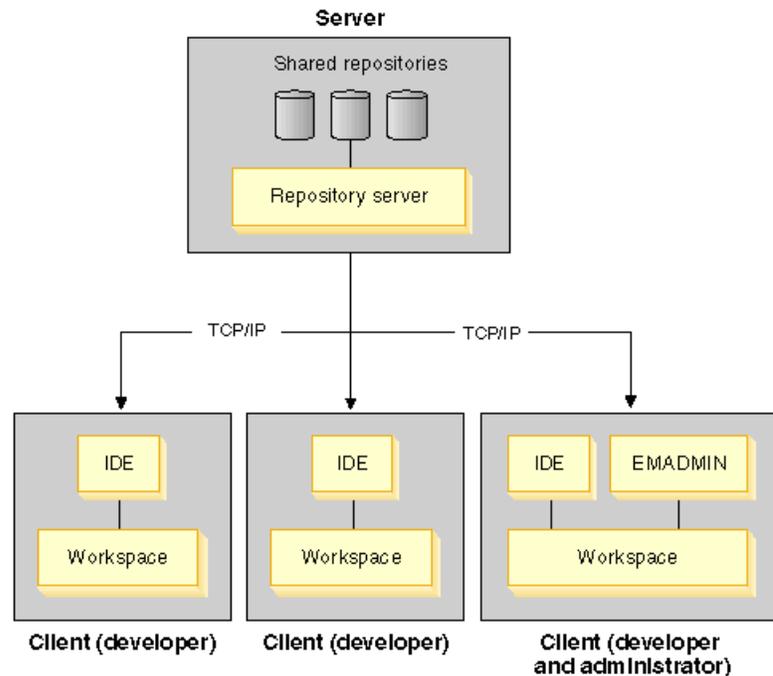
Setting up a team server - overview

RELATED REFERENCES

Repository files

Team client/server configuration

The following drawing shows the client and server components that make up a VisualAge for Java team development environment.



Example of a VisualAge for Java team network

Connectivity is provided by a TCP/IP network.

Any computer where a shared repository will reside must be a server, which is to say it must run the repository server program (EMSRV). There may be more than one server in your environment. Each server has the following components:

- One or more source code repositories (.dat files), shared by the team
- The repository server (emsrv.exe, emsrv.nlm, or emsrv). This program manages concurrent client access to the shared repositories on the server.

VisualAge for Java clients have the following components:

- The Integrated Development Environment (ide.exe)
- A workspace (ide.icx)
- Optionally, a local repository (.dat file) for offline development
- The EMADMIN utility (emadmin.exe), which is most commonly used by the person responsible for operating the repository server, for example to check the server's status

The amount of clients a server can support varies and is dependent on numerous factors, such as the number of I/O requests serviced per second.

RELATED CONCEPTS

Team development - overview

The repository server (EMSRV)

Repository
Workspace
Overview of the VisualAge for Java IDE

RELATED TASKS

Setting up a team server - overview

RELATED REFERENCES

The EMADMIN utility - overview

The repository server (EMSRV)

EMSRV is the program that manages concurrent access to shared repositories on the server. It uses native locking calls to manage file input/output requests against the repository files on the server.

The administrator must start the repository server, using the **emsrv** command, before clients can connect to the shared repository. The administrator uses the EMADMIN utility to manage the repository server.

No more than one EMSRV process should be run at a time on the same server.

The EMSRV working directory

The EMSRV working directory is the default directory that the repository server uses to locate shared repositories when, for example, a team member is changing repositories or importing from another repository. The repository server also writes its log in the EMSRV directory.

For ease of use, it is recommended that you store all shared repositories in the EMSRV working directory. This allows team members to connect to shared repositories without providing path information.

Refer to the related topics below for more information on EMSRV, including supported operating systems and changing the EMSRV working directory.

RELATED CONCEPTS

Team development - overview
Team client/server configuration
EMSRV user
Repository administrator
Server considerations in team development
Server files and directories
Server security

RELATED TASKS

Starting the repository server on Windows®
Starting the repository server on OS/2®
Starting the repository server on AIX®, HP-UX, Solaris, or Linux
Starting the repository server on NetWare
Changing the EMSRV working directory

RELATED REFERENCES

Repository files
EMSRV startup parameters
The EMADMIN utility - overview

Editions and versioning

In VisualAge for Java, whenever you work with any project, package, class, or resource file you are actually working with a specific *edition* of that program element. At any time, you can only have one edition of each program element in the workspace. To see which editions are in the workspace, click the Show Edition

Names  button.

Resource files do not appear in the workspace. To work with the resource files for a particular project, from the project's pop-up menu select **Open to > Resources**.

[ENTERPRISE] You will usually work with open and versioned editions; occasionally, you may also create scratch editions of program elements to experiment with. You will periodically release editions of classes and packages that you have been working on, to provide a baseline for the team and to make your changes easily available to them. Editions, releasing, and ownership are all fundamental to managing application changes in the team environment. Editions are discussed below; releasing and ownership are discussed as separate topics.

Open editions

Open editions are works in progress. Before you can make changes to an existing project, package, or class, you must create an open edition of it. The Workspace can only contain one edition of a program element. In the repository, however, you can have multiple open editions of the same program element, with each one implemented differently. For example, if you are adding features to an application that you have customized for different industries, you might have multiple open editions of a package with the same name stored in the repository.

Open editions appear in VisualAge for Java windows with a timestamp, in parentheses, showing when they were created. Here is an example:

PackageA (3/28/98 4:21:15 PM)

Versioned editions

Versioned editions are editions that can not be changed. You version your open editions for the following reasons:

- To keep a copy of a program element at some meaningful point, so you can return to it at a later date. In the case of packages and projects, versioning freezes a specific configuration of the contained program elements, which must also be versioned.
- **[ENTERPRISE]** To make your changed classes available to other team members who are browsing the repository.
- **[ENTERPRISE]** To release a class into its containing package, thereby updating the team baseline. Classes must be versioned before they can be released.

Versioned editions appear in VisualAge for Java windows with version names, as opposed to the timestamps that identify open editions. When you version an open edition of a program element, VisualAge for Java can automatically assign a name for you, or you can specify your own name. Here are some examples of versioned editions:

PackageA 1.6.1
PackageB VersionBRe12
PackageC JS - Fixed print problems for CustomerX

Versioning does not prevent you from ever changing a program element again. To make changes, create a new open edition of the program element. To revert to an earlier version, replace the edition in the workspace with a different edition from the repository, and create an open edition based on that.

You will probably version your classes frequently, whereas you may leave packages and projects open for extended periods of time.

[ENTERPRISE] In the team development environment, version control is achieved by means of releasing editions into a team baseline. Only program element owners can release. See the list of related topics at the end of this document for links to more information on ownership, baselines, or releasing.

[ENTERPRISE] Scratch editions

Scratch editions are editions that no other users of the shared repository can see. Scratch editions appear in VisualAge for Java windows with < > around the edition name:

PackageA <1.0>

Scratch editions are discussed separately.

[ENTERPRISE] Undefined editions

You may see a class or interface whose edition name is “undefined”:

ClassA Undefined

This means that someone has created a class or interface, but has never versioned or released it. VisualAge for Java has reserved the new program element’s name in the shared repository. Such editions are also marked with the undefined  symbol.

[ENTERPRISE] Tools for managing your editions

VisualAge for Java provides two tools for working with editions in a team development environment:

- The Managing page of the Workbench window consolidates information about all the editions that are in the workspace, and is a convenient place to perform activities such as versioning and releasing.
- The Management Query tool helps you search for program elements in the workspace by edition status. Open it by selecting **Management Query** from the **Workspace** menu.

You can also view edition details, such as status and ownership, by selecting **Properties** from a program element’s pop-up menu.

RELATED CONCEPTS

Projects and other program elements
Workspace
Repository

RELATED TASKS

Creating an open edition
Versioning a program element
Replacing editions in the workspace (reloading)

Scratch editions

Scratch editions are private. Unlike open editions, you can not version them to make them available to other developers on the team. You can create scratch editions of projects or packages. You can not create a scratch edition of a class, but you can create open editions of classes contained in scratch editions of packages.

You *can not release* into a scratch edition of a package or a project.

You might create a scratch edition for any of the following purposes:

- To learn how someone else's code works. After you finish experimenting, delete the scratch edition from your workspace or replace it with another edition from the repository.
- To test a change that you think the owner should make. If you think your change is good, talk to the class owner about integrating it into an edition that can be versioned and released.
- To start development on a class when the containing package has been versioned and the package owner is not available to create an open edition for you. You can make your changes, test and debug them, and version your class edition, but you can not release it until the package owner creates an open edition of the package for you.

If you have configured your VisualAge for Java options to show edition names, your scratch editions will be designated with < > around the program element's version name:

```
PackageA <1.0>  
PackageB 1.2
```

In the example above, PackageA is a scratch edition that was created from a versioned edition called 1.0. PackageB is not a scratch edition; it is a versioned edition.

You do not explicitly create scratch editions. VisualAge for Java automatically creates scratch editions from versioned editions, under the following circumstances:

- If you modify a class contained in a versioned edition of a package, and then save your changes. A scratch edition of the package is created in the workspace.
- If you replace the edition of a class in a versioned package with another edition of that class, a scratch edition of the package is created in the workspace.
- If you create a new open edition of a package, in a project that has been versioned, a scratch edition of the project is created in the workspace.

RELATED CONCEPTS

Editions and versioning
Baselines, releasing, and reloading
Ownership and team roles - overview
Package groups

RELATED TASKS

Creating a scratch edition
Creating an open edition

Versioning a program element
Releasing a program element
Replacing editions in the workspace (reloading)

Baselines, releasing, and reloading

Every edition of a containing program element maintains a list of editions for its content. For example, MyPackage 1.1 may contain MyClass 1.0 and YourClass 1.3, whereas MyPackage 1.2 may contain Myclass 1.1 and YourClass 1.3.

As long as a project or package is an open edition, its configuration of package or class editions can be changed. Program element owners change project or package configurations by *releasing* different package or class editions into them, or by deleting editions. Once a project or package is versioned, that particular configuration of editions is frozen.

In the VisualAge for Java team development environment, these configurations are called project or package *baselines*. Baselines determine which editions a team developer has available to work with, after adding a project or package from the shared repository or after replacing an edition in the workspace with another edition (*reloading*). Baselines allow developers to get a common view of the current state of the application, and to catch inconsistencies early.

A class owner can update a package baseline by releasing a class into the package. A package or project owner can update a project baseline by releasing a package into a project. Classes must be versioned before they can be released. Packages, on the other hand, may be released while they are still open editions. Releasing one or more open packages into a project has the effect of establishing a dynamic, or rolling, baseline for the project. As long as the project contains an open edition of a package, the *project's* configuration of classes is immediately updated every time a class is released into the contained package. The benefit of a rolling baseline is that team members can resynchronize in one step, by reloading the project instead of reloading individual packages or classes.

Since releasing affects every team member who reloads a baseline, changes should be tested before they are released. Classes may be versioned every day, but they should only be released when they are stable.

Periodically, project and package owners will preserve a baseline by versioning the project or package. At that point, all contained packages and classes must also be versioned. The result is a frozen configuration to which the team can return, if necessary.

For more information on baselines, see the team development scenarios that are listed as related topics, below.

RELATED CONCEPTS

- Version control for resource files
- Team development - overview
- Ownership and team roles - overview
- Editions and versioning
- Team development scenarios - overview
- Team development scenario - single package, multiple developers
- Team development scenario - multiple packages, multiple developers
- Sample life cycle of an application

RELATED TASKS

- Building a team baseline
- Releasing a program element or resource file
- Finding unreleased editions in the workspace
- Replacing editions in the workspace (reloading)
- Managing editions of program elements
- Adding projects and packages from the repository to the workspace

Version control for project resource files

VisualAge for Java, Enterprise Edition, Version 3.5 includes two new functions for resource files - versioning and releasing. *Resource files* are files that are not Java source files or bytecode. For example, HTML files, image files, audio clips, or SQLJ source files.

Versioning project resource files

Any project resource files contained in the project owner's local `project_resources` directory are automatically versioned when the owner versions the project. Since only the owner can version the project, all other developers on the team must provide a copy of their project resource files to the project owner if they want them versioned. Team members can do this by individually releasing project resource files, which the project owner can then access on the server.

Releasing project resource files

When a project owner versions a project, all the project resource files in the project owner's local `project_resources` directory are automatically released, updating the project baseline. Therefore, the project owner should make sure that the project resource files are correct when they version the project.

Project resource files can also be released individually by the members of the development team. When released individually, they must be released into open editions of the project. A team member may choose to explicitly release project resource files while the team is working on an open edition, so that if anyone loads or reloads the project, they will get the latest changes.

If you are working with an open edition of a project and release a resource file, then modify it and re-release it, your previous edition of the file will automatically be replaced.

Team members can add, delete, rename, or replace resource files in their workspace but may only perform these same operations on released resources if they are the project owner or the owner of the affected resources. For example, a team member can create a local copy of a resource, but they cannot change the released edition of it, unless assigned ownership of it.

For a team member to own a released edition of a resource, the project owner must release the resource and then assign ownership to it.

You can assign ownership in the Resources page of the project browser or the Workbench.

Resource files do not appear in the Repository Explorer until they have been released.

Refer to the list of related tasks below for more information on performing these tasks.

Released resources in the file system

When resources are released, they are stored in a directory in the same location as the shared repository. The name of the directory is the name of the repository with the suffix '.pr'. For example, if your repository is called `ivj.dat`, your directory would be called `ivj.dat.pr`.

Note: Never delete files directly from the file system. You should always work in VisualAge for Java to purge, copy or back up resource files.

For all platforms: Use caution when you are using clients and servers with different file systems where one file system is case-sensitive and the other is not. For example, if you have UNIX® clients and a PC server, and you have a resource called `TEST.TXT` and a resource called `test.txt` and you release them both, one will overwrite the other as they both cannot exist on the PC filesystem. Conversely, if you have a PC client and a UNIX server and you have a project with the resources `TEST.TXT` and `test.txt`, when you load those resources, one will overwrite the other as they both cannot exist on the PC file system.

Loading a project with project resource files into your workspace

When users load an edition of a project into their workspace, they can view and edit all the resources associated with a project.

If you replace an edition of a project in your workspace without releasing the resources, any changes you make to them will be lost.

When users load an edition of a project into their workspace, all of the resources associated with it are copied into the local `project_resources` directory for the project (any old resources in this directory will be deleted first). When a project is deleted from the workspace, the project resource files are deleted from the local `project_resources` directory. If you load an edition of the project into the workspace that was created in Version 2.0 or 3.0x of VisualAge for Java and does not have any project resource files associated with it, the contents of the `project_resources` directory will remain untouched.

RELATED CONCEPTS

- Resource files and directories
- Version control for Java program elements
- Baselines, releasing and reloading

RELATED TASKS

- Releasing a program element or resource file
- Sharing resource files
- Creating resource folders
- Adding resources from the file system
- Replacing a resource file with the released version
- Changing the owner of a resource file or folder

Ownership and team roles - overview

Ownership

In the team development environment, change control is based on program element *ownership*. In VisualAge for Java, Enterprise Edition, every project, package, class, and project resource file has an owner.

The owner of each program element is responsible for its integrity and consistency. For example, while several developers might work concurrently on the same class (in separate open editions), the owner of the class is responsible for merging their changes into a single edition, versioning that edition, and releasing the new version to update the team baseline.

This ownership-based system differs from development environments where change management is based on reserving program elements. In those environments, developers “check out” files to prevent others from concurrently modifying the same component. Often, any developer can check out a program element, change it, and check it back in, thereby affecting the mainstream of development.

By contrast, in the VisualAge for Java team development environment team members can work in parallel on the same program elements (in their own open editions) and their changes are automatically saved in the shared repository, but owners control the mainstream of development; they must approve changes before they are released into the team baseline. This approach offers both programmer productivity and quality control.

Team roles

In VisualAge for Java, the following team roles exist:

- Classes have *developers* and *owners*
- Packages, projects (and project resource files), have *owners*
- The repository has an *administrator*
- The repository server is started and stopped by a user called *the EMSRV user*

Each of these team roles is discussed as a separate topic. See the list of related topics, below.

There are certain operations that only these individuals can perform. In actual practice, a team member will probably perform more than one role. For example, the same person might be the developer of several classes, the owner of a few classes, and the owner of one package. Another team member may be the repository administrator and the EMSRV user.

All team members (that is, everyone who has been added to the repository user list by the administrator) are automatically class developers. All team members can browse program elements in the repository, develop classes, and create scratch editions of existing packages to experiment with. To release their work into a team baseline, for other team members to work with, developers must belong to a package group and have the appropriate ownership privileges. See the list of related topics below, for more information.

RELATED CONCEPTS

Team development - overview
Class developer
Class owner

Package owner
Project owner
Repository administrator
EMSRV user
Package groups
Repository user list
Team and project organization
Scratch editions
Editions and versioning
Baselines, releasing, and reloading
Team development scenarios - overview
Sample life cycle of an application

RELATED TASKS

Adding users to the repository user list
Adding members to a package group
Changing a program element's owner
Viewing a program element's owner
Viewing a class or interface's developer

Class developer

Any team member can create new classes and create open editions of existing classes, thereby becoming the *developer* of that open edition of the class.

Only the developer of a particular class edition can version that edition, whereas only the class owner can release the version.

The class developer and the class owner may be the same person, or there may be more than one developer working on the same class. (When multiple developers work on the same class, they work in their own editions.)

RELATED CONCEPTS

Team development - overview
Editions and versioning
Baselines, releasing, and reloading
Ownership and team roles - overview
Class owner

RELATED TASKS

Creating an open edition
Versioning a program element

Class owner

A class owner is the team member responsible for the integrity of a particular class and its methods.

Only the owner of a class can update the team baseline in the following ways:

- Release editions of the class into the containing package
- Delete editions of the class from the containing package

Often, the owner and developer of the class are the same person. Sometimes two or more developers may be working on the same class. In that case, the class

owner is responsible for comparing and merging everyone's changes into a single edition of the class. The class owner then versions the merged edition and releases it.

Within an edition of a package, each class has one owner (but may have multiple developers). The same class can have a different owner in another edition of the package.

Any member of the package group can change ownership of the class to any other member of the package group.

RELATED CONCEPTS

Team development - overview
Ownership and team roles - overview
Editions and versioning
Baselines, releasing, and reloading
Class developer
Package groups

RELATED TASKS

Comparing editions of a program element
Merging editions of a class or interface
Versioning a program element
Releasing a program element
Building a team baseline
Changing a program element's owner

Package owner

A package owner is responsible for the overall quality of a package. The owner of a package coordinates the activities of the class owners who are releasing their versioned classes into that package.

Only the owner of a package can do these things:

- Version the package to preserve a team baseline
- Create an open edition of the package so that class owners can update the team baseline by releasing or deleting classes
- Add members to the package group and delete members from the group

Both the project owner and the package owner can update the team baseline in the following ways:

- Release the package into its containing project edition
- Delete the package from its containing project edition

Both the administrator and the package owner can transfer ownership of the package to another member of the package group.

If you create a package, you are by default the owner of that package. Different editions of one package may have different owners.

Package owners control when a package is frozen and when it is opened for further development. They should ensure that an open edition of their package is available when class owners need to release their versioned classes or create new

classes. Otherwise, the class owners in the package group can only work in their own scratch editions of the package, and they can not release their work.

RELATED CONCEPTS

Team development - overview
Ownership and team roles - overview
Editions and versioning
Baselines, releasing, and reloading
Package groups
Scratch editions

RELATED TASKS

Versioning a program element
Releasing a program element
Creating an open edition
Adding members to a package group
Changing a program element's owner

Project owner

A project owner is responsible for the overall quality of a project. A project owner coordinates the activities of the package owners who are releasing their packages into that project.

Only the owner of a project can do the following:

- Version the project to preserve a team baseline
- Create an open edition of the versioned project so that package owners can update the team baseline by releasing their packages
- Add packages to the project and delete packages from the project, thereby updating the team baseline
- Assign ownership of project resource files (when they are first created)

Both the project owner and the administrator can change ownership of the project. Both the project owner and the package owner can release packages into the package.

It is worth noting that if the project owner does not create an open edition, package owners can no longer release editions and versions of their packages into the project. They can make changes to their packages, but they cannot release those changes into the project. This gives the project owner complete control over when a project is frozen and when it is reopened for further development.

RELATED CONCEPTS

Team development - overview
Ownership and team roles - overview
Baselines, releasing, and reloading
Scratch editions
Version control for Java program elements
Version control for resource files
Resource files and directories

RELATED TASKS

Creating an open edition
Versioning a program element
Releasing a program element or resource file
Sharing resource files

Repository administrator

Each repository has one administrator in its repository user list. The administrator is the only person who can do the following tasks:

- Add, delete, and change users on the repository user list
- Compact the repository

The repository administrator, along with the owners of the affected program elements, can also do these tasks:

- Purge projects and packages from the repository
- Change the owner of a project, package, or class

The repository administrator should know how your VisualAge for Java projects are organized, and which responsibilities have been assigned to each developer on the team.

Combining repository administrator with other roles

The person who performs the tasks listed above may also be a developer on the team, but the repository administrator's ID should *not* be used for application development. Change workspace owner to alternate between working as the administrator and working as a developer. This separation of roles provides the following benefits:

- Normal, ownership-based change control mechanisms can not be bypassed accidentally. For example, a developer who is not connected as Administrator can not inadvertently purge another developer's packages.
- The role of the administrator can be transferred to another team member later, without necessitating transfer of owned program elements.
- It is easier for team members to identify the real owner or developer of a program element, by looking at its properties.

The repository administrator may also be the EMSRV user. (The EMSRV user starts and stops the repository server, and backs up the repository with the **emadmin copy** command.) Unlike the EMSRV user, the repository administrator requires access to a workstation running the IDE.

Your repository administrator may also perform the following tasks, although no special VisualAge for Java privileges are required:

- Install the repository server
- Create shared repositories
- Back up shared repositories
- Enable password validation

These responsibilities may be assumed by the same person who is your LAN administrator.

The repository administrator ID

When you first install VisualAge for Java, Enterprise Edition, the only user in the repository list is Administrator. This is the default full name for the repository administrator. The first user to connect to the shared repository will connect as the

repository administrator. This person should add team members to the repository user list, and then change workspace owner before developing code.

The repository administrator's full name (which appears in user lists like the Change Workspace Owner window) and network login name (which is used to validate the administrator's password validation) can be changed by opening the User Administration window. The repository administrator's unique name cannot be changed.

RELATED CONCEPTS

- Repository user list
- EMSRV user
- Ownership and team roles - overview
- Workspace owner

RELATED TASKS

- Adding users to the repository user list
- Changing workspace owner
- Enabling password validation - overview
- Changing a program element's owner
- Creating a repository
- Backing up a shared repository
- Purging program elements from the repository
- Compacting a repository
- Setting up a team Server - overview

EMSRV user (All server operating systems except OS/2)

    The person who starts the repository server (EMSRV) is known as the EMSRV user. The EMSRV program runs with that user's privileges.

  When you start the repository server (EMSRV), you must provide the name of a user under whose authority the emsrv program will run, as one of the EMSRV startup parameters.

The EMSRV User's Password

You must know the EMSRV user's password to perform the following operations:

Server operating system	EMSRV user's password is required
   	To stop the server remotely
 	To start the server To stop the server remotely
	Not applicable. Optionally, you can provide a password as a server startup parameter, in which case the same password must be provided to stop the server remotely. This password does <i>not</i> have to be the login password of the user who starts the server.

Combining the Role of the EMSRV User with Other Roles

The EMSRV user may also be the administrator. (The administrator maintains the

list of users authorized to use the repository, and performs tasks like compacting the repository.) Unlike the administrator, the EMSRV user does not require access to the IDE.

The EMSRV user may be an existing user, or you may want to create a new user for operating the repository server. The EMSRV user requires access to the following files:

- The shared repositories that EMSRV manages on the server
- The EMSRV log file (automatically created and owned by the EMSRV user)
- The passwd.dat file, if you are using it for password validation

Security considerations

You may want to restrict access to the above files so that only the EMSRV user has authority over them.

WIN To start the repository server, the EMSRV user must belong to the Administrators group, and must have the Windows advanced user right “Act as part of the operating system”. See the Related Task reference on authorizing the EMSRV user.

WIN The EMSRV user’s password can be seen when EMSRV is started from a command prompt or as a manually started service in the Windows registry. You can reduce the risk of exposing network passwords by creating a new user just to operate EMSRV and by limiting that user’s access to anything other than VisualAge for Java files. If you install EMSRV as a service in the registry as an automatically started service, you can hardcode the EMSRV user’s name and password and they will not be visible.

RELATED CONCEPTS

The repository server (EMSRV)
Repository administrator

RELATED TASKS

Starting the repository server on Windows
Starting the repository server on OS/2
Starting the repository server on AIX, HP-UX, Solaris, or Linux
Starting the repository server on NetWare
Stopping the repository server
Authorizing the EMSRV user (Windows)
Enabling password validation - overview

RELATED REFERENCES

Repository files
EMSRV startup parameters

Workspace owner

To be connected to a shared repository, your workspace must have an owner. You choose the owner from the repository user list. The unique name (not the full name) of the workspace owner identifies you to the repository server.

The workspace owner’s privileges determine what you can do with program elements in the shared repository. For example, if the workspace owner belongs to

a package group, you can create classes in that package. When you create new program elements, the current workspace owner automatically becomes the owner of those elements.

When you save the workspace, for example by exiting the IDE, the workspace owner's name is saved. The next time you start the IDE, the workspace will be connected automatically to the same repository, with the same owner. You can change the workspace owner while you are connected to a shared repository.

The workspace owner's name is always displayed in the title bar of the Workbench window and the Repository Explorer window.

If password validation has been enabled in your environment, you will be prompted for the workspace owner's password when you start the IDE, change repositories, or change workspace owner.

RELATED CONCEPTS

- Team development - overview
- Workspace
- Repository
- Repository user list
- Package groups
- Repository administrator
- Ownership and team roles - overview

RELATED TASKS

- Adding users to the repository user list
- Changing workspace owner
- Enabling password validation - overview
- Connecting to a shared repository
- Changing repositories

Repository user list

Each shared repository on the server has a list of VisualAge for Java users who are allowed to use that repository. Team members must be added to the repository user list before they can connect to the repository or be assigned to package groups.

The repository administrator adds team members to the user list. (In addition, if password validation is enabled, the administrator will add the same users to the passwd.dat file or create user accounts in the server operating system.) Usually package owners, not the administrator, add team members to their package groups.

RELATED CONCEPTS

- Repository
- Repository administrator
- Ownership and team roles - overview
- Package groups
- Package owner
- Workspace owner

RELATED TASKS

Adding users to the repository user list
Enabling password validation - overview
Adding members to a package group
Creating a repository
Connecting to a shared repository

Package groups

In VisualAge for Java, Enterprise Edition, each edition of a package has a group of developers who are assigned to work with classes in that package. These developers are the *package group* for that edition.

Package group members can do the following things:

- Own classes in the package
- Create classes in the package
- Release classes that they own, into the package
- Delete classes that they own, from the packages
- Change ownership of *any class in the package* (not just the classes that they own)

Different editions of one package can have different package groups.

The package owner adds team members to the package group, after the administrator has added them to the repository user list.

RELATED CONCEPTS

Ownership and team roles - overview
Package owner
Class developer
Class owner
Repository user list
Repository administrator

RELATED TASKS

Adding members to a package group
Adding users to the repository user list

Chapter 2. Team development scenarios

Team Development Scenarios - overview

To take full advantage of the team programming features of VisualAge for Java, you should establish modes of operation that are suitable for your environment, and adhere to them. To help you design procedures for your own environment, this documentation presents some team development scenarios:

- Basic class development pattern
- Single package, single developer
- Single package, multiple developers
- Multiple packages, multiple developers
- Project wrap-up and delivery

These scenarios build on each other, so it is recommended that you read them in the order shown. They are adapted from the IBM redbook, *VisualAge for Java Enterprise Team Support* (SG24-5245-00). For information on VisualAge for Java books, select the **Library** link at <http://www.software.ibm.com/ad/vajava/>.

RELATED CONCEPTS

Team development - basic class development pattern

Team development scenario - single package, single developer

Team development scenario - single package, multiple developers

Team development scenario - multiple packages, multiple developers

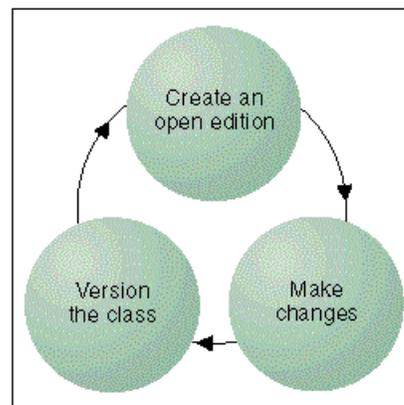
Team development scenario - project wrap-up and delivery

Team development - overview

Sample life cycle of an application

Team development - basic class development pattern

As a class developer, you move through a cycle of creating an open edition, making changes to it, and versioning it.



Basic development pattern for classes

This basic pattern forms the core of all the work you do in VisualAge for Java. You may enter the cycle at different points, depending on the context in which you are working. You leave the cycle having created a new version of the class.

The frequency with which you version your classes depends on personal style and on your team's established practices. One guideline is to version whenever you reach a known state that you may wish to return to. Versioning classes at least once a day would be typical. This pattern naturally supports an incremental development style that is usual in object-oriented programming. When you are working as part of a team, there may also be a requirement to version your classes at defined intervals. Versioning allows your classes to be seen by other developers or to be released by the class owner for the creation of a package baseline.

RELATED CONCEPTS

- Team development scenarios - overview
- Team development scenario - single package, single developer
- Team development scenario - single package, multiple developers
- Team development scenario - multiple packages, multiple developers
- Team development scenario - project wrap-up and delivery
- Team development - overview
- Sample life cycle of an application
- Editions and versioning
- Baselines, releasing, and reloading

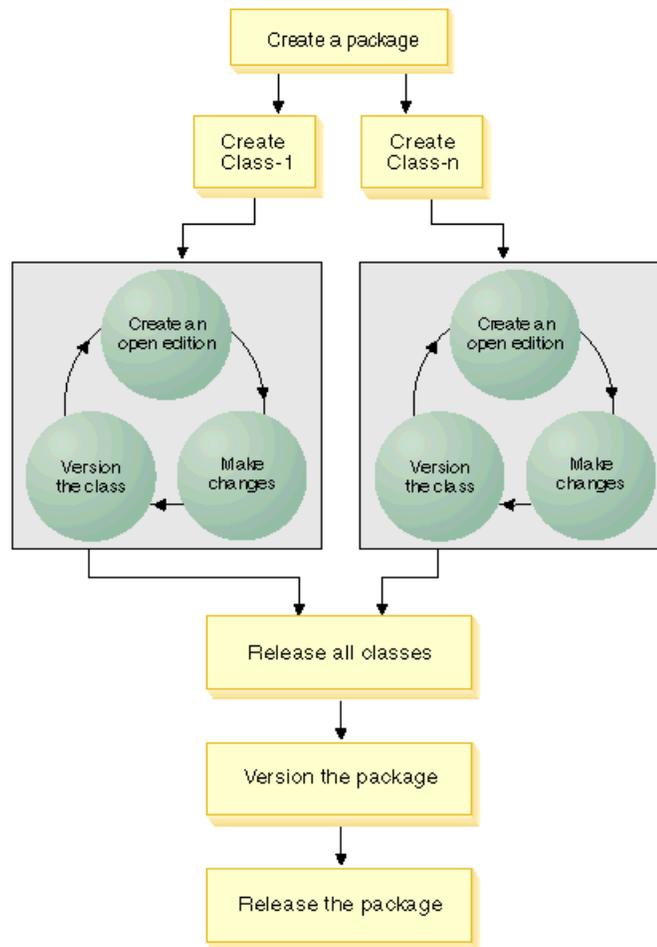
RELATED TASKS

- Creating an open edition
- Versioning a program element

Team development scenario - single package, single developer

This scenario describes the simplest way of working with VisualAge for Java, Enterprise Edition. This is a situation where one person on the team does all of the work for a small application, which consists of a single package. The application may use classes from other packages, but the developer does not need to change them.

The overall process is shown in the following illustration. It builds on the basic class development pattern that is described as a separate topic in this documentation.



Single developer working in a single package

As the developer in this scenario, you play the roles of package owner, class owner, and class developer. A project owner creates a package and transfers ownership of the package to you. You create the classes and interfaces required for the application. You follow the basic class development pattern by creating an open edition, making changes, debugging, and versioning.

When you have finished testing, you release your classes into the package, version the package, and release it. (If you work in an environment that uses rolling baselines, you would have released the package when you created the open edition. Rolling baselines are described in the team development scenario with multiple developers working on multiple packages.)

Although this is a simple scenario, many small applications will be developed this way using VisualAge for Java, Enterprise Edition.

The next team scenario describes an environment where multiple developers work together on a single package.

RELATED CONCEPTS

Team development scenarios - overview

Team development - basic class development pattern

Team development scenario - single package, multiple developers

Team development scenario - multiple packages, multiple developers

Team development scenario - project wrap-up and delivery
Team development - overview
Sample life cycle of an application
Editions and versioning
Ownership and team roles - overview
Baselines, releasing, and reloading

RELATED TASKS

Creating an open edition
Creating a package
Creating a class
Versioning a program element
Releasing a program element
Building a team baseline

Team development scenario - single package, multiple developers

In this team development scenario, multiple developers work on an application that consists of a single package. This scenario highlights the cooperative nature of VisualAge for Java team development, and introduces the concept of a package baseline.

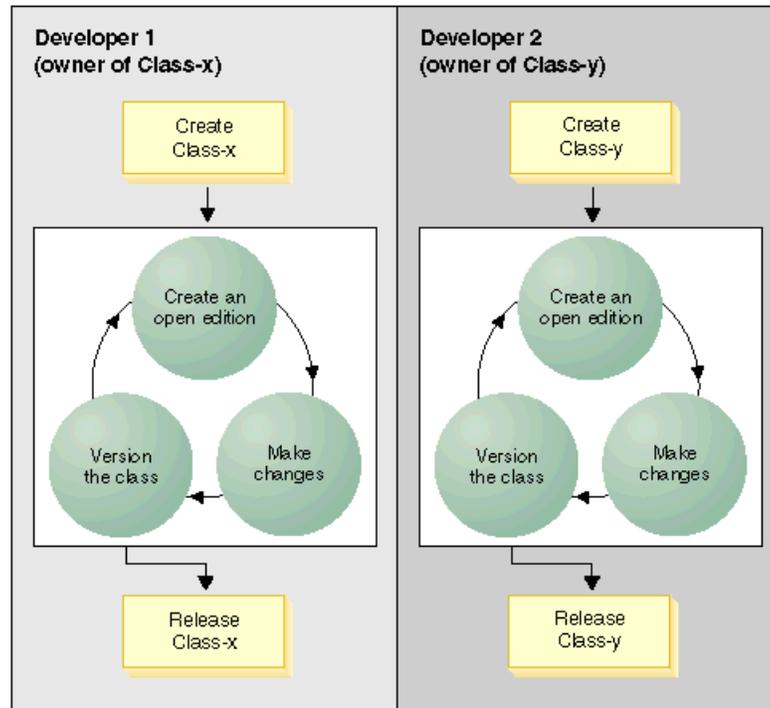
Project set-up

Typically, in this situation, the lead developer acts as the package owner. The project owner creates the package and assigns ownership to the lead developer.

After some analysis of the problem and after an initial design, the set of classes and interfaces that make up the application are known. The lead developer allocates responsibility to each of the other developers, for a number of classes. As the package owner, the lead developer also adds team members to the package group so that they can create classes and interfaces within the package.

Basic development pattern

As with the other team scenarios, the development process in this environment is built on the basic class development pattern. As the following figure illustrates, each developer follows the basic pattern independently.



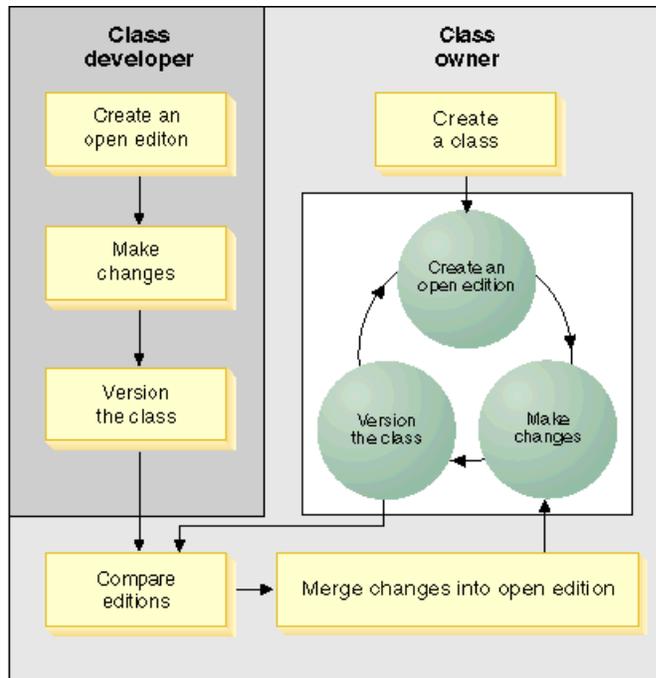
Developers working independently on their own classes

The developer who creates a class or interface automatically becomes its owner. Other developers may make changes to a class they do not own but, as described later, the owner must accept or reject those changes and make them available to the rest of the team. The owner is responsible for the class.

Merging changes made by different team developers

In practice, team development is not always as simple as suggested above, since developers are often dependent on each other's work. For example, a developer working on one class may need new function in another class, or may be affected by a bug in another class. And, the owner of that class may be too busy to make the required changes immediately. In the VisualAge for Java team environment, the developer who needs the change can create an open edition of the relevant class, make the changes there, and use that edition to continue working productively.

The person who creates an open edition of a class is the only person who can version that particular edition, but the class owner is the only person who can release the class into the team baseline. As the next step in this scenario, the developer versions the edition to make the changes visible to the class owner, and then consults with the class owner to get approval. The class owner may release the versioned edition immediately, but it is quite likely that the class has been further developed by the owner in the meantime, in which case the owner would merge the two sets of changes in a single open edition. This process is shown in the following illustration.



Class owner reconciles changes made by another team member

Change reconciliation is easy to do with the comparison browser in VisualAge for Java; the owner views class and method definitions side-by-side, and selectively loads from other developer's version into the open edition in the workspace. Reconciliation of class changes should occur on a regular basis. Fewer errors occur when reconciliations are small and frequent.

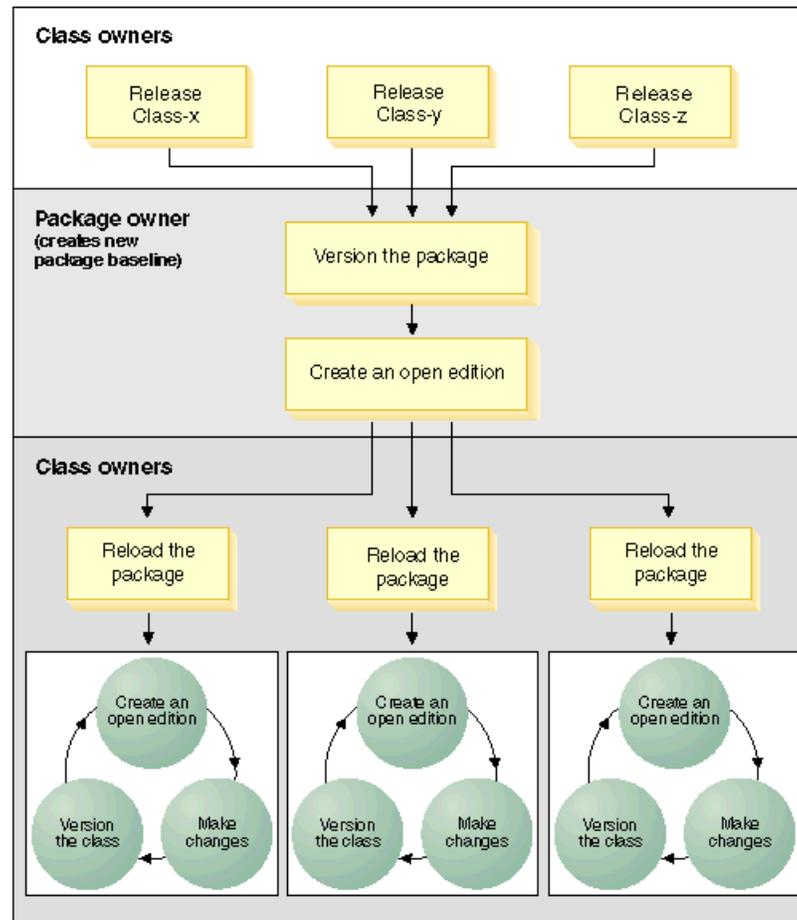
Once the changes have been merged, the normal development pattern resumes.

Establishing a package baseline

The project manager has established a plan that identifies milestones in the application development process. Milestones generally represent the achievement of a certain level of functionality within the application. They also provide an opportunity for all developers to synchronize with each other and get a common view of the current state of the application. This process is called establishing a *baseline*. Baselines help the team to find inconsistencies early in the development process, before they become entrenched.

When a deadline is approaching, the lead developer expects programmers to have all their classes in a working state, and instructs the class owners to release. The lead developer, who is the package owner, establishes a new baseline by versioning the package. The lead developer then opens a new edition of the package, so the team can resume work on it. Before continuing with their development work, the team developers synchronize with the new baseline by reloading the package. To reload, they select **Replace with > Another Edition** from the package's pop-up menu in the Workbench, and choose the new open edition.

The following figure illustrates the process of establishing a baseline for a package.



Establishing a package baseline

At this point, the class owners resume the normal development pattern until the next project milestone and the production of a new baseline.

The next team scenario describes a more complex environment, where multiple developers work on multiple packages.

RELATED CONCEPTS

- Team development scenarios - overview
- Team development - basic class development pattern
- Team development scenario - single package, single developer
- Team development scenario - multiple packages, multiple developers
- Team development scenario - project wrap-up and delivery
- Team development - overview
- Sample life cycle of an application
- Editions and versioning
- Ownership and team roles - overview
- Baselines, releasing, and reloading

RELATED TASKS

- Creating an open edition
- Creating a package
- Creating a class

- Versioning a program element
- Releasing a program element
- Building a team baseline
- Comparing editions of a program element
- Merging editions of a class or interface
- Replacing editions in the workspace (reloading)

Team development scenario - multiple packages, multiple developers

This scenario is typical of team development on a large-scale project. In this environment, a number of people work together on an application that includes multiple packages. This scenario extends the other team development scenarios included in this documentation.

Set-up and basic development pattern

A large development project includes many distinct subsystems, partitioned into separate packages. The project team is divided into smaller teams; each team is responsible for a particular package. The project owner, who is perhaps the chief architect, creates the packages and assigns ownership of each package to the corresponding team leader. Each team leader adds the team members to the package group.

As in the previous team development scenarios, developers in this environment follow the basic class development pattern: they create open editions, make changes, and version their open editions. As in the scenario with multiple developers working on a single package, package baselines are established on a regular basis, and class owners reconcile other developers' changes to their classes. Change reconciliation sometimes occurs across package boundaries, when a developer working in one package has to change a class that is in another package. The process for merging changes across packages is exactly the same as it is within a single package.

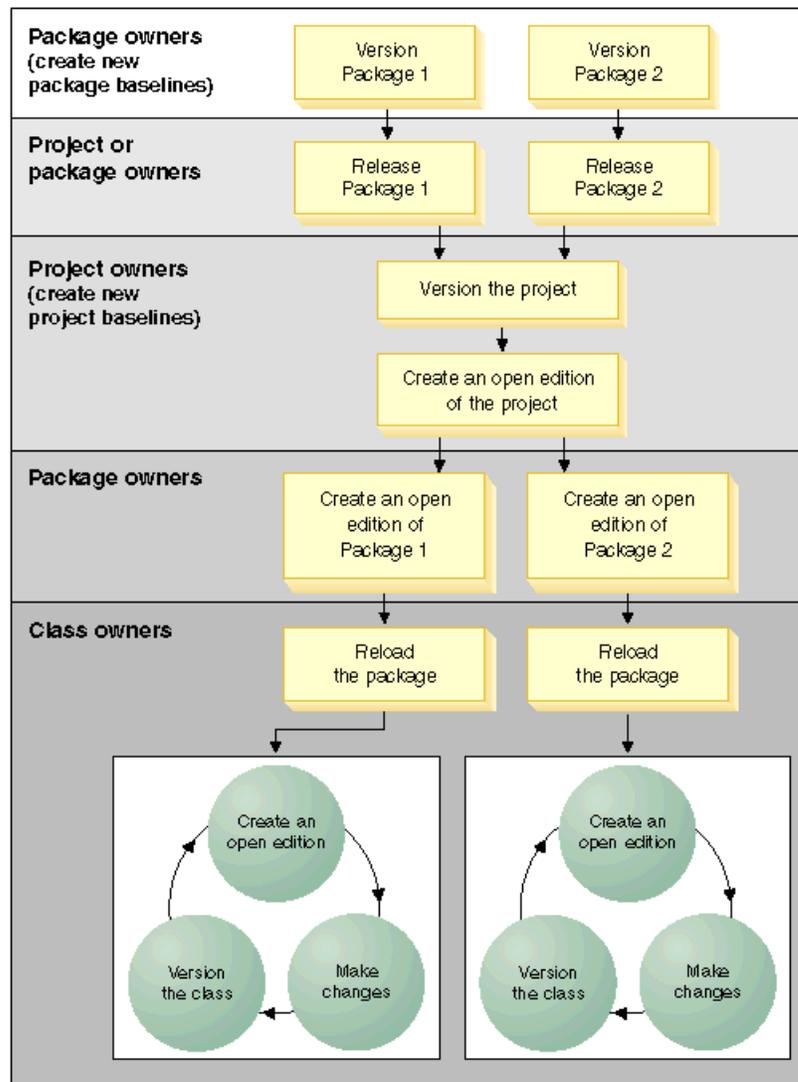
Standard project baseline

There are two approaches to establishing a project baseline for the team. One is the standard project baseline, which is static; the other is the rolling baseline, which is dynamic.

The standard project baseline is analogous to the package baseline that was described in the scenario with multiple developers working on a single package. To establish a standard project baseline, you start by creating a baseline for each package in the project. Next, the versioned packages are released into the project, either by the individual package owners or by the project owner. Once all the packages are versioned and released, the project owner creates a new baseline by versioning the project.

The new version represents an immutable state of the project, its packages and all their classes. This frozen state distinguishes the standard baseline from the rolling baseline that is described below. With a standard project baseline, if work is to continue, the project owner must create an open edition of the project so the package owners can create open editions of their packages and the class developers can recommence releasing changes into the packages.

The following figure illustrates the process of establishing and loading a standard project baseline.



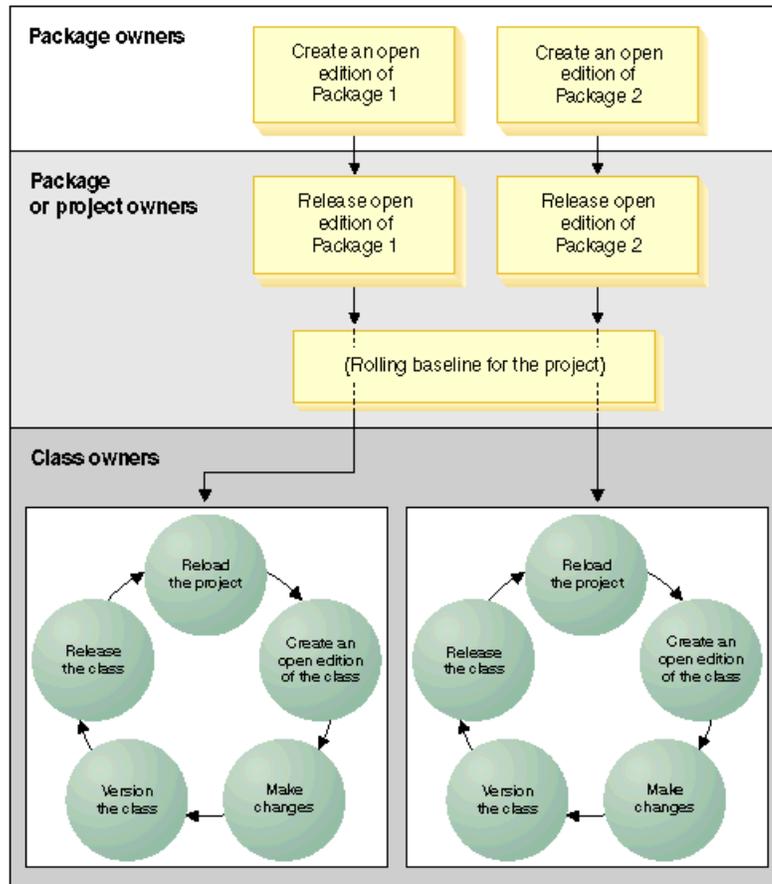
Establishing a standard project baseline

Rolling project baseline

You are most likely to create a standard project baseline at major project milestones and at the end of a project. For ongoing development, it can be quite a burdensome process. It is often difficult for the team to produce a baseline for all packages at the same time. VisualAge for Java provides a more flexible way for the team to synchronize with the current state of the application. The alternative is to establish a rolling project baseline.

The main difference in this approach is that packages are released into the project while they are still *open editions*. Unlike classes, packages do not have to be versioned before they can be released. As a result, the project baseline is dynamic: when a class is released into a package, the configuration of the containing project is simultaneously updated. Any time a developer reloads the project (by selecting **Replace with > Released Contents** from the project's pop-up menu in the Workbench), the class editions that are in the workspace are all replaced with the editions that the class owners have most recently released.

The following figure illustrates the process of establishing and loading a rolling project baseline.



Establishing a rolling project baseline

Combining baseline approaches

Caution should be exercised with the rolling baseline technique. Releasing a class straight into the project has the effect of bypassing any integration testing that, in a standard baseline environment, would occur at the package level. The release of a defective class could seriously affect the productivity of any programmer who loads it. For that reason, you might want to restrict the rolling baseline technique to packages where the class owners are senior programmers who always test their classes prior to release.

RELATED CONCEPTS

- Team development scenarios - overview
- Team development - basic class development pattern
- Team development scenario - single package, single developer
- Team development scenario - single package, multiple developers
- Team development scenario - project wrap-up and delivery
- Team development - overview
- Sample life cycle of an application
- Editions and versioning
- Ownership and team roles - overview
- Baselines, releasing, and reloading

RELATED TASKS

- Creating an open edition
- Creating a package

Creating a class
Versioning a program element
Releasing a program element
Building a team baseline
Comparing editions of a program element
Merging editions of a class or interface
Replacing editions in the workspace (reloading)

Team development scenario - project wrap-up and delivery

Like the basic class development pattern that is presented as a separate topic, the process for delivering a project is common to both small and large team development environments.

The first step is to produce a standard project baseline, as described in the team development scenario with multiple developers working on multiple packages. You may wish to create a separate project for the released version, to reinforce the separation of development and production versions of the project. Or, you may use the existing project as the reference version. In either case, *add comments* to the open edition of the project before you version it. It is useful to establish a standard format for comments, and to use it extensively.

When you have a completed, versioned project, choose a delivery mechanism. VisualAge for Java, Enterprise Edition, offers a number of options for exporting your work so that it can be delivered to your customer:

- Export .class or .java files to the file system
- Publish an applet
- Produce a .jar file
- Export to another repository

These tasks are discussed as related topics.

RELATED CONCEPTS

Team development scenarios - overview
Team development - basic class development pattern
Team development scenario - single package, single developer
Team development scenario - single package, multiple developers
Team development scenario - multiple packages, multiple developers
Team development - overview
Sample life cycle of an application
Editions and versioning
Baselines, releasing, and reloading

RELATED TASKS

Building a team baseline
Exporting and publishing code
Deploying an applet on the network station
Deploying an application on the network station
Exporting to another repository

RELATED TASKS

Creating a project
Versioning a program element

Sample life cycle of an application

The following example shows the steps that a team of developers might follow to deliver enhancements to an existing VisualAge for Java application. In the example, there are five team members:

- The repository administrator
- A project owner
- A package owner
- Two class owners

In your own environment, one person may perform two or more of these roles, or your team may be much larger than five people. The basic flow is the same, regardless of whether you have a “team of one” or a team of twenty or more developers.

Similarly, the example shows the creation of one project, one package, and two classes. In your own projects you may be working with existing program elements, and you may have many more of them.

Phase 1: Project initiation

The steps in this phase would only be performed once, at the beginning of the project.

Team member	Steps performed
Administrator	<ul style="list-style-type: none">• Adds all members of all project teams to the repository user list.
Project Owner	<ul style="list-style-type: none">• Creates a new project, using the Comments pane of the SmartGuide to document the purpose. By default, the edition status is open.• Adds any necessary packages from the shared repository to the open edition of the project in the workspace, and then releases them to update the project baseline.• Creates one or more new packages, including PackageA, in the open project. New packages are released automatically.• Adds Package Owner, Class Owner A, and Class Owner B to the package group for PackageA.• Transfers ownership of PackageA to Package Owner.
Administrator	<ul style="list-style-type: none">• Reviews ownership of existing packages. If classes in those packages require changing, transfers package ownership to someone on the team. (The new package owner would add developers to the package group and assign ownership of classes.)
Package Owner, Class Owner A, Class Owner B	<ul style="list-style-type: none">• Connect to the shared repository and add the open edition of the project to workspace. This action will automatically add the released package and class editions contained in the project.

At this point, every team member’s workspace contains the open edition of the new project, any existing packages (and any contained classes) that the project owner added to the project, and an open edition of PackageA. The team is ready to begin development.

Phase 2a: Ongoing development

In this phase, the team develops the planned enhancements to their application. As is always the case with VisualAge for Java, the development process is iterative.

Team Member	Steps Performed
Class Owner A	<ul style="list-style-type: none"> • Creates one or more new classes in PackageA, including ClassA. • Versions open class editions after completing significant blocks of work.
Class Owner B	<ul style="list-style-type: none"> • Creates one or more new classes in ProjectA, including ClassB. • Versions open class editions after completing significant blocks of work.
All Team Members	<ul style="list-style-type: none"> • As necessary, add other team members' versioned classes to their workspaces, for testing. • As necessary, make changes to classes. (Each team member works in a different open edition of each class.) • Version open class editions after completing significant blocks of work.
Class Owner A	<ul style="list-style-type: none"> • On request, compares other developers' changes to ClassA and merges them into a single versioned edition. • Releases the latest stable version of ClassA into the open edition of PackageA in the repository, to update the team baseline. • Tells the team when to replace the edition of ClassA that is in their workspaces, with the latest released contents.
Class Owner B	<ul style="list-style-type: none"> • Performs the same activities as Class Owner A, for the classes he owns (including ClassB).
All Team Members	<ul style="list-style-type: none"> • When advised by class owners, replace the class editions in their workspaces with more recently versioned editions from the shared repository by selecting Replace with Released Edition from the pop-up menu for the <i>class</i>. • When they want to reload <i>all</i> the classes within a package or project, select Replace with Released Contents for the <i>package</i> or <i>project</i>. If this backs out changes that they want to keep, they can retrieve their open editions by selecting Replace with Another Edition for those specific program elements.
Administrator	<ul style="list-style-type: none"> • Every night, backs up the shared repository.

You may have noticed that, in this example, the package and project editions remain open while the classes are frequently versioned. That is a typical development pattern. If you prefer to version projects or packages while classes are still being developed, then it is important that project and package owners create new open editions immediately after versioning, so that class developers can continue to release their changes into the team baseline. For more information on development patterns and team baselines, see the team development scenarios that are included elsewhere in this documentation.

Phase 2b: A deviation from the main development flow

This minor variation illustrates how members of the VisualAge for Java package group are really "trusted peers".

Team member	Steps performed
Package Owner	<ul style="list-style-type: none"> • Makes some useful changes in an open edition of ClassB.
Class Owner B	<ul style="list-style-type: none"> • Likes the changes, but is too busy to incorporate them. Suggests that Package Owner incorporate the changes herself.

Package Owner (new owner of ClassB)	<ul style="list-style-type: none"> • Transfers ownership of ClassB to herself. (Any member of the package group can do this.) • Compares her changes with the latest released version of ClassB, in case there are new changes to be merged. • Versions her open edition of the class, and releases it.
Class Owner B (original owner of ClassB)	<ul style="list-style-type: none"> • Loads Package Owner's released version of ClassB into the workspace (Replace with Released Edition). • Changes ownership of ClassB back to himself.

This example shows how flexible the team development environment is, and why communication and cooperation among team members is very important.

Phase 3: Preparation for testing

At this point, the team is ready to close off work, for acceptance testing.

Team member	Steps performed
Class Owners	<ul style="list-style-type: none"> • Search for their unversioned editions (using Management Query from the Workspace menu), so they can version and release them. • Search for remaining unreleased versions and release them.
Package Owner	<ul style="list-style-type: none"> • Searches to make sure there are no unversioned or unreleased class editions. • Versions PackageA.

At this point, someone on the team would bundle the VisualAge for Java program elements with the necessary resource files (for example, images or audio clips) and deliver the application to the testers. The steps required to do this, which might include exporting to the file system or publishing to an intranet, are not dependent on any particular VisualAge for Java privileges or team roles.

The package will remain frozen until testing has been completed. Meanwhile, team members who want to experiment with changes to classes in the package can do so by creating their own scratch editions of the package.

Phase 4: Wrap-up

Assuming that acceptance testing goes well and the users only report a few small problems, the team would follow a process similar to the following one in order to wind down the project.

Team member	Steps performed
Package Owner	<ul style="list-style-type: none"> • Creates a new open edition of PackageA so class owners can make changes. • Releases the open edition of the package, so the project baseline will be updated automatically as classes are released.
Class Owners	<ul style="list-style-type: none"> • Reload the project (Replace with Another Edition). • Make final changes to their classes. • Version and release their changed classes into the open edition of PackageA.
Package Owner	<ul style="list-style-type: none"> • Updates the comments for Package A. • Versions PackageA (which was released when it was created).
Project Owner	<ul style="list-style-type: none"> • Updates the comments for the project. • Versions the project.

Again, someone on the team would take the necessary steps, such as exporting, to actually deploy the application. Deploying is not dependent on any VisualAge for Java privileges or team roles.

RELATED CONCEPTS

- Team development - overview
- Ownership and team roles - overview
- Editions and versioning
- Baselines, releasing, and reloading
- Team and project organization
- Team development scenarios - overview
- Team development scenario - single package, multiple developers
- Team development scenario - multiple packages, multiple developers
- Team development scenario - project wrap-up and delivery

RELATED TASKS

- Creating an open edition
- Creating a scratch edition
- Comparing editions of a program element
- Merging editions of a class or interface
- Versioning a program element
- Releasing a program element
- Building a team baseline
- Replacing editions in the workspace (reloading)
- Managing editions of program elements
- Finding unversioned editions in the workspace
- Finding unreleased editions in the workspace

Chapter 3. Working in a team environment

Logging in to the server

If the VisualAge for Java repository server (EMSRV) has been started with password validation enabled, you will be prompted for a password when you perform either of these tasks:

- Connect to a shared repository on the server
- Change workspace owner

Otherwise, there is no “login” required. While you are using VisualAge for Java, you have the privileges of the current workspace owner. To work as someone else, select **Change Workspace Owner** from the **Workspace** menu.

RELATED CONCEPTS

Team client/server configuration
The repository server (EMSRV)
Workspace owner
Repository user list

RELATED TASKS

Connecting to a shared repository
Changing workspace owner
Enabling password validation - overview
Adding users to the repository user list

RELATED REFERENCES

EMSRV startup parameters

Connecting to a shared repository

Your workspace is automatically connected to a shared repository when you perform *one* of these tasks:

- Start the IDE in the team development environment. The workspace will connect to the same server and shared repository that you were using when you last exited the IDE.
- Change to another shared repository, after starting the IDE.

The first time that you connect to a shared repository, you will be asked to select a workspace owner from the repository user list. Contact the repository administrator if your name is not on the list. If password validation has been enabled on the repository server, you will need to provide your password to connect to any repository on that server.

Whenever you connect to a repository, VisualAge for Java performs a complete check for consistency with the workspace. This can take some time. Inconsistencies are noted in the Log window.

Once you have connected to a shared repository, you can browse it using the Repository Explorer window. Ask your project leader which editions you should add from the repository to your workspace, so you can start programming.

The ide.ini file

The names of the server and the repository are stored in the [JavaDevelopment] section of the ide.ini file on your client workstation. This information is provided when you install VisualAge for Java on the client, and it is saved when you exit the IDE. Here is an example:

```
[JavaDevelopment]
ServerAddress = bestteam
DefaultName = team1.dat
```

In the above example, “bestteam” is the IP host name of the server and team1.dat is the name of the shared repository. Because no path information is provided for the repository, the repository server the EMSRV working directory. See the list of related topics, below, for more information about the EMSRV working directory.

Here is another example:

```
[JavaDevelopment]
ServerAddress = 9.55.55.155
DefaultName = j:\javateam\ivj.dat
```

In the second example, the server’s IP address is used instead of its host name, and explicit path information is provided for the shared repository. In this example, j: is a local drive on the *server*.

Handling connection errors

If you have trouble connecting to a shared repository, check the following things:

- Are the server and repository names spelled correctly in the ide.ini file?
- Has EMSRV been started on the server? Use the **emadmin stat *hostname*** command to verify.
- Did the administrator supply the right EMSRV startup parameter for working directory? To see the current EMSRV working directory, issue the **emadmin opts** command.
- Has TCP/IP been started on the server and the client? To verify, ping the server from the client, using the appropriate command for your TCP/IP software.
- Has some process other than EMSRV locked the repository file?

RELATED CONCEPTS

Team development - overview
Team client/server configuration
The repository server (EMSRV)
Repository administrator
Repository user list

RELATED TASKS

Changing repositories
Adding users to the repository user list
Enabling password validation - overview
Changing workspace owner
Adding projects and packages from the repository to the workspace
Adding classes and methods from the repository to the workspace
Reconnecting after a server failure
Working at a standalone workstation
Changing the EMSRV working directory

RELATED REFERENCES

EMSRV startup parameters
The EMADMIN utility - overview
The EMADMIN stat command
Repository files

Changing workspace owner

The workspace owner's unique name, from the repository user list, is what identifies you to the repository server. At times, you might want to work as someone else. For example, you may want to work as the repository administrator so you can add users. You can do this by changing the current workspace owner.

To change workspace owner:

1. From the **Workspace** menu, select **Change Workspace Owner**. A dialog box will appear, listing all the names from the repository user list.
2. Select the new workspace owner, and click **OK**.
3. If password validation is enabled on your server, you will be prompted for the network password of the new workspace owner. Type the password and click **OK**.

You can also change the workspace owner by double-clicking on the current owner's name in the status bar at the bottom of the window.

On all VisualAge for Java windows, the title bar will show the new workspace owner's name.

RELATED CONCEPTS

Workspace owner
Repository user list

RELATED TASKS

Enabling password validation - overview
Adding users to the repository user list

Releasing a program element or resource file

In VisualAge for Java, Enterprise Edition, you *release* a class, package or resource file to update the team baseline. A baseline is the combination of class editions that make up a specific edition of a package, or the combination of package editions that make up a specific edition of a project.

Releasing is very important in the team development environment, because it determines which editions get added to the workspace when a team member performs any of the following actions:

- Adds a project or package to the workspace.
- Selects **Replace with Released Edition** for a package, class or resource file.
- Selects **Replace with Released Contents** for a project or package.

Resource files and folders are automatically released when the project owner versions the project. Resource files can also be released individually. You can also release entire resource folders and all the resources contained in them (you must select the folder and all of the files to do so). Team members can only release project resource files or folders that they own the released edition of. If you are

working with an open edition of a project and release a resource file, then modify it and re-release it, your previous edition of the file will automatically be replaced.

When you release your resources, they are stored in a directory in the same location as the repository. The name of the directory is the name of the repository with the suffix '.pr'. For example, if your repository is called `ivj.dat`, your directory would be called `ivj.dat.pr`.

If you export projects with project resources to a repository, a stored resource directory will be created for the resources. If you are going to make a copy of the repository, you should always ensure that you also copy the stored resources directory.

Unlike solutions, projects, types, and methods, resources only exist on the server when they are released. After you have released a resource file, you cannot re-release it until you have modified it.

Once team members have released their resource files, the project owner can load the latest copies of the them into their workspace, and then version the project, setting a new baselines, making the newly updated resource files available to all the team members.

To release packages and classes:

1. In a browser or the Workbench window, select the editions that you want to release. (You can make multiple selections by holding down the Ctrl key.)
2. From the pop-up menu, select **Manage > Release**.

To release a resource file or resource file folder, follow these steps:

1. Select and double-click the project that contains the project resource files you wish to work with. The project browser opens.
2. Click the **Resources** tab.
3. Select the resource or resource folder that you want to release and from its pop-up menu select **Release**. When you release a folder, the files in it are *not* automatically released. You must select them, as well as the folder, if you want to release them.

Once you have released the editions, the unreleased (>) marker no longer appears beside their names in the Managing page of the Workbench window. The open edition of the containing project or package is updated with the newly released edition of the class, package or project resource file. Team members who had previously added the project or package to the workspace *are not notified* when the baseline is changed. Inform the team when you release a class, package or project resource file, so they can take the appropriate action to replace the edition in their workspaces.

To determine the last released edition, select **Properties** from the program element's pop-up menu and then select the **Info** page of the Properties notebook.

The following considerations govern releasing:

Program element	Consideration
Project	<ul style="list-style-type: none">• Not applicable.

Package	<ul style="list-style-type: none"> • Both open and versioned editions of packages can be released. • The project into which you are releasing must be an open edition. • You must be the package owner or the project owner. • When you create a package, VisualAge for Java automatically releases the initial open edition of the package. • When you version a project, VisualAge for Java automatically releases any unreleased packages that it contains, irrespective of who owns the package.
Class	<ul style="list-style-type: none"> • Only versioned editions of classes can be released. • The package into which you are releasing must be an open edition. • You must be the class owner. • When you version a package, VisualAge for Java automatically versions and releases the contained classes that you own.
Method	<ul style="list-style-type: none"> • Methods are automatically released when you save them.
Project resource files	<ul style="list-style-type: none"> • You must be the project owner or the resource file owner • The project into which you are releasing must be an open edition. You must be the project owner or if there is already a released edition of the resource, you must own the currently released edition.

RELATED CONCEPTS

Version control for Java program elements
 Baselines, releasing and reloading
 Version control for resource files
 Ownership and team roles - overview
 Workspace

RELATED TASKS

Finding unreleased editions in the workspace
 Managing editions of program elements
 Versioning a program element
 Replacing editions in the workspace (reloading)
 Searching the workspace by edition status, owner, or developer
 Building a team baseline
 Sharing resource files
 Replacing a resource file with the released version

Creating a scratch edition

Scratch editions are private. They reside in the workspace; no one else can see them in the shared repository. In the workspace, you can have scratch editions of projects or packages, open editions of classes contained in scratch editions of packages, and open editions of packages contained in scratch editions of projects.

If you have configured your VisualAge for Java options to show edition names, your scratch editions will be designated with < > around the program element's version name:

```
PackageA <1.0>
PackageB 1.2
```

In the example above, PackageA is a scratch edition that was created from a versioned edition called 1.0. PackageB is not a scratch edition; it is a versioned edition.

You may use a scratch edition to experiment, for example to learn how someone else's code works, or to test a change that you think the program element's owner should make. You can version program elements that are contained in a scratch edition, but you *can not release* them.

To create a scratch edition of a package, do *one* of these things:

- Modify a class contained in a versioned edition of a package, and then save your changes. A scratch edition of the package will be created automatically.
- Replace the edition of a class in a versioned package with another edition of that class. A scratch edition of the package will be created automatically.

To create a scratch edition of a project, do *one* of these things:

- Create a new open edition of a package that is a scratch edition. A scratch edition of the project will be created automatically.
- Create a new open edition of a versioned package contained in a versioned edition of a project. A scratch edition of the project will be created automatically.

To find all of your scratch editions at once, select the **Management Query** tool from the Workspace menu, and specify **Scratched** as one of your search criteria.

RELATED CONCEPTS

Scratch editions
Editions and versioning
Baselines, releasing, and reloading
Ownership and team roles - overview
Workspace

RELATED TASKS

Versioning a program element
Creating an open edition
Releasing a program element
Replacing editions in the workspace (reloading)
Searching the workspace by edition status, owner, or developer

Sharing resource files

In the team development environment, resource files are stored on the team server. The following is a typical development cycle dealing with shared resources:

1. (Optional) The project owner creates resource folders in their workspace to separate the files into groups. For example, the project owner may want to create a folder for each team member.
2. The project owner adds resource files to the project.
3. After the project owner has added all the resource files to the project, the project owner should release any resources that have not been released previously. Any resources that have been previously released appear in the Resources pane of the Repository Explorer Projects page. Ownership cannot be assigned to resource files or folders before they have been released. As well, the other team members cannot load any resource files into their workspaces that have not been released.

4. The project owner transfers ownership of the released files and folders to the team members who will be responsible for releasing changes to the files.
5. The team members then load the project into their workspaces. The project comes with all the resource files (and folders) associated with it, which are stored in the local `project_resources` directory.
6. The team members progress through the normal development cycle, working with program elements and resource files.
7. When they are at a stage that they are ready to share their changes with other team members, team members release their resource files to the server.
8. When team members want to pick up other team members' changes, they replace a resource file with the released copy.
9. When the project owner is ready to set a new baseline, they replace all of the resource files in their workspace with the released copies, and then they version the project. Team members reload the project and start the development cycle again.

RELATED CONCEPTS

Baselines, releasing and reloading
Version control for resource files
Resource files and directories

Creating resource folders

Project owners can create folders for project resource files and assign ownership to these folders. When a developer owns a folder, the developer should own all the project resource files contained in the folder.

To create a resource folder, follow these steps:

1. In the project browser, select the **Resources** tab.
2. Right-click anywhere in the Resources page and select **Add > Folder**.
3. Enter the name of the new folder. You can create a folder that contains a subfolder, by entering the name of the folder followed by a backslash (`\`) and the name of the new folder:
`Newfolder\subfolder\anotherfolder`
4. If you want to add a subfolder to a folder after you have created it, select the folder and select **Add > Folder** from its pop-up menu. Repeat step 5.

The project owner can now assign ownership to the folder (and subfolders) and add resource files to it.

All team members can create folders locally by following the steps above, but they cannot release them or change the owner of them. They can add project resource files to them, but they cannot release the files unless they own the released version of the resource file.

RELATED CONCEPTS

Resource files and directories
Version control for Java program elements
Baselines, releasing and reloading
Version control for resource files

RELATED TASKS

Releasing a program element or resource file
Sharing resource files
Adding resources from the file system
Replacing a resource file with the released version
Changing the owner of a resource file or folder

Adding resource files to the project from the file system

Team members can add, delete, rename, or replace resource files in their workspace but may only perform these same operations on released resources if they are the project owner or the owner of the released resources.

All team members can add resource files from the local file system or network. Files that are added to folders are not automatically assigned the same owner as the folder (if the folder has an owner); you must change their owner (if you have that authority) manually.

You can add resource files to a versioned edition of a project. You cannot release the resources, however, until you create an open edition of the project.

To add resource files to the workspace, follow these steps:

1. In the project browser, select the **Resources** tab.
2. If you want to add the resource files to a folder, first select the folder and select **Add > Resources** from the folder's pop-up menu.
If you do not want to add the resource files to a folder, right-click anywhere in the Resources page and select **Add > Resources** from the pop-up menu.
3. Select the directory that contains the resource files you want to add. Click **OK**.
4. The Add Resources window opens. All the available resources (from both the directory and any subdirectories it contains) are automatically selected. Deselect any resources you do not want to add to your resource folder. Click **OK**.
5. The resources are copied into the local project_resources directory for your project. Any relative paths will be retained.

All team members other than the project owner can edit the resources, but they cannot change the ownership of them or release them. The project owner can release the files, and then change ownership of them.

After ownership has been assigned to the files and folders, the team members can load the project into their workspace.

RELATED CONCEPTS

Resource files and directories
Version control for Java program elements
Version control for resource files
Baselines, releasing and reloading

RELATED TASKS

Releasing a program element or resource files
Sharing resource files
Creating resource folders
Replacing a resource file with the released version
Changing the owner of a resource file or folder

Replacing a resource file with the released version

All team members can replace the current copy of a resource file in their workspace with the most recently released version of it. They can choose to replace all resource files in the project at once, or to replace files individually.

To replace a resource file with the most recently released version of it, follow these steps:

1. In the project browser, select the **Resources** tab.
2. Select the resource that you want to replace. Right-click and select **Replace With > Released Resource**.
3. The released version is imported into your workspace.

RELATED CONCEPTS

Resource files and directories
Version control for Java program elements
Version control for resource files
Baselines, releasing and reloading

RELATED TASKS

Releasing a program element or resource files
Sharing resource files
Creating resource folders
Adding resource files to the project
Changing the owner of a resource file or folder

Building a team baseline

Baselines allow team developers to synchronize the editions that they have in their workspaces, to have a common view of the application, and to catch inconsistencies in their work.

Updating a team baseline

A class owner updates a package baseline by creating a new class in the package, adding an existing class to the package, releasing an updated class into the package, or deleting a class from a package.

A project owner updates a project baseline by creating a new package in the project, adding an existing package to the project, releasing an edition of a package into the project, or deleting a package from the project. A project owner can also update the project baseline by releasing project resource files.

A package owner can also update the project baseline by releasing or deleting a package.

Only open editions of packages and projects can be modified in this manner. When the package or project is versioned, that particular baseline is frozen. When the package or project owner creates a new open edition, the team can start building a new baseline from that point.

Synchronizing with a baseline

Team members load a baseline by adding a project or package to their workspaces. This automatically loads the class versions most recently released into that edition of the project or package from the shared repository.

When team members want to resynchronize with a baseline, they reload by selecting the open edition of the project or package in the Workbench window, and then selecting **Replace with > Released Contents** from the program element's pop-up menu. This action refreshes all of the class editions in the workspace with the versions most recently released into that edition of the package or project.

RELATED CONCEPTS

Team development - overview
Baselines, releasing, and reloading
Team development scenarios - overview
Team development scenario - single package, multiple developers
Team development scenario - multiple packages, multiple developers
Sample life cycle of an application

RELATED TASKS

Creating a class
Creating a package
Adding classes and methods from the repository to the workspace
Adding projects and packages from the repository to the workspace
Releasing a program element
Deleting program elements from the workspace
Replacing editions in the workspace (reloading)
Adding projects and packages from the repository to the workspace

Managing editions of program elements

Edition management in a team development environment can become complex when there are many developers, program elements and editions. The Managing page of the Workbench consolidates information about team editions, and is a convenient place to perform activities such as versioning, releasing, and replacing editions of program elements in your workspace.

The Workspace menu's **Management Query** selection allows you to search the workspace, using different combinations of the following search criteria:

- Search by edition status (open, versioned, unreleased, scratch, or undefined)
- Search by kind of program element (type, package, or project)
- Search by scope (workspace or working set)
- Search by owner
- Search by developer

This is very useful, for example if you are a class developer and you want to know which of your classes you have not versioned, or you are an owner who wants to know which of your program elements you have not released. From the search results list, you can select editions and use their pop-up menus to browse, version, release, delete from the workspace, or replace the edition in the workspace with another edition from the repository. You can also copy from the search results list to the system clipboard.

For more information on using the management query tool, refer to the related information on searching the workspace by edition status.

RELATED CONCEPTS

Editions and versioning
Baselines, releasing, and reloading
Ownership and team roles - overview

RELATED TASKS

Searching the workspace by edition status, owner, or developer
Creating an open edition
Versioning a program element
Releasing a program element
Building a team baseline
Replacing editions in the workspace (reloading)
Changing a program element's owner

Finding unreleased editions in the workspace

Editions that have not yet been released are marked with > beside their names in the **Managing** page of the Workbench window. In the following example, the open edition of PackageA has been released, but the versioned edition of PackageB and the open edition of PackageC have not been released.

```
PackageA (3/28/98 4:21:15 PM)
>PackageB 1.2
>PackageC (4/12/98 10:15:11 AM)
```

You can use the Management Query tool to find all the unreleased editions that are in your workspace. Here is an example of searching for unreleased editions that you own:

1. From the Workspace menu, select **Management Query**.
2. In the Status area of the Management Query window, select **Unreleased**.
3. In the Owners area, select **Current User**.
4. Click the **Start Query**  button.

The side pane of the Management Query window will show the search results:

- For packages, the search results include both open and versioned editions.
- For classes and interfaces, the search results show editions that have been versioned but not released.

RELATED CONCEPTS

Editions and versioning
Baselines, releasing, and reloading

RELATED TASKS

Finding unversioned editions in the workspace
Searching the workspace by edition status, owner, or developer
Releasing a program element
Building a team baseline
Replacing editions in the workspace (reloading)

Finding unversioned editions in the workspace

To tell whether an edition is open or versioned, look at its edition name in VisualAge for Java browsers and in the Workbench window. (If you do not see edition names, click the Show Edition Names  button.)

Open editions have a date and timestamp (in parentheses) showing when they were created. Versioned editions have actual names, which you assign when you version them. In the following example, FredsClass and KimsClass have been versioned, while MyClass is still an open edition.

```
FredsClass 1.3  
KimsClass KP-4Feb-Fix Show()-1.2  
MyClass (03/08/98 12:51:32 PM)
```

You can use the Management Query tool to find all the unversioned editions that are in your workspace. Here is an example of how you would search for unreleased editions that you own:

1. From the Workspace menu, select **Management Query**.
2. In the Status area of the Management Query window, select **Open Edition**.
3. In the Owners area, select **Current user**.
4. Click the **Start Query**  button.

Depending on what other search criteria you selected, the side pane of the Management Query window will show your unversioned projects, packages, and classes.

RELATED CONCEPTS

Editions and versioning
Baselines, releasing, and reloading

RELATED TASKS

Finding unreleased editions in the workspace
Searching the workspace by edition status, owner, or developer
Versioning a program element
Replacing editions in the workspace (reloading)

Viewing a class or interface's developer

Classes and interfaces can only be versioned by the user who developed them. To see who is the developer for a class or interface, select the program element in any browser or in the Workbench window. The developer's name will appear in parentheses in the status line at the bottom of the window. (When you select an edition of a project or package, the status line will show you the *owner's* name.)

You can also see who developed a particular class or interface by selecting **Properties** from the program element's pop-up menu.

RELATED CONCEPTS

Ownership and team roles - overview
Editions and versioning

RELATED TASKS

Changing a program element's owner
Viewing a program element's owner
Searching the workspace by edition status, owner, or developer

Viewing a program element's owner

To see who owns a project or package in the workspace, select the program element in any browser or in the Workbench window. The owner's name appears in the status line at the bottom of the window. (When you select an edition of a *class* or *interface*, the status line will show you the *developer's* name.)

You can also use the Managing page of the Workbench window to find out who owns program elements that are in your workspace:

- When you select a project from the Projects pane, the owner's name appears in the Project Owner pane.
- When you select a package from the Packages pane, the owner marker (>) appears next to the owner's name in the Package Group Members pane.
- When you select a class or interface from the Types pane, the owner's name appears in the Type Owner pane.

You can also see who owns a program element by selecting **Properties** from the element's pop-up menu.

RELATED CONCEPTS

Ownership and team roles - overview
Editions and versioning

RELATED TASKS

Changing a program element's owner
Viewing a class or interface's developer
Searching the workspace by edition status, owner, or developer

Changing a program element's owner

In the team development environment of VisualAge for Java, change control is based on ownership. At times, you will want to reassign ownership of program elements, for example when someone leaves the team.

Here are the rules governing change of ownership for program elements:

Program element	Who is authorized to change the owner	Who is eligible to be the new owner
Class	The administrator or any member of the package group	Any member of the package group
Package	The administrator or the package owner	Any member of the package group
Project	The administrator or the project owner	Anyone on the repository user list
Project resource files	The administrator or the project owner	Anyone on the repository user list

Ownership of project resource files is assigned differently than for all other program elements. Refer to the list of related tasks below for information about changing the owner of project resource files.

To change the owner of a program element:

1. Change workspace owner to the appropriately authorized user, based on the table shown above. If password validation has been enabled, you will be prompted for a password.
2. In the Workbench, click the **Managing** tab. The owner marker (>) appears beside the names of package owners in the Package Group Members pane.
3. Select the program elements whose ownership you want to change. From the pop-up menu, select **Manage > Change Owner**. A list of eligible users will appear.
4. Select the name of the user who will be the new owner, and click **OK**.

The new owner's name will now appear in the appropriate pane at the bottom of the Workbench's Managing window.

RELATED CONCEPTS

Ownership and team roles - overview
Package groups

RELATED TASKS

Viewing a program element's owner
Searching the workspace by edition status, owner, or developer
Changing workspace owner
Adding members to a package group
Enabling password validation - overview
Changing a resource's owner

Adding members to a package group

Before developers can own, create, or release classes in a given package, they must be members of the appropriate package group. Package group members can also change ownership of classes in the package.

Only the package owner can change the membership of the package group. To see who the owner is, select the package on the Managing page of the Workbench window, and look for the owner marker (>) in the Package Group Members pane.

Users must be in the repository user list before they can be added to a package group.

To add users to a package group, do the following steps:

1. In the Workbench, click the **Managing** tab.
2. On the Managing page, select one or more packages. If you select only one, the members of the group will be listed in the Package Group Members pane.
3. Open the pop-up menu for the selected packages, and select **Manage > Add User to Group**. The Add Users dialog box will show a list of repository users who are not yet members of the package group.
4. Select one or more user names, and click **OK**.

The users will be added to the appropriate package groups. To verify, select each package and review the list of names in the Package Group Members pane.

Removing members from a package group

To remove users from a package group:

1. In the Workbench, click the **Managing** tab.
2. On the Managing page, select a package. The Package Group Members pane will list the team members who are currently in the group.
3. Select one or more user names.
4. Open the pop-up menu for the selected names, and select **Remove**.

The users' names will be removed from the list of group members.

RELATED CONCEPTS

Ownership and team roles - overview

Package owner

Package groups

RELATED TASKS

Adding users to the repository user list

Changing a program element's owner

Changing the owner of a resource file or folder

A team member must own a project resource file or folder before they can release it. If they do not own a resource file, they can edit it, but they cannot make their changes available to other team members.

Only the project owner or the owner of the released edition of a resource or folder can assign ownership to that resource file or folder. Team members can add, delete, rename, or replace resource files in their workspace but may only perform these same operations on released resources if they are the project owner or the owner of the affected resources. Only the project owner can release a resource to a project if there is not already a released edition of the resource. After doing so, the project owner may assign ownership to another user.

If the ownership of a resource file or folder is not assigned to anyone, it is implicitly owned by the project owner. If the project owner is changed, then resource files or folders that have not been assigned an owner are automatically owned by the new project owner. If a project owner wants to maintain ownership of a resource file or folder, they must explicitly assign it to themselves.

To assign ownership to a project resource file or folder, you must be working with an open edition of the project. Follow these steps:

1. In the project browser, select the **Resources** tab.
2. The current owner of the resource file or folder is listed in the Owner column. If the resource file or folder has not been released, this column will be blank. You must release the resource before an owner can be assigned.
3. From the resource file's or folder's pop-up menu select **Change Owner**.
4. Select the new owner. Click **OK**.
5. The new owner's name is now listed in the Owner column.
6. When you change the owner of the folder, it does not change the owners of any subfolders or resources contained in it. You must change the ownership of each folder, subfolder, and resource individually.

The new owner can now release different editions of the resource into the project. The project owner can also do this, as the project owner always has the right to modify released resources.

RELATED CONCEPTS

Resource files and directories
Version control for Java program elements
Version control for resource files
Baselines, releasing and reloading

RELATED TASKS

Releasing a program element or resource files
Sharing resource files
Creating resource folders
Adding resource files to the project
Replacing a resource file with the released version

Changing a resource file folder's owner

After a project owner has created a resource file folder, the project owner can change the resource folder's owner. By default, the creator of the folder is the owner.

1. Select and double-click the project that contains the project resources files you wish to work with. The Project Browser opens.
2. Click the **Resources** tab.
3. The current owner of the folder is listed in the **Owner** column.
4. From the folder's pop-up menu select **Change Owner**.
5. Select the new owner. Click **OK**.
6. The new owner's name is now listed in the **Owner** column.
7. When you change the owner of the folder, it does not change the owners of any subfolders or resources contained in it. You must change the ownership of each folder, subfolder and resource individually.

Viewing information about your repository connection

To see details about the repository to which the workspace is currently connected, do the following steps.

1. From the **Window** menu, select **Repository Explorer**. The Repository Explorer window will open.
2. From the **Admin** menu, select **Properties**. The Properties dialog box appears, showing information about the repository file and the repository's contents.

If you are connected to a repository on a server, the Repository field in the Properties dialog box will show the repository name in *server::filename* format. Here is an example:

```
teamserv::team1.dat
```

If you are connected to the local repository, the Repository field in the Properties dialog box will display path information for the .dat file.

RELATED CONCEPTS

Repository

RELATED TASKS

Changing repositories
Connecting to a shared repository

RELATED REFERENCES

Repository files

Changing repositories

The workspace can only be connected to one repository at a time. You can change repositories, for example to browse program elements developed by another team. You can change to another shared repository, perhaps on a different server, or you can change to a local repository on your client workstation.

When you exit the IDE, VisualAge for Java will save the server and directory names for your last connection, in the client's `ide.ini` file. The next time you start the IDE, your workspace will connect to that repository. To confirm which repository you are currently using, select **Properties** from Admin pull-down menu in the Repository Explorer window.

Changing to a shared repository on a server

To connect your workspace to a shared repository on a server:

1. It is recommended that you delete projects from your workspace before connecting to a repository that does not contain those projects.
2. From the Repository Explorer window, select **Admin > Change Repository**. The Select repository dialog box will appear.
3. Select **Use a shared repository with EMSRV server address**. In the entry field, type the IP address or host name of the server where the shared repository resides.
4. Click **Browse**. A second dialog box will appear, listing the repositories (`.dat` files) that are available in that server's working directory.
5. Select the repository file that you want to use, and click **Open**. (To navigate through the server directory structure, double-click entries in the Directories pane.) The first dialog box will reappear, with the name of the repository entered.
6. Click **OK**.
7. If you are not on the user list for the new repository, you will be asked to choose a new workspace owner. (VisualAge for Java compares the unique name, not the full name, when it checks the repository user list.) Select an owner name and click **OK**. If password protection has been enabled on the repository server, you will need to provide the new workspace owner's password.
8. A message will confirm that the workspace is being connected to the new repository. If there are inconsistencies between the workspace and the new repository, a message will instruct you to check the Log. Click **OK** and open the Log window.

If you attempt to browse a program element that does not reside in the new repository, you will see the text "Source code not available" in the browser's Source pane. If you plan to continue working with this repository, select **Save Workspace** from the File menu. This action will ensure that the next time you start the IDE, you will be connected to the same repository and you will have the same editions in your workspace.

Changing to a repository on your workstation

To connect your workspace to a local repository on your own hard drive:

1. It is recommended that you delete projects from your workspace before connecting to a repository that does not contain those projects.
2. From the Repository Explorer window, select **Admin > Change Repository**. The Select repository dialog box will appear.
3. Select **Use a local repository** and click **Browse**. A second dialog box will appear, showing the contents of the VisualAge for Java program directory on your machine.
4. Navigate through the directory structure until you see the repository (.dat file) that you want to use. Select it and click **Open**. The first dialog box will reappear, with the name of the repository entered.
5. Click **OK**.
6. If you are not on the user list for the new repository, you will be asked to choose a new workspace owner. (VisualAge for Java compares the unique name, not the full name, when it checks the repository user list.) Select an owner name and click **OK**.
7. A message will confirm that the workspace is being connected to the new repository. If there are inconsistencies between the workspace and the new repository, a message will instruct you to check the Log. Click **OK** and open the Log window.

If you attempt to browse a program element that does not reside in the new repository, you will see the text "Source code not available" in the browser's Source pane. If you plan to continue working with this repository, select Save Workspace from the File menu. This action will ensure that the next time you start the IDE, you will be connected to the same repository and you will have the same editions in your workspace.

RELATED CONCEPTS

Team client/server configuration
Workspace
Repository
Repository user list
Repository administrator
Workspace owner

RELATED TASKS

Connecting to a shared repository
Changing workspace owner
Adding users to the repository user list
Enabling password validation - overview
Changing the EMSRV working directory
Saving the workspace

RELATED REFERENCES

Repository files
EMSRV startup parameters

Working at a standalone workstation

As a team developer, you may sometimes need to work away from the office. You can do this by creating a repository to use when you are not connected to the LAN.

Here is an example of how you might replicate program elements from the server for use on a home computer. In the example, the home computer has its own local repository and its own copy of VisualAge for Java, Enterprise Edition.

1. While connected to the shared repository, version the packages you need to work with.
2. Export those packages to a temporary repository on the server. Call the new repository `home.dat`. You cannot export any code that is not in the repository.
3. Copy `home.dat` from the server to an offline medium, for instance diskette.
4. Start the IDE on your home computer and connect to your default local repository (`ivj.dat`).
5. Copy `home.dat` onto your home computer and import its packages to `ivj.dat`.
6. Use the Repository Explorer to add the packages to your workspace. (If they are already in your workspace, reload them by selecting **Replace with > Another Edition** from their pop-up menus.)
7. Open a new edition of the packages whose classes you need to change. Modify the classes as required, and then version and release them.
8. Reverse the process to transport your work back to the shared repository:
 - a. On your home computer, version the changed packages and export them to another repository
 - b. Copy the results of the export to an offline medium; copy the portable repository to the server
 - c. Import from it to the shared repository; reload the new versions.

There are variations on this approach:

If the server and the home computer have the appropriate hardware, you can eliminate the two copy steps by exporting to a Zip disk.

If the client is a laptop, you could import your versioned packages directly from the shared repository to a local repository on the laptop's hard drive, and then change to that repository before leaving the office. (If you forget to change repositories while you are still connected to the LAN, you can still connect to your local repository by editing the client's `ide.ini` file. For more information, see the related topic on connecting to a shared repository.)

RELATED CONCEPTS

Repository
Team client/server configuration
The repository server (EMSRV)

RELATED TASKS

Importing from another repository
Exporting to another repository
Changing repositories
Connecting to a shared repository

RELATED REFERENCES

Repository files

Recovering from a server failure

If you are working in the IDE and connected to a shared repository, you may experience the following symptoms after a server failure:

- You are unable to open any new browsers
- You do not receive any confirmation message when you try to save a method
- You get an error message when you try to save the workspace
- You see one or more occurrences of the following error in the Log window:

```
An exception occurred in a system program
Terminating Default
```

These symptoms all occur because your connection to the shared repository has broken. To reconnect to the shared repository after a server failure:

1. Confirm that the server has been restarted.
2. From the **File** menu of any VisualAge for Java window, select **Quick Start**.
3. Select **Repository Management > Change Repository**, and click **OK**. The Select repository dialog box will appear.
4. Select **Use a shared repository with EMSRV server address**.
5. Accept the name of the current server, and click **Browse**. A second dialog box will appear, listing the repositories (.dat files) that are available in that server's working directory.
6. Select the repository file that you were using before the server failed, and click **Open**. The first dialog box will reappear, with the name of the repository entered.
7. Click **OK**.
8. A message will confirm that the workspace is being connected to the repository.

If your workspace is not successfully reconnected to the shared repository, then you must restart the IDE. If this is the case, you may need to re-add some open editions from the repository, or replace editions in the workspace in order to retrieve changes to methods that you had saved.

See the topics listed below for links to more information about workspace recovery.

RELATED CONCEPTS

Team client/server configuration

RELATED TASKS

Saving changes to code

Saving the workspace

Backing up a shared repository

Connecting to a shared repository

Replacing editions in the workspace (reloading)

RELATED REFERENCES

Recovering the workspace

Reinstalling the workspace

Important files to back up

Chapter 4. Team administration

TCP/IP network considerations in team development

In the VisualAge for Java team development environment, all servers and clients are connected by means of a TCP/IP network. VisualAge for Java does not provide the TCP/IP software itself; typically your operating system would provide TCP/IP support.

EMSRV for NetWare requires the NetWare TCP/IP stack (TCPIP.NLM) loaded and configured on the server. When EMSRV for NetWare is started, if this NLM is not loaded, it will automatically be loaded. EMSRV for NetWare will also automatically load the NWSNUT.NLM which is required by the EMSRV for NetWare user interface.

The default limit for client connections to a server is 256. This limit can be changed by using the **-M** parameter of the **emsrv** command. Some TCP/IP stacks will run out of stream sockets before this limit is reached.

RELATED CONCEPTS

Team client/server configuration
The repository server (EMSRV)
Server considerations in team development

RELATED TASKS

Setting up a team server - overview

RELATED REFERENCES

EMSRV startup parameters
EMSRV and TCP/IP

Server considerations in team development

The repository server (EMSRV) must be installed on any computer where one or more shared repositories will reside.

You may have one or more servers in your team development environment. Below are some issues to consider when planning where to install shared repositories and the repository server.

Capacity and availability

Run EMSRV on server-class computers. For optimal availability and performance, servers should be dedicated; that is, shared repositories should not reside on a developer's workstation.

EMSRV uses TCP/IP for its client/server network connections. The default limit for client connections to a server is 256, but some TCP/IP stacks will run out of stream sockets before this limit is reached.

Number and placement of shared repositories

Repository planning considerations, including information on file sizes, are covered as a separate topic in this documentation.

Ease of management

It is simpler to manage one server than multiple servers. For example, it is easier to design a backup strategy for one server.

Supported operating systems

The following operating systems are supported as VisualAge for Java servers:

- Windows NT Workstation Version 4.0
- Windows NT Server Version 4.0
- Windows 2000 Professional
- Windows 2000 Server
- Windows 2000 Advanced Server
- OS/2 Warp Version 4.0
- OS/2 Warp Server for e-business
- AIX Version 4.3.2, 4.3.3
- HP-UX Version 10.20, 11.0
- Sun Solaris Version 2.6, 7.0
- Novell NetWare Versions 4.2, and 5.1
- Red Hat Linux 6.1, 6.2

RELATED CONCEPTS

Team client/server configuration
The repository server (EMSRV)
Network considerations in team development
Server security
Number and placement of shared repositories

RELATED TASKS

Setting up a team server - overview
Optimizing server performance

RELATED REFERENCES

Repository files

Server files and directories

Server files

The team development server requires the following files:

- The executable program for the repository server (emsrv.exe, emsrv.nlm, emsrv, or emsrv.shadow)
- The repository server log file (default name emsrv.log)
- One or more shared repositories (default name ivj.dat)

The repository server program must be installed on any computer where a shared repository will reside. The EMADMIN utility (emadmin.exe) may also be installed for local monitoring or stopping of the repository server. As project resource files are added to projects and versioned, subdirectories are added to the directory where the shared repository resides.

It is not necessary to install the IDE on the server.

Server directories

VisualAge for Java repositories must reside on the same system as EMSRV; remote file systems are not supported.

The server files may reside in any directory on the server. It may be convenient to keep them together in the same directory, for the following reasons:

- If repository files are in the same directory, and if that directory is the EMSRV working directory, then team members do not have to provide path information to connect, export, import, or change repositories. See the related topics below, for links to more information on the EMSRV working directory.
- If repository files are all in one directory, it is easier to write a script to back them up.
- By default, the emsrv.log file is written in the EMSRV working directory. (To override the default, use the **-lf** startup parameter of the **emsrv** command.)

 If your server has multiple devices, you may have to determine where your files reside and specify them with the **-xd** option on the EMSRV command line.

File access rights

All files (including resource files) created by the repository server are owned by the EMSRV user. The repository server has the file access rights of the EMSRV user. To ensure the integrity of shared repositories, you may wish to restrict rights for repository files to the EMSRV user.

Planning for repository growth

In the team development environment, it is quite common for the shared repository to reach a size of several hundred megabytes. Project and package owners should be encouraged to purge program elements that are no longer needed, so the administrator can compact the repository. Projects can be purged as well as packages.

Users who do not own any projects, packages or classes are not copied when the repository is compacted. After the compaction, the Administrator will have to re-create them.

RELATED CONCEPTS

Team client/server configuration
Repository
The repository server (EMSRV)
Number and placement of shared repositories
Server considerations in team development
Server security
EMSRV user

RELATED TASKS

Setting up a team server - overview
Changing the EMSRV working directory
Creating a repository
Compacting a repository

RELATED REFERENCES

Repository files
EMSRV startup parameters

Server security

There are different levels of security that you can implement for VisualAge for Java servers.

Password validation

If password validation is enabled on the repository server, team members must provide valid passwords to do these tasks:

- Connect to a repository managed by that server
- Change workspace owner

In all cases, the workspace owner's privileges determine what the developer who is using that workspace can do. For example, if a developer connects to the shared repository as Administrator, then that developer can purge any program element or change ownership of any program element.

The VisualAge for Java default is to run with *no* password validation on the repository server.

File security

The server comprises the following files:

- The executable file for the repository server (emsrv.exe, emsrv.nlm, or emsrv)
- The server log file (default name emsrv.log)
- One or more shared repositories (default name ivj.dat)

The only user who requires access to these files is the EMSRV user, under whose authority the repository server runs. (The repository server handles input/output requests on behalf of the clients.) You can add an additional measure of security by restricting access to only the EMSRV user. Do this using the standard facilities of your server operating system.

Physical security

For optimal performance and availability, the server should be a dedicated machine, not a developer's workstation. As with any kind of server, you should exploit hardware and software features like keyboard passwords, and you should consider the benefits of placing the server in a physically secure area.

RELATED CONCEPTS

Team client/server configuration
EMSRV user
Server files and directories
Server considerations in team development
Repository administrator

RELATED TASKS

Setting up a team server - overview
Enabling password validation - overview
Adding users to the repository user list

RELATED REFERENCES

Repository files

Number and placement of shared repositories

In the team development environment, there will be at least one shared repository. You may decide to have more than one, residing on the same server or on different servers. Team members can change from one shared repository to another, but they can only be connected to one repository at a time.

Below are some issues to consider when deciding how many shared repositories the team will use, and where they will reside.

Performance

There are several factors that affect the time required for a client to connect to a shared repository, and to work with the repository after connecting. Those factors include the number of clients using the same server, and the speed of the server itself.

Repository growth and maximum size

The source code repository, `ivj.dat`, will grow as team members experiment and take advantage of incremental development. It is not uncommon to have a repository that is several hundred megabytes in size. The administrator can reduce the size by compacting the repository after team members have purged any package editions they no longer need.

You should be careful to try and avoid excess, unnecessary repository growth. For example, if all your team members import a class library, this will take up more space in the repository than if one team member imports it and the rest load it into their workspace from the repository.

 The maximum repository size is 2 gigabytes.

 The maximum repository size is 4 gigabytes.

 The maximum repository size is 2 gigabytes for FAT drives, 4 gigabytes for FAT32 drives, and 16 gigabytes for NTFS drives.

   The maximum repository size is 16 gigabytes.

 The maximum repository size is 2 gigabytes.

Ease of management

It is simpler to manage one shared repository than multiple repositories, and it is simpler to manage one server than multiple servers. For example, if everything resides in one repository then it is easier to ensure your developers have access to all of the program elements that they require. Trade-offs include performance and the effort required to divide a repository later.

Team and project organization

If most of your developers need access to each others' classes, then ideally they should use a single shared repository. This must be weighed against performance and server capacity considerations.

If your developers are working on very different projects and are not dependent on each other's classes, then you should consider designing your server and project structure so they are working with different repositories. You might start by having

two or more repositories on one server to begin with, and then move them to different servers later if you need more capacity.

Effort required to divide a repository

If you decide, after several months of development, that your team should be working with multiple repositories, then the administrator will have to divide the repository that everyone has been using. One way to do this is to create a second repository, export selected program elements into it, and purge those program elements from the first repository. Another approach is to copy the existing repository, and purge what is not needed from both the original and the copy.

In any case, you must identify which group of developers needs which classes. This process may require reorganization of projects and packages, and may involve some trial and error.

Server capacity

Considerations for server selection are covered as another topic.

Restrictions

Repository files must reside on the same system as the EMSRV executable program. The repository server does not support remote repositories.

RELATED CONCEPTS

Repository
Team client/server configuration
Server considerations in team development
Server files and directories
Team and project organization
Repository administrator

RELATED TASKS

Setting up a team server - overview
Creating a repository
Compacting a repository
Purging program elements from the repository
Dividing a repository
Optimizing server performance

RELATED REFERENCES

Repository files

Team and project organization

Before the team starts a new application development project, you should discuss the following issues:

- How to organize the work
- How to structure the VisualAge for Java project, packages and classes
- Who should own each project, package, and class

After the design is complete, the appointed project owner should set up the project for the rest of the team.

Organization and assignment of classes

Once you know what classes will be required and how they will be grouped into packages, consider the following issues:

Who will belong to each package group? These are the developers who can own, create, change, and delete classes in the package. The package group members are “trusted peers”; they can take ownership of any class in the package, at any time.

Which developer is likely to make the most changes in each class? That developer should probably own that class.

Are the developers dependent on classes in other packages? If not, perhaps you should ask the administrator to create a separate shared repository for this project. This would be simpler than dividing a large repository later.

Organization and assignment of packages and projects

Here are some other questions to answer before setting up the project in VisualAge for Java.

Who will own the package? The package owner appoints members to the package group, monitors whether they have versioned and released their classes, and decides when the package is stable enough to release or version. The package owner should be available to re-open the package after it has been versioned so that class owners are not restricted to working in scratch editions. The package owner probably owns some classes in the package.

Who will own the project? The project owner sets up the project by adding packages to it and assigning them to owners. The project owner coordinates the activities of the package owners and versions the entire project. This person, who might have a job title like project leader or architect, probably also owns classes.

For an illustration of how the project manager would proceed to set up the project and how the team members would perform their roles, see the sample application life cycle that is described as a related topic.

For more a more detailed discussion of setting up the team development environment, see the IBM redbook, *VisualAge for Java Enterprise Team Support* (SG24-5245-00). For information on VisualAge for Java books, select the **Library** link at <http://www.software.ibm.com/ad/vajava/>.

RELATED CONCEPTS

- Ownership and team roles - overview
- Package groups
- Sample life cycle of an application
- Number and placement of shared repositories

RELATED TASKS

- Creating a project
- Creating a package
- Adding users to the repository user list
- Adding members to a package group
- Changing a program element's owner

Setting up a team server - overview

The administrator performs the following tasks to prepare a server for the team development environment. For more detailed information, see the links to related information at the end of this document.

1. Plan the server installation. Review the following subjects:
 - Network considerations
 - Server considerations
 - Number and placement of shared repositories
 - Server file and directory structures
 - Server security
2. Plan the team and project organization.
3. Install and configure TCP/IP on the server.
4. Install the VisualAge for Java server code. This action will install the EMSRV executable program and a repository called ivj.dat on the server. Refer to the Installation and Migration guide on the product CD for information on how to install the product.
5. Decide which user account will be used to start EMSRV, and create the user account if necessary.
6. **WIN** Give the EMSRV user the necessary Windows privileges.
NetWare Give the EMSRV user appropriate rights to any paths where repositories can reside.
7. **WIN** Install EMSRV as a service in the Windows registry. (As an alternative, you may wish to start EMSRV from a command line until you are familiar with the startup parameters, and then install it in the registry later.)
8. If native password validation will be used, create the user accounts on the server. If VisualAge for Java password validation will be used, create the passwd.dat file in the EMSRV working directory.
9. If it has been decided that more than one shared repository is necessary, create the additional repositories. For example, you might make copies of ivj.dat with different names, in the same directory.
10. Start EMSRV.
11. From a VisualAge for Java client, connect to the shared repository as Administrator, and add users to the repository list. If there will be multiple shared repositories, do this for each one.

RELATED CONCEPTS

Team client/server configuration
The repository server (EMSRV)
Network considerations in team development
Server considerations in team development
Number and placement of shared repositories
Server files and directories
Server security
Team and project organization
EMSRV user

RELATED TASKS

Installing EMSRV as a service in the Windows registry
Removing EMSRV from the Windows registry

Authorizing the EMSRV user (Windows)
Starting the repository server on Windows
Starting the repository server on OS/2
Starting the repository server on AIX, HP-UX, Solaris, or Linux
Starting the repository server on NetWare
Enabling password validation - overview
Connecting to a shared repository
Adding users to the repository user list
Creating a repository

RELATED REFERENCES

Repository files
EMSRV and TCP/IP
EMSRV startup parameters

Installing EMSRV as a service in the Windows registry

You can install EMSRV in the Windows NT or Windows 2000 registry, if you prefer to start EMSRV as a service rather than from a command prompt.

There are two advantages to installing EMSRV as a service:

- You can specify automatic startup so that EMSRV will start whenever the repository server is booted.
- You can specify the default settings that you want EMSRV to use. For example, you might want to ensure that password validation is always enabled.

If EMSRV is started as a service, the default EMSRV working directory is the Windows NT or 2000 system32\ directory. It is recommended that you *change this default* by using the **-W** parameter when you install EMSRV as a service in the Windows registry.

To install EMSRV as a service:

1. From a command prompt, change to the directory where the emsrv executable program is installed.
2. Issue `emsrv -install [parameter2] [parameter3] ...`. The first parameter must be `-install`; the others are the EMSRV startup parameters that you have chosen for your environment.

Here is an example:

```
emsrv -install -u joe -p donttell -W j:\sharedrep -rn
```

This example installs EMSRV as a service in the Windows registry, with joe as the EMSRV user name and donttell as joe's password. By default, the EMSRV working directory will be j:\sharedrep and native password validation will be enforced.

A message will confirm that EMSRV has been installed.

3. Steps a and b are slightly different for Windows NT and Windows 2000.
 - a) From the Windows NT Control Panel, double-click **Services**. The Services dialog box will appear. Select **EMSRV** from the list of services.
 - b) From the Windows 2000 Control Panel, double-click **Administrative Tools**. Double-click **Services**. Double-click EMSRV.

4. In the Startup Parameters text box, type the EMSRV startup parameters that you want to use. If you are specifying the working directory for EMSRV to use, you must type an *extra backslash* for each backslash in the path. Here is an example:

```
-u emsrvacc -p secret -W d:\\javateam
```

5. Click **Start**. A message will appear, informing you that EMSRV is starting.

EMSRV is now installed as a service in the registry and the necessary DLLs have been copied to the system directory. The parameters that you provided will be used, by default, whenever EMSRV is started. You can also override or add to these parameters if you start EMSRV manually from the **Services** icon of the Windows Control Panel.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Authorizing the EMSRV user (Windows)
Starting the repository server on Windows
Removing EMSRV from the Windows registry
Enabling password validation - overview
Enabling password validation with the passwd.dat file
Changing the EMSRV working directory
Displaying active EMSRV settings

RELATED REFERENCES

EMSRV startup parameters

Removing EMSRV from the Windows registry

Warning: Removing EMSRV from the registry will also stop EMSRV if it is running. You should *not* stop EMSRV until all clients have disconnected. Otherwise, developers may not be able to save their workspaces.

To remove EMSRV from the registry on Windows NT or Windows 2000:

1. Make sure no team members are currently connected to the repository server. To confirm this, issue `emadmin list` from the command prompt of any network-attached workstation where the EMADMIN utility is installed.
2. To stop the repository server, enter `EMADMIN stop`.
3. Enter `EMSRV -remove .`

EMSRV will be removed from the registry. To confirm this, double-click the **Services** icon in the Windows NT Control Panel. For Windows 2000, double-click the **Administrative Tools** icon, then the **Services** icon in the Control Panel. EMSRV no longer appears on the list of services.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Stopping the repository server
Installing EMSRV as a service in the Windows NT or 2000 registry

RELATED REFERENCES

The EMADMIN utility - overview
The EMADMIN stop command
The EMADMIN list command

Authorizing the EMSRV user (Windows)

On Windows NT or Windows 2000, the person who starts the repository server (EMSRV) must provide the name of a user account under whose privileges the repository server will run. This is known as the EMSRV user. The EMSRV user must meet the following two requirements:

- The user must be a member of the Windows Administrators group
- The user must be granted the Windows advanced user right, “Act as part of the operating system”.

This is necessary to allow EMSRV to authenticate user accounts. Authentication for the EMSRV account takes place when EMSRV is started and stopped. Other accounts users may also be authenticated if EMSRV is run with the -rn option to authenticate clients.

To authorize the EMSRV user to start the repository server on Windows NT:

1. Log in to Windows NT as an Administrator.
2. From the Start menu, select **Programs > Administrative Tools (Common) > User Manager**. The User Manager dialog box will appear.
3. If desired, create a new user to be the EMSRV user.
4. Still in the User Manager dialog box select **User Rights** from the **Policies** menu. (It does not matter which user’s name is selected when you do this.) The User Rights Policy dialog box will appear.
5. Select **Show Advanced User Rights** and then click the down arrow to see the Right pull-down list. The list should now include “Act as part of the operating system”.
6. Select **Act as part of the operating system** from the list, and click **OK**.
7. The Grant To pane will list the users who currently have this privilege. Click **Add**. The Add Users and Groups dialog box will appear.
8. Click **Show Users**. Scroll down the list of users in the Names pane and select the **EMSRV user** from the list.
9. Click **Add**. The EMSRV user’s name will appear in the Grant To pane at the bottom of the Add Users and Groups dialog box. Click **OK**.
10. The EMSRV user’s name now appears in the Grant To pane of the User Rights dialog box. Click **OK**.

To authorize the EMSRV user to start the repository server on Windows 2000:

1. From the Start menu, select **Settings > Control Panel**. Double-click **Administrative Tools**.
2. Double-click **Local Security Policy**. The Local Security Settings window opens.
3. Double-click **Local Policies**. From the list that opens, select **User Rights Assignment**. In the right-hand pane, you will see a list of all the operating system properties.

4. Double-click **Act as part of the operating system**. The Local Security Policy Setting dialog opens.
5. Click **Add**. The Select Users or Groups dialog opens.
6. Select the EMSRV user and click **Add**. Click **OK**.
7. The dialog closes. The EMSRV user appears in the **Local Security Policy Setting** dialog. Click **OK**.
8. Log off as Administrator and log back on as Administrator. You must do this, otherwise, the property will not be set properly.
9. To verify the property has been set properly, follow steps 1 to 4 and ensure that both the "Local Policy Setting" and "Effective Policy Setting" check boxes are selected for the EMSRV user.

The EMSRV user now has the Windows operating system privileges needed to start the repository server. If the user can not start EMSRV successfully, shut down Windows on the server and reboot the machine. It is sometimes necessary to take this action before changes in Windows privileges take effect.

Tip: For Windows 2000 Server, the Local Security Policy may be overridden by the Domain Controller Security Policy or the Domain Security Policy. User rights for the domain controller or domain may need to be set in order for the Effective Policy Setting in the Local Security Policy to appear checked.

For Windows 2000 Advanced Server, there is no Local Security Policy. Assign the right for the Domain Controller Security Policy and Domain Security Policy as necessary.

Authentication for the EMSRV account is slightly different when authenticating clients. In each case, additional user rights may be required for successful operation:

- To start and stop EMSRV as a service, both the account from which EMSRV is started and the EMSRV account must have the Log on as a service user right assigned. For each account, you must also ensure that the Deny logon as a service right is not set.
- To start and stop EMSRV interactively or from a batch file, both the account from which EMSRV is started and the EMSRV account must have the Log on locally user right. For each account, you must also ensure that the Deny logon locally right is not set.
- Each client authenticated by EMSRV must have the Access this computer from the network right set for the account used by that client. For each account, you must also ensure that the Deny access to this computer from the network right is not set.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Starting the repository server on Windows
Starting the repository server on OS/2
Starting the repository server on AIX, HP-UX, Solaris, or Linux
Starting the repository server on NetWare

Starting the repository server on Windows

As the administrator, you must start the repository server (EMSRV) on Windows NT and Windows 2000, before clients can connect to shared repositories. You can start EMSRV from a command prompt or as a service.

A full list of the EMSRV startup parameters is available from a reference link provided at the end of this topic. At a minimum, you must use the **-u** and **-p** parameters, to specify the EMSRV user's name and password.

Prerequisite

The EMSRV user must be part of the Windows Administrators group, and requires the advanced user right, "Act as part of the operating system".

Starting EMSRV from a command prompt

To start EMSRV:

1. Change to the directory where the `emsvr` executable program is installed. EMSRV *must* be started from this directory.
2. Enter the `emsvr` command with the desired startup parameters. Here is an example:

```
emsvr -u emsvracc -p secret -W d:\javateam
```

starts the repository server under the authority of a user called `emsvracc`. The user's password is `secret`. The working directory where EMSRV will write its log and where it will search for repositories is `d:\javateam`.

You cannot qualify the path name in any way when you start EMSRV from the command line. For example:

```
.\emsvr -u emsvracc -p secret -W d:\javateam
```

will not work.

Messages will be logged in the EMSRV log and in the DOS Command Prompt window, confirming that EMSRV has started and listing the parameters that are in effect.

Starting EMSRV as a service

EMSRV must be installed in the Windows NT or 2000 registry before you can start it as a service. For installation instructions, refer to the related tasks listed below.

When EMSRV is started as a service, the default EMSRV working directory is the Windows NT or 2000 `system32\` directory. It is recommended that you change this default by using the **-W** parameter when starting EMSRV as a service.

If you have installed EMSRV in the registry as an automatically started service, it will start whenever the Windows NT or 2000 operating system is restarted. If you have installed EMSRV in the registry as a manually started service, you can start it by following these steps:

1. Steps a and b are slightly different for Windows NT and Windows 2000.
 - a) From the Windows NT Control Panel, double-click **Services**. The Services dialog box will appear. Select **EMSRV** from the list of services.
 - b) From the Windows 2000 Control Panel, double-click **Administrative Tools**. Double-click **Services**. Double-click **EMSRV**.

- In the Startup Parameters text box, type the EMSRV startup parameters that you want to use. If you are specifying the working directory for EMSRV to use, you must type an *extra backslash* for each backslash in the path. Here is an example:

```
-u emsrvacc -p secret -W d:\\javateam
```

- Click **Start**. A message will appear, informing you that EMSRV is starting.

Potential problems

The following table summarizes some errors that you may see when starting EMSRV as a Windows NT or 2000 service.

Problem	Recommended action
An internal Windows NT or Windows 2000 error has occurred. Error message appears at startup.	<ul style="list-style-type: none"> Verify that the startup parameters have been entered correctly: <ul style="list-style-type: none"> Were the EMSRV user's name and password correctly specified? If you specified an EMSRV working directory, did you include the extra backslashes in the path? Verify that the EMSRV user's password has not expired, by logging in to Windows NT or 2000 as that user. Verify that the EMSRV user is a member of the Administrators group, and has the Windows advanced user right, "Act as part of the operating system".
Normal EMSRV (not a service) hangs on startup.	A pathname cannot be specified for the executable in Windows NT or 2000. It must be run in the directory of the executable. It will not run from a different directory.
The specified service is disabled and cannot be started error message appears at startup.	<p>The service may have been disabled.</p> <p>For Windows NT:</p> <ol style="list-style-type: none"> From the Control Panel, double-click Services. Select EMSRV from the list of installed services. Click HW Profiles. If EMSRV's original configuration shows as disabled, click Enable. <p>For Windows 2000:</p> <ol style="list-style-type: none"> From the Control Panel, double-click the Administrative Tools icon. Double-click Services. Double-click EMSRV. Click Log On. If EMSRV's original configuration shows as disabled, click Enable.

<p>The start button is disabled.</p>	<p>The service could already be running on the local machine. Otherwise, the service could be disabled.</p> <p>For Windows NT:</p> <ol style="list-style-type: none"> 1. From the Control Panel, double-click Services. 2. Select EMSRV from the list of installed services. 3. Click Startup to open the Startup Options window. 4. Select Manual for Startup Type. <p>For Windows 2000:</p> <ol style="list-style-type: none"> 1. From the Control Panel, double-click the Administrative Tools icon. 2. Double-click Services. Double-click EMSRV. 3. From the Startup Type drop down list, select Manual.
<p>The service did not start due to a logon failure error message appears at startup.</p>	<p>An invalid password was entered in the Startup Window of the Service Control Panel. Open the Startup window and change the password or start as the System Account.</p> <p>To open the Startup window on Windows NT:</p> <ol style="list-style-type: none"> 1. From the Control Panel, double-click Services. 2. Select EMSRV from the list of installed services. 3. Click Startup to open the Startup Options window. 4. Change the password. <p>To open the Startup window on Windows 2000:</p> <ol style="list-style-type: none"> 1. From the Control Panel, double-click the Administrative Tools icon. 2. Double-click Services. Double-click EMSRV. 3. Click Log On. 4. Change the password.
<p>The process terminated unexpectedly error message appears at startup.</p>	<p>EMSRV may have already been started from a command prompt. Use the emadmin stat command to check the status, or use the Windows NT or 2000 Task Manager to look for EMSRV on the list of running processes.</p>

Confirming that EMSRV is running

To verify that EMSRV is running, issue the **emadmin stat hostname** command from any workstation where the EMADMIN utility is installed.

RELATED CONCEPTS

The Repository Server (EMSRV)
EMSRV User

RELATED TASKS

Setting up a team server - overview
Installing EMSRV as a service in the Windows registry
Authorizing the EMSRV user (Windows)
Stopping the repository server
Enabling password validation - overview
Changing the EMSRV working directory

RELATED REFERENCES

EMSRV startup parameters
The EMADMIN utility - overview

Starting the repository server on OS/2

As the administrator, you must start the repository server (EMSRV) before clients can connect to shared repositories.

To prevent unauthorized remote shutdown of EMSRV, provide a password when you start it. For a full list of the EMSRV startup parameters that are available, refer to the reference link provided at the end of this topic.

To start EMSRV from an OS/2 command line:

1. Change to the directory where the `emsrv` executable program is installed.
2. Type the `emsrv` command with the desired startup parameters. Here is an example:

```
emsrv -p secret -W d:\javateam
```

where `secret` is the password that would be required to stop the repository server remotely and `d:\javateam` is the working directory where EMSRV will look for shared repositories.

Press Enter.

Messages will be logged in the EMSRV log and in the OS/2 Command Prompt window, confirming that EMSRV has been started.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Setting up a team server - overview
Stopping the repository server
Enabling password validation - overview

RELATED REFERENCES

EMSRV startup parameters
The EMADMIN utility - overview

Starting the repository server on AIX, HP-UX, Solaris, or Linux

As the administrator, you must start the repository server (EMSRV) before clients can connect to shared repositories.

A full list of the EMSRV startup parameters is available from a reference link provided at the end of this topic.

If your server has multiple devices, you may have to determine where your files reside and specify them with the `-xd` option on the EMSRV command line.

AIX You must start EMSRV at its root if you do not want to use password checking. To be able to start EMSRV as any user, change the file permissions by using the following commands:

```
chmod 4775 emsrv
chown root emsrv
```

To start EMSRV:

1. Log in as the EMSRV user.
2. Change to the directory where the emsrv executable program is installed.
3. **HP-UX** If your system is using shadow passwords, issue the **emsrv.shadow** command with the desired startup parameters.

AIX **SOLARIS** **LINUX** If your system is using shadow passwords, issue the **emsrv** command with the desired startup parameters. You do not need to use the **emsrv.shadow** command.

If your system is not using shadow passwords, issue the **emsrv** command instead.

Here is an example:

```
emsrv.shadow -w -lc -lflogfile
```

This example starts the repository server on an operating system that uses shadow passwords. Lock tracking is turned on, and messages are logged both to the system console and to a file called logfile.

Press Enter.

EMSRV will start on the server. To verify, issue the **emadmin stats hostname** command from any workstation where the EMADMIN utility is installed.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Setting up a team server - overview
Enabling password validation - overview
Stopping the repository server

RELATED REFERENCES

Starting the repository server on NetWare

As the administrator, you must start the repository server (EMSRV) before clients can connect to shared repositories.

For information on startup parameters, refer to the “EMSRV startup parameters” file. At a minimum, you must specify the EMSRV user’s name and password, and a working directory. If you do not enter these three parameters at the console, you will be prompted as the NLM loads.

To get the NLM to load automatically when the NetWare file server is rebooted, you can add an appropriate command line to the file server autoexec.ncf file.

To load the EMSRV NLM, type `load emsrv` (with the desired startup parameters) at the NetWare console, and press Enter. The EMSRV for NetWare menu will appear at the server console.

Here is an example of a command for starting EMSRV from the server console, on a server that uses the Novell Directory Services (NDS) NLM:

```
load emsrv -u emsrvacc -p secret -W volname:\path -rn -SC nyc
```

The above example loads the NLM with the account name `emsrvacc` and password `secret`. The working directory where EMSRV will write its log and look for shared repositories is `volname:\path`. The `-rn` parameter indicates that native password validation should be enforced when users connect to the shared repository. Because native password validation is being used and because the server is running EMSRV for NetWare (NDS), the `-SC` parameter is also provided to specify the NDS context (`nyc`) for the network login names that the administrator supplied when adding users to the VisualAge for Java repository user list.

If the EMSRV for NetWare users are found in a container other than `[root]`, we advise the Network Administrator to use the `-SC` option to set the context to the container object containing the users.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Setting up a team server - overview
Changing the EMSRV working directory
Enabling password validation - overview
Enabling native password validation
Adding users to the repository user list
Changing EMSRV settings (NetWare)
Stopping the repository server

RELATED REFERENCES

EMSRV startup parameters
The EMADMIN utility - overview

Stopping a client connection

As the administrator, you may want to stop one or more client connections to the repository server, for example in preparation for backing up the shared repository.

Warning

Stopping a team client's connection to the server could corrupt that client's workspace, if the client was in the process of loading code from the repository. To see whether clients are active, use the **emadmin list** command to check the last time a request was issued.

Stopping a client connection using the EMADMIN utility

To stop a client's connection to the repository server:

1. Issue the **emadmin list** command to display the list of active connections. You will see information similar to the following:

```
EMSRV Connection list for: localhost
      Active Last
  ID IP Address Locks Request Library
-----
  0 9.21.35.196 0   19:37:08 ivj.dat
  1 9.25.32.196 0   18:12:19 ivj.dat
```

2. Use the ID information you have just obtained to identify the **connection number** that you want to stop. In the example above, you may decide that connection 1 can be terminated.

3. Continuing with the example, to stop connection 1 issue this command:

```
emadmin stop -k1
```

You will be prompted for the EMSRV user's password.

The client's connection to the repository server will be stopped. You can verify this by issuing the **emadmin list** command again.

NetWare Stopping a client connection from the NetWare console

From the EMSRV for NetWare menu, select **View Connections**. To display statistics for a particular client connection, select it from the list and press **Enter**. To terminate that connection, press **Delete**. Press **Esc** to return to the menu.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Displaying server connections
Stopping the repository server

RELATED REFERENCES

The EMADMIN utility - overview
The EMADMIN stop command
EMSRV and TCP/IP

Stopping the repository server

Warning

To ensure developers have had a chance to save their work, do not stop EMSRV while team clients are still connected. To display active connections, issue the **emadmin list** command.

Stopping EMSRV remotely

You can stop EMSRV from any network-attached workstation that has the EMADMIN utility installed. To stop the repository server remotely, use the **emadmin stop** command:

```
emadmin stop [-p password] [-h host]
```

A message will inform you that the server has been scheduled to stop.

OS/2 If a password was used to start the server, you must provide the same password to stop it.

WIN **AIX** **HP-UX** **SOLARIS** **NetWare** **UNIX** The password is the EMSRV user's password. If you do not provide the **-p** parameter, you will be prompted for it. If the EMSRV user does not have a password, issue **emadmin stop -p** with no *password* argument.

HP-UX If the operating system on the server uses shadow passwords, remote shutdown will fail unless EMSRV was started with the **emsrv.shadow** command.

WIN Stopping EMSRV from the control panel

If EMSRV has been installed as a service in the Windows registry, you can stop it from the Control Panel on the server:

1. Double-click **Services**.
2. Select **EMSRV** from the list of services.
3. Select **Stop**.

NetWare Unloading EMSRV from the NetWare console

You can not unload EMSRV for NetWare using the **unload** command from the server console. You can use the EMADMIN utility as described above, or you can use the EMSRV for NetWare menu.

To stop from the menu, select **Shutdown EMSRV** and provide the EMSRV user's password when prompted.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Installing EMSRV as a service in the Windows registry
Displaying server connections
Stopping a client connection

RELATED REFERENCES

The EMADMIN utility - overview
The EMADMIN stop command
EMSRV startup parameters

Displaying server connections

To display active connections to the repository server, for example when preparing to stop EMSRV, issue **emadmin list** from a command prompt. You can use the **-s** and **-l** parameters to display statistics and active locks for a particular connection.

EMSRV must be running on the server in order for you to communicate with it using the EMADMIN utility.

NetWare From the EMSRV for NetWare menu, select **View Connections**. To display statistics for a particular connection, select the connection from the list and press Enter. Press Esc to return to the menu.

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Displaying server statistics
Stopping a client connection

RELATED REFERENCES

The EMADMIN utility - overview
The EMADMIN list command

Displaying server statistics

The **emadmin stat** command displays statistics for the repository server, covering the time period since EMSRV was last started. This provides information such as the following:

- Elapsed time since the repository server was started
- Number of connects and disconnects
- Reads, writes, and locks
- Packets sent and received
- The EMSRV working directory

EMSRV must be running on the server in order for you to communicate with it using the EMADMIN utility.

NetWare From the EMSRV for NetWare menu, select **EMSRV Statistics**. (If EMSRV Statistics has already been selected, select it again to force a screen update.) This will display statistics covering the time period since the EMSRV NLM was loaded. Press Esc to return to the menu.

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Displaying server connections
Changing the EMSRV working directory

RELATED REFERENCES

The EMADMIN utility - overview
The EMADMIN stat command

Displaying active EMSRV settings

To display the options that are currently in effect for the repository server, issue the **emadmin opts** command from a command prompt. This will display information such as the following:

- The EMSRV working directory
- Whether password validation has been enabled
- What level of messages are being logged
- The name of the log file
- The maximum number of concurrent connections allowed
- The threshold for free disk space on the repository server

EMSRV must be started on the server in order for you to communicate with it using the EMADMIN utility.

NetWare From the Menu console, select **Change Settings**. A form will appear, showing you the current EMSRV settings.

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Changing EMSRV settings (NetWare)
Changing the EMSRV working directory

RELATED REFERENCES

The EMADMIN utility - overview
The EMADMIN opts command
EMSRV startup parameters

Displaying EMSRV messages

When debugging an EMSRV problem, it is often useful to watch the messages that get written to the log or the server console, as they occur.

If you are not sure where EMSRV is currently logging, issue the **emadmin opts** command. This will tell you the name of the log file, whether logging to the server console is also enabled, and what the current logging level is.

NetWare To display messages logged by EMSRV, select **View Message Screen** from the EMSRV for NetWare menu. To return to the menu, press Esc.

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Setting EMSRV message logging options

RELATED REFERENCES

The EMADMIN utility - overview

The EMADMIN opts command

EMSRV startup parameters

Changing the EMSRV working directory

The EMSRV working directory is the default directory that the repository server uses to locate shared repositories when, for example, a user is changing repositories or exporting to another repository. For ease of use, it is recommended that you store all shared repositories in the EMSRV working directory. This allows team members to find shared repositories without providing path information.

WIN If EMSRV is started as a service, the default EMSRV working directory is the Windows NT or 2000 system32\ directory. It is recommended that you change this default by using the **-W** parameter when you install EMSRV as a service in the Windows registry.

WIN If EMSRV is started from a command prompt, the default EMSRV working directory is the directory where emsrv.exe is installed. To change the default, use the **-W** parameter of the **emsrv** command when starting the repository server.

OS/2 By default, the EMSRV working directory is the directory where emsrv.exe is installed. To change the default, use the **-W** parameter of the **emsrv** command when starting the repository server.

NetWare By default, the EMSRV working directory is SYS:\. To change the default, use the **-W** parameter of the **emsrv** command when starting the repository server.

AIX **HP-UX** **LINUX** **SOLARIS** The EMSRV working directory is the current working directory at the time that EMSRV is started.

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Installing EMSRV as a service in the Windows registry

Starting the repository server on Windows

Starting the repository server on OS/2

Starting the repository server on AIX, HP-UX, Solaris, or Linux

Starting the repository server on NetWare

RELATED REFERENCES

EMSRV startup parameters

Setting the server disk threshold

EMSRV will write warnings in its log file when there is a minimum amount of free space left on the disk drive where the VisualAge for Java shared repositories reside. The default is to log a warning when there are fewer than 10,000 kilobytes of free space remaining.

To change the disk storage threshold on the server, use the **-b *kilobytes* startup parameter for EMSRV.**

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Starting the repository server on Windows

Starting the repository server on OS/2

Starting the repository server on AIX, HP-UX, Solaris, or Linux

Starting the repository server on NetWare

Displaying EMSRV messages

RELATED REFERENCES

EMSRV Startup Parameters

Setting EMSRV message logging options

You can change where the repository server logs messages, and you can change the level of detail of the messages logged.

Specifying where EMSRV logs messages

By default, messages are logged to a file called `emsrv.log` in the EMSRV working directory. To specify a different log name, use the **-lf** parameter when starting EMSRV.

WIN **OS/2** To log messages to the Command Prompt window from which EMSRV was started, use the **-lc** parameter when starting EMSRV. Messages will also be logged to the log file.

NetWare EMSRV for NetWare logs to the EMSRV message screen as well as to the log file.

AIX **HP-UX** **SOLARIS** **LINUX** To log messages to stdout instead of a log file, use the **-ls** parameter when starting EMSRV. If you do this, you must also use the **-f** parameter to run EMSRV in the foreground.

Specifying EMSRV message logging levels

You can set the message logging level when you start EMSRV, using the appropriate startup parameter:

- **-s0** logs all operations
- **-s1** logs warning and error messages
- **-s2** logs only errors (default for all platforms)

For performance reasons, it is recommended that you use the default reporting level during normal operation. Log more detailed information only when you are trying to diagnose a problem.

You can also change the message logging level while the repository server is running, by issuing the **emadmin opts** command with the **-s** parameter. You will be prompted for the EMSRV user's password.

NetWare Use the EMSRV for NetWare menu to select **Change Settings**. A form will appear, allowing you to view and change the current EMSRV settings, including logging level. Once you have made your changes, press Esc to save them.

Verifying the EMSRV message logging level

To confirm which level of messages are currently being logged, issue the **emadmin opts** command.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Starting the repository server on Windows
Starting the repository server on OS/2
Starting the repository server on AIX, HP-UX, Solaris, or Linux
Starting the repository server on NetWare
Displaying EMSRV messages
Changing the EMSRV working directory

RELATED REFERENCES

The EMADMIN utility - overview
The EMADMIN opts command
EMSRV startup parameters

Changing EMSRV settings (NetWare)

You can change the following settings for EMSRV while it is running:

- Logging level and log file name
- Whether repositories can be truncated
- Whether password validation is in effect
- How often EMSRV should refresh its statistics screen

The available choices for these settings are the same as the EMSRV startup parameters. To change any of these settings:

1. From the EMSRV for NetWare menu on the server, select **Change Settings**. A form will appear.
2. Use the Arrow keys to highlight the setting that you want to change. Make your change and press Enter. The setting will be changed.
3. Press Esc to return to the menu.

Changes made from the menu are only in effect until EMSRV is stopped. The next time you start the repository server, you can specify the same settings (and many others) using startup parameters of the **load emsrv** command.

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Setting EMSRV message logging options
Enabling password validation - overview
Starting the repository server on NetWare

RELATED REFERENCES

EMSRV startup parameters

Adding users to the repository user list

The administrator maintains the list of team members who can access each shared repository.

If you plan to enable password validation with VisualAge for Java, then team members also require either user accounts on the server, or entries in the passwd.dat file.

Adding a user to the repository list

To add a user to the repository list:

1. Change the workspace owner to Administrator.
2. From any **Window** menu, select **Repository Explorer**.
3. From the **Admin** menu, select **Users**. The User Administration dialog box will appear. The left pane lists the users who are currently in the repository list.
4. Click **New**.
5. Enter a unique name, full name, and network login name. Click **Save**. The user list on the left will be updated with the new user.

Changing a user on the repository list

You can change the Full Name or Network Login Name for a user on the list. To change a user on the repository list:

1. Open the User Administration dialog box, as described above.
2. Select the user to be updated.
3. Modify the Full Name or Network Login Name.
4. Click **Save**. The information for that user will be changed.

Deleting a user from the repository list

Before deleting a user, ensure the user does not own any program elements associated with this repository. To delete a user from the repository list:

1. Open the User Administration dialog box, as described above.
2. Select the user to be deleted.
3. Click **Delete**. The user list on the left will no longer show the user.

RELATED CONCEPTS

Repository
Repository administrator
Repository user list

RELATED TASKS

Changing workspace owner
Adding members to a package group
Changing a program element's owner
Enabling password validation - overview

Enabling password validation - overview

When password validation is enabled, team members must provide valid passwords to do these tasks:

- Connect to a repository managed by that server.
- Change workspace owner.

By default, the VisualAge for Java repository server runs with *no* password validation. You have two other options:

- Use native operating system accounts and passwords (available on all server operating systems except OS/2).
- Use a VisualAge for Java password file (passwd.dat) to verify user names and passwords (available on all server operating systems).

Refer to the Related Tasks for details on enabling either password option.

SOLARIS **LINUX** These platforms both support authentication using Pluggable Authentication Modules (PAM). This support enables you to start these server operating systems using the same command (emsrv) whether or not you are using shadow passwords.

PAM must be correctly configured on a machine running EMSRV otherwise it will not even be possible to shutdown EMSRV using EMADMIN.

Refer to your server operating system documentation for more information about PAM.

LINUX In addition, Red Hat Linux, Version 6.2 supports MD5 passwords and EMSRV for Linux also supports these via PAM. You can use the `emsrv` command to start EMSRV for Linux if you are using MD5 passwords.

AIX This platform supports authentication using the `system authenticate()` function. This allows both shadow and non-shadow passwords, in addition to DCE authentication, to be supported with one EMSRV executable. This means that you can use the same command (`emsrv`) to start the server operating system regardless of which method of authentication you are using.

Refer to your server operating system documentation for more information about DCE authentication.

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Enabling native password validation
Enabling password validation with the `passwd.dat` file
Starting the repository server on Windows

Starting the repository server on OS/2
Starting the repository server on AIX, HP-UX, Solaris, or Linux
Starting the repository server on NetWare

RELATED REFERENCES

EMSRV startup parameters

Enabling native password validation

OS/2 Native password validation is not available on OS/2.

You can use native operating system accounts to enforce password validation when team members connect to a shared repository. To enable native password validation:

1. Create an account for each user on the server, following standard procedures for your operating system.
2. Test the new user accounts by logging in to the operating system itself.
3. Add the new users to the VisualAge for Java repository user list. In the User Administration dialog box, provide the operating system account as the **Network Login Name** for each user.
4. **WIN** Include the **-rn** startup parameter when you start EMSRV. You can set your own default EMSRV startup parameters, including **-rn**, by installing EMSRV as a service in the Windows registry.

NetWare Include the **-rn** startup parameter when you start EMSRV. In a Novell Directory Services (NDS) environment, you should also use the **-SC** startup parameter of the **emsrv** command to set an NDS context for the network login names that were provided when the users were added to the VisualAge for Java repository user list. If the **-SC** parameter is not specified, the NDS context will be root.

AIX **SOLARIS** **LINUX** Regardless of the authentication method being used (local password file, shadow passwords, and so on.), start the server by issuing the **emsrv** command with the **-v** startup parameter.

HP-UX If the server operating system uses shadow passwords, start the server by issuing the **emsrv.shadow** command with the **-v** startup parameter. If the server operating system does *not* use shadow passwords, start the server by issuing the **emsrv** command with the **-v** startup parameter.

AIX **HP-UX** **SOLARIS** **LINUX** Root access is required to authenticate users. This can be accomplished by setting the owner of the EMSRV executable to 'root' and setting the SUID bit of the executable. When EMSRV attempts to authenticate a user, it will temporarily change the authority of the EMSRV process to the authority of the owner of the executable. For security reasons, this method is recommended over the more simplistic method of simply running the EMSRV executable as 'root'.

Root access for authentication is required regardless of how EMSRV actually implements authentication. Interfaces such as PAM only provide a common API to permit applications to support multiple authentication methods; configuration specific to each method of authentication must still be correct.

RELATED CONCEPTS

Team client/server configuration
The repository server (EMSRV)
Server security

RELATED TASKS

Enabling password validation - overview
Adding users to the repository user list
Starting the repository server on Windows
Starting the repository server on OS/2
Starting the repository server on AIX, HP-UX, Solaris, or Linux
Starting the repository server on NetWare
Enabling password validation with the passwd.dat file
Installing EMSRV as a service in the Windows registry

RELATED REFERENCES

EMSRV startup parameters

Enabling password validation with the passwd.dat file

As an alternative to using native operating system passwords, the repository administrator can maintain a file of VisualAge for Java users and passwords. This file is called `passwd.dat`. If the appropriate EMSRV startup parameter for password checking is used, this file will be checked whenever a team member connects to a shared repository on the server.

The `passwd.dat` file

The `passwd.dat` file resides in the EMSRV working directory. There is one `passwd.dat` file per server; that file is used for all shared repositories on the same server. The `passwd.dat` file contains one entry per team member, with one user name and password per entry. The user name is first, separated from the password by a single space. Here is an example:

```
fred mypassword  
barney secret  
wilma hello  
betty ZXF65
```

See the instructions below for more information on the user name.

The `passwd.dat` file is *not* encrypted. As a safety precaution, passwords should *not* be the users' network login passwords. You may wish to restrict file access to just the EMSRV user and the team administrator who maintains the file.

The `passwd.dat` file must be in the directory you start EMSRV from.

Enabling VisualAge for Java password checking

To use the `passwd.dat` file for password validation, do the following steps:

1. Create an entry for each user in the `passwd.dat` file on that server.
2. Add each user to the repository user list, providing their name from the `passwd.dat` file as the **Network Login Name** in the User Administration dialog box.

3.    When you start the repository server, use the `-rp` startup parameter of the `emsrv` command.

    When you start the repository server, use the `-r`

startup parameter of the **emsvr** command.

Note: Your `passwd.dat` file must always be in the directory where you start the repository server from.

Users will be prompted for their system password when they connect to a shared repository on the server or when they change workspace owner.

WIN You can set your own default EMSRV startup parameters, including **-rp**, by installing EMSRV as a service in the Windows registry.

RELATED CONCEPTS

Team client/server configuration
The repository server (EMSRV)
Repository administrator
EMSRV user

RELATED TASKS

Adding users to the repository user list
Enabling native password validation
Installing EMSRV as a service in the Windows registry
Changing the EMSRV working directory

RELATED REFERENCES

EMSRV startup parameters

Providing a standard workspace

In the team development environment, you may wish to provide a standard workspace at the beginning of a project, to ensure that team members start with the same editions of the same program elements. You can do this by copying the workspace file, `ide.icx`.

The following procedure is an example of how you can copy the workspace from one client (the source) to another (the target).

1. On the source client, connect to the shared repository.
2. Add the desired projects and packages from the repository to the workspace, and delete projects and packages that are not desired. The result is your standard workspace. See the comments about password validation, below.
3. Exit the IDE. The workspace is saved as the `ide.icx` file on the source client's workstation. The server and repository names are saved in the client's `ide.ini` file.
4. Using file system commands, copy the source client's `ide.icx` and `ide.ini` files. For example, you might call the copies `team1.icx` and `team1.ini`. You may wish to store these on the server, in a directory to which the team has read-only access.
5. On the target client workstation, preserve the existing `ide.icx` and `ide.ini` files by renaming them, for example to `icx.old` and `ini.old`.
6. Copy `team1.icx` and `team1.ini` into the target client's VisualAge for Java program directory, naming them `ide.icx` and `ide.ini`.

7. Start the IDE on the target client. The workspace will connect to the shared repository and will contain the desired projects and package editions. By default, the workspace will be owned by the user who created the standard workspace.
8. Change the workspace owner.

In an environment where password validation is enabled for VisualAge for Java, to start the IDE on the target workstation you must provide the password of the user who owned the standard workspace when it was saved. To handle this situation, the administrator could create a dummy user who owns the standard workspace but has no other privileges, and team members could first start the IDE with the dummy user's password and then change workspace owner.

RELATED CONCEPTS

Team client/server configuration
Workspace
Repository

RELATED TASKS

Saving the workspace
Changing workspace owner
Adding users to the repository user list
Enabling password validation - overview
Connecting to a shared repository
Adding classes and methods from the repository to the workspace
Adding projects and packages from the repository to the workspace
Setting IDE options

RELATED REFERENCES

Important files to back up

Creating a repository

When you first install VisualAge for Java on the team server, one shared repository is provided. By default, this file is called `ivj.dat`. You can create additional repositories, to separate the source code for two completely distinct groups of developers.

Creating a new repository by copying an existing one

You can create a new repository by copying an existing repository. For example, you might copy the original `ivj.dat` repository as a basis for every new repository. Give each copy a different name, such as `team1.dat` and `team2.dat`. The new repository will be exactly the same size as the copied repository, and will have the same repository user list.

Note: If you create a new repository by copying an existing one, ensure that there are no open editions in the `.dat` file you are copying.

For details about performing this task, see the related topic on backing up a shared repository.

Create a new repository by exporting

You can also create a repository by exporting from one repository (the source) to another repository that does not exist yet (the target). The new repository will only

contain the projects and packages that you choose to export, so it will be smaller than the original repository. It will contain all editions of the exported projects and packages. The new repository's user list will include the following users:

- All owners of exported projects, packages, and classes
- Developers of exported classes
- Members of package groups for exported packages

Remember to export the base libraries on which your classes depend. There are four base projects:

- IBM Java Implementation
- Java class libraries
- JFC class libraries
- Sun class libraries

If you forget to include some program elements, you can export them into the same target repository later.

You cannot export any code that is not in the repository.

Creating a new repository by compacting

Finally, you can create a new repository by compacting an existing repository. Compacting copies all program elements from the source repository, but it *only* copies versioned projects, versioned packages, and versioned classes that are contained in versioned packages. Otherwise, this approach is similar to exporting.

RELATED CONCEPTS

Repository
Repository user list
Ownership and team roles - overview

RELATED TASKS

Dividing a repository
Backing up the repository
Backing up a shared repository
Exporting to another repository
Compacting a repository
Adding users to the repository user list
Connecting to a shared repository

RELATED REFERENCES

Repository files
The EMADMIN utility - overview
The EMADMIN copy command

Backing up a shared repository

Overview

The procedures described below are for backing up and restoring a shared repository on a team server. If you wish to back up a local repository on a VisualAge for Java client, see the links to related information at the end of this document.

The shared repository is where all of the team's work is saved. The administrator should back up the repository every day. There two basic alternatives:

- Use the **emadmin copy** command, without stopping the repository server (EMSRV). This command locks the source repository to ensure that no one changes it while it is being copied. **Emadmin copy** creates a new repository on the same server or on another server that is also running EMSRV.
- Use operating system commands or a backup utility to copy the repository file, perhaps to offline media. If you take this approach, you are responsible for ensuring that no one can change the repository while it is being backed up.

It is recommended that you back up resource files, such as images or sound files, at the same time as the source repository. This is easiest to coordinate if team developers use a shared resource directory, as described elsewhere in this documentation.

Backing up with the Emadmin copy command

The **emadmin copy** command locks the repository while it is being copied. Team developers should not be browsing or saving code when the administrator starts the backup.

1. To check the last request time (LRT) of any clients that are still connected, issue the **emadmin list** command.
2. Issue the **emadmin copy** command to copy the repository to another .dat file. The following example copies team1.dat from the EMSRV working directory, to a file called bkup.dat in another directory:

```
emadmin copy team1.dat j:\backups\bkup.dat
```
3. Use operating system commands to back up the shared resource directory.

For more information on **emadmin copy** and its parameters, see the related links that are listed below.

Backing up without the Emadmin copy command

If you prefer to use operating system commands or a server backup utility, then you must ensure that no changes can be written to the repository file while it is being copied. Your operating system or backup utility may provide file locking support. If not, you should disconnect VisualAge for Java clients during the backup, as described in the following example.

1. Ask users to exit the IDE.
2. To ensure that no clients are connected, issue the **emadmin list** command.
3. To prevent clients from reconnecting to the repository while you are backing it up, issue the **emadmin stop** command. This stops EMSRV.
4. Use operating system commands or a backup utility to copy the repository files (.dat files) and any resource files that reside on the server.
5. Issue the **emsrv** command to restart the repository server.

Restoring a repository

Here is an example of a procedure for restoring a repository:

1. To check that no clients are connected to the repository that you are about to replace, issue the **emadmin list** command.
2. To prevent clients from connecting to the repository while it is being restored, stop EMSRV.
3. Rename the repository (.dat file) that you are about to replace. For example, rename team.dat to obsolete.dat.
4. Use operating system commands to copy the backup version of the file into the EMSRV working directory, giving it the original name, for example team.dat.

- Restart the repository server and tell the team they can reconnect to the shared repository.

RELATED CONCEPTS

Repository
Repository administrator
The repository server (EMSRV)
EMSRV user

RELATED TASKS

Displaying server connections
Backing up a local repository
Stopping the repository server
Connecting to a shared repository
Changing repositories
Starting the repository server on Windows
Starting the repository server on OS/2
Starting the repository server on AIX, HP-UX, Solaris, or Linux
Starting the repository server on Windows
Changing the EMSRV working directory

RELATED REFERENCES

Repository files
The EMADMIN utility - overview
The EMADMIN copy command
The EMADMIN stop command
EMSRV startup parameters
Important files to back up

Dividing a repository

As your team grows, you may decide to divide the shared repository into two or more repositories, based on team responsibilities. Here is an example of how you could divide a repository called team1.dat:

- Back up the existing repository, as a precaution.
- Copy the existing repository, team1.dat, to team2.dat.
- Purge program elements from both repositories, based on each team's requirements.
- Tell the members of the second team to change repositories to team2.dat.
- After some period of time - perhaps two weeks - compact both repositories to permanently delete the purged items and reclaim disk space.
After the repository is compacted, only owners are kept, not developers. You have to recreate them, and reassociate them.

If the new repository is on the same server, no user administration is required. If the new repository is on a different server, then the administrator for that server will have to add the developers of team2.dat to the repository user list.

RELATED CONCEPTS

Repository

RELATED TASKS

- Backing up a shared repository
- Purging program elements from the repository
- Changing repositories
- Adding users to the repository user list
- Creating a repository
- Exporting to another repository

RELATED REFERENCES

Repository files

Optimizing server performance

The most important factor in your server's performance is the capacity of the machine itself; fast processors, fast disk drives, and a fast networking connection are obviously better. For optimal performance and availability, the server should be a dedicated machine, not a developer's workstation.

EMSRV is not particularly CPU intensive - performance is I/O-bound. Unless very large resource files are being stored on the server, disk bandwidth is not as important as disk seek times since repository transactions consist mostly of many small read and write operations. All source and compiled code is stored in the repository which is implemented as one large file. It is important that this file does not become too fragmented otherwise performance may degrade. The simplest way to avoid fragmentation is to ensure that there is plenty of free disk space available on the volume that is hosting the repository. Any optional volume or file compression or encryption options should be disabled as they usually have a negative impact on performance.

EMSRV and VisualAge for Java clients communicate using TCP/IP so it is important that the TCP/IP stacks on the client and the server are optimized for the network environment they are operating in.

Optimizing performance of shared repositories

The size of the repository does not have a large impact on performance unless it is severely fragmented. To reduce the size of the shared repository and to reduce the number of file and directory entries being used in the stored resource directory, the administrator should periodically compact the repository. This will create a new repository and a corresponding new resource directory. Compacting the repository is only useful if package and project owners first purge the editions of their program elements that are no longer required.

The size of the repository does not have a large impact on performance unless it is severely fragmented. To reduce the size of the shared repository and to reduce the number of file and directory entries being used in the stored resource directory, the administrator should periodically compact the repository. This will create a new repository and a corresponding new stored resource directory.

After the repository is compacted, only owners are kept, not developers. You have to recreate them, and reassociate them.

Setting message logging levels for better performance

You can change the level of detail of the messages logged by EMSRV. For performance reasons, it is recommended that you use the default reporting level during normal operation. Log more detailed information only when you are trying to diagnose a problem.

RELATED CONCEPTS

Server considerations in team development
 Number and placement of shared repositories
 Server files and directories

RELATED TASKS

Creating a repository
 Dividing a repository
 Purging program elements from the repository
 Compacting a repository
 Setting EMSRV message logging options
 Changing the EMSRV working directory

RELATED REFERENCES

Repository files

EMSRV startup parameters

The following table lists and describes parameters you can use when starting the VisualAge for Java repository server. All parameters are case-sensitive.

Parameter	Server operating system	Description
-A <i>0, 1</i>	All	The file system requires read locks. The default setting is 0 which indicates the file system does not require read locks.
-a <i>seconds</i>	AIX, HP-UX, Solaris, Linux	Sets the number of seconds before a connection is deemed inactive. The default is 360 seconds.
-b <i>Kilobytes</i>	All	Sets the low-volume threshold warning in kilobytes. The default is 10,000 kilobytes. If the available disk space is less than the low-volume threshold, EMSRV will log warning messages to the log file.
-f	AIX, HP-UX, Solaris, Linux	Sets EMSRV to run in the foreground.
-h	All	Displays the help text that lists the valid parameters.
-i <i>q,t</i>	AIX, HP-UX, Solaris, Linux	Ignores signals. <i>q</i> = ignore SIGQUIT; <i>t</i> = ignore SIGTERM.
-install	WinNT, Windows 2000	Installs EMSRV as a service in the Windows registry.
-lc	WinNT, Windows 2000, Netware, OS/2	Logs messages to the server console. By default, messages are not written to the console. (See also -lf.)
-lf <i>name</i>	WinNT, Windows 2000, Netware, OS/2	Writes the log to file <i>name</i> . By default, the log is written to emsrv.log. The file <i>name</i> must specify a valid path for which the EMSRV user has sufficient rights. (See also -lc.)
-lp <i>seconds</i>	AIX, HP-UX, Solaris, Linux	Sets the maximum number of seconds to wait for a lock. The default is 15 seconds.
-ls	AIX, HP-UX, Solaris, Linux	Logs messages to stdout instead of a log file. You must also use -f parameter if -ls is used.
-lt <i>seconds</i>	AIX, HP-UX, Solaris, Linux	Sets the maximum number of seconds to hold a lock.

-M <i>numberOfConnections</i>	All	Specifies the maximum number of connections that can be established to EMSRV. The default is 256, but some TCP/IP stacks will run out of stream sockets before this limit is reached. To start EMSRV with a maximum of 80 client connections, use the following parameter: EMSRV -M80
-n	AIX, HP-UX, Solaris, Linux	Turns off statistics gathering.
-P <i>portNumber</i>	All	Specifies the port number that EMSRV uses. The default is 4800. To start EMSRV using port number 4899, use the following parameter: EMSRV -P4899
-p <i>password</i>	WinNT, Windows 2000 NetWare, OS/2	Provides the password for the EMSRV user. The same password is required for certain EMADMIN functions such as shutting down the repository server remotely. (See also: -u.) For NetWare and Windows you must specify -p without the <i>password</i> argument, if the EMSRV user has no password.
-R <i>0,1</i>	All	The file system releases locks on file close. The default setting is 1 which indicates the file system releases locks when a file is closed. 0 indicates the file system does not release locks when a file is closed. If the latter is true, then locks are manually released before closing a library.
-r	AIX, HP-UX, Solaris, Linux	Rejects users who are not in the passwd.dat file.
-rd	WinNT, Windows 2000, OS/2	Disables password checking for clients. This is the default setting.
-remove	WinNT, Windows 2000	Removes the EMSRV service from the registry.
-rn	WinNT, Windows 2000, NetWare	Rejects users who do not supply a user name and password recognized by the server operating system. By default, no password is required to connect to the shared repository.
-rp	WinNT, Windows 2000, NetWare, OS/2	Rejects users who are not in the passwd.dat file. By default, no password is required to connect to the shared repository.
-s <i>0, 1, 2</i>	All	Sets the reporting level to the specified severity level: <ul style="list-style-type: none"> • -s0 logs all operations • -s1 logs warning and error messages • -s2 logs only errors (default for all platforms)
-SC <i>context</i>	NetWare	Sets the default EMSRV for NetWare, NetWare Loadable Module (NLM), Novell Directory Services (NDS) context. See also -rn.
-t	All	Protects existing files from truncation. By default, files are created over existing ones; that is, the existing file is truncated to 0 length.
-u <i>username</i>	WinNT, Windows 2000, NetWare	Specifies the EMSRV user. (See also: -p.)
-v	AIX, HP-UX, Solaris, Linux	Verifies password using system authentication..

<code>-W path</code>	WinNT, Windows 2000, NetWare, OS/2	Specifies the EMSRV working directory. The <i>path</i> must be a valid path for which the EMSRV user has sufficient rights to read and write.
<code>-w</code>	All	Specifies that EMSRV should track locks for each connection.
<code>-xn</code>	AIX, HP-UX, Solaris, Linux	Allows databases to be opened on non-native devices.
<code>-xddevice number</code>	AIX, HP-UX, Solaris, Linux	Specifies valid devices for the database. Do not put a space between <code>-xd</code> and the device number.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Setting up a team server - overview
Installing EMSRV as a service in the Windows registry
Starting the repository server on Windows
Starting the repository server on OS/2
Starting the repository server on AIX, HP-UX, Solaris, or Linux
Starting the repository server on NetWare
Changing the EMSRV working directory

RELATED REFERENCES

The EMADMIN utility - overview

EMSRV and TCP/IP

TCP/IP is the only supported network protocol for the VisualAge for Java team development environment.

By default, EMSRV uses port 4800. To change this, use the **-P** parameter when issuing the **emsrv** command to start the repository server. For example:

```
emsrv - P4803
```

The running EMSRV process will listen for connections on port 4803 instead of default port 4800.

Clients must be configured to use the port that EMSRV will use.

The default limit for client connections to a server is 256. This limit can be changed by using the **-M** parameter of the **emsrv** command. Some TCP/IP stacks will run out of stream sockets before this limit is reached.

VisualAge for Java client/server connections use the TCP keep-alive timer. Some TCP/IP stacks allow the keep-alive timer to be changed. Typically it is 120 minutes. After this time has elapsed, the EMSRV TCP stack sends a keep-alive packet. If the client does not reply, the connection on the server and all its resources are cleaned up.

RELATED CONCEPTS

Team client/server configuration
The repository server (EMSRV)

RELATED REFERENCES

EMSRV startup parameters

The EMADMIN utility - overview

The EMADMIN command-line utility allows you to manage a repository server (EMSRV) from any network-attached workstation where emadmin.exe is installed.

The following table summarizes the EMADMIN commands that are available. Each of these commands is discussed as a separate topic in this reference. To display a list of these commands and their most common parameters, issue the **emadmin** command with no parameters.

Command	Description
bench	Runs bench tests between the client TCP stack and the server TCP stack. The tests determine any performance problems.
copy	Copies a VisualAge for Java repository on the server.
list	Displays the current EMSRV connection list or information about a specific connection.
opts	Displays the current EMSRV settings.
stat	Displays EMSRV statistics.
stop	Shuts down EMSRV remotely or closes an active connection.

Supported Operating Systems

EMADMIN is supported on the following operating systems:

- Windows 95 and 98
- Windows 2000 Professional
- Windows 2000 Server
- Windows 2000 Advanced Server
- Windows NT Workstation 4.0 (with Service Pack 5)
- Windows NT Server 4.0 (with Service Pack 5)
- OS/2 Warp 4.0
- OS/2 Warp Server for e-business
- AIX 4.3.2, 4.33
- Sun Solaris 2.6, 7.0
- HP-UX 10.20, 11.0
- Red Hat Linux 6.1, 6.2

Syntax

EMADMIN uses the following syntax:

```
emadmin command [parameter1] [parameter2]...
```

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED REFERENCES

The EMADMIN copy command
The EMADMIN list command
The EMADMIN opts command
The EMADMIN stat command
The EMADMIN stop command

The EMADMIN bench command

The **emadmin bench** command is used to do problem determination, when there are suspected performance problems between the TCP stacks on the clients and server. It is normally run at the request of IBM's technical support organization, who would normally ask you to submit the log files to them for analysis.

The **emadmin bench** command runs a series of read tests on a temporary source repository to determine the performance for different client buffer sizes. The test takes approximately 1 to 5 minutes. You should delete the temporary source repository after the test has been completed.

Syntax

```
emadmin bench [-t test repository] [-l logfile] [-s size] [-P portNumber]
```

Parameter	Description
-t	Specifies the name of the temporary repository file to create. The default name is testlib.dat.
-l	Specifies the name of the log file to write. The default name is embench.log. If the log file exists, it is overwritten.
-s	Specifies the size of the benchmark sweep.
-P	Specifies the port that EMSRV is using. The default port number is 4800.

Example

```
emadmin bench myserver.com.us -t test1.dat
```

This example creates the test repository test1.dat and runs the bench test, logging to the default log file embench.log.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Creating a repository

RELATED REFERENCES

The EMADMIN utility - overview

The EMADMIN copy command

The **emadmin copy** command allows you to copy a VisualAge for Java repository (the source) to another repository (the target). The target repository may be on the same server, or on another server in the network. The **emadmin copy** command locks the source repository to make sure it is not changed during the copy operation.

Restrictions

- EMSRV must be running on the source and target servers.
- You must know the EMSRV user's password.
- The source and target repositories must reside in directories to which the EMSRV user has access. They can not reside on remote file systems.
- While the repository is being copied, team developers should not be browsing, saving code, or running scripts that lock the library. The administrator can use the **emadmin list** command to see whether clients are still connected, or to see the the last request time (LRT) of clients that are still connected.

Syntax

```
emadmin copy source target -p password [-o] [-q] [-P portNumber]
```

Parameter	Description
<i>source</i>	The VisualAge for Java repository file that you want to copy. The file specification has the following format: <i>[ip_address]:[path]filename</i> <i>ip_address</i> is optional; it is the host name or IP address of the server where EMSRV is running.
<i>target</i>	The name of the repository to which you are copying. The format is the same as for the source file. If the repository does not exist, it will be created. If it does exist, you will be prompted to confirm that you want to overwrite it. The file specification has the same format as the source repository.
<i>-p password</i>	The password of the EMSRV user (the user who started EMSRV) on the source server.
<i>-o</i>	Specifies that the target file may be overwritten without prompting.
<i>-q</i>	Specifies quiet operation so that EMADMIN will not issue any prompts or write the number of bytes transferred to the console.
<i>-P</i>	Specifies the port that EMSRV is using. The default port number is 4800.

Example

To copy a repository file to a backup file, issue the following command:

```
emadmin copy team1.dat bkup.dat -o -p emsrvpw
```

In this example, the team1.dat repository will be copied to the bkup.dat repository, overwriting bkup.dat if it exists. Both repositories are in the EMSRV working directory on the same server, so no server or directory information needs to be provided. The EMSRV user's password is emsrvpw.

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Backing up a shared repository
Creating a repository
Changing the EMSRV working directory

RELATED REFERENCES

The EMADMIN utility - overview

The EMADMIN list command

The **emadmin list** command shows current connections to the repository server. You can use it to obtain information about connection statistics and active locks.

Syntax `emadmin list host [-s connectionNumber] [-l] [-P portNumber]`

Parameter	Description
<i>host</i>	The host name or IP address of the server whose active connections you want to display. By default, this is the name of the host from which you are issuing the emadmin command.
-s	Display the statistics for the connection specified by connection number.
-l	Display the active locks for the connection specified by connection number.
-P	Specifies the port that EMSRV is using. The default port number is 4800.

Example 1

Here is an example of the information that might be displayed in response to issuing `emadmin list teamserv` from a command prompt:

```
EMSRV Connection list for: teamserv
  Active Last
  ID IP Address  Locks Request Repository
-----
  0 9.55.55.155  0 18:18:27 ivj.dat
  1 9.55.55.156  0 18:38:14 another.dat
-----
```

Example 2

Here is an example of the information that might be displayed, in response to issuing `emadmin list 9.12.13.14 -s 0` from a command prompt:

```
EMADMIN 7.0
Copyright (C) IBM Corporation 1989-2000
Server Type : EMSRV
Server Version : EMSRV 7.0 for Windows NT Apr 4 2000 15:36:56 (AEST)
-----
Connection 0 statistics (9.21.13.14)
Repository: ivj.dat
Total Opens:    2 Total Closes:    1
Total Reads:   6 Total Writes:    0
Total Locks:   0 Total Unlocks:    0
Current Locks: 0
-----
```

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Displaying server connections

RELATED REFERENCES

The EMADMIN utility - overview

The EMADMIN opts command

The **emadmin opts** command allows you to do the following tasks:

- Display the parameters that were used to start the repository server
- Change the logging level for the EMSRV log, without restarting EMSRV

Syntax

`emadmin opts host [-s level] [-P portNumber]`

Parameter	Description
<i>host</i>	The host name or IP address of the server whose active connections you want to display. By default, this is the name of the host from which you are issuing the emadmin command.
<i>-s level</i>	Specifies a new logging level for EMSRV. You can specify the following logging levels: 0 Logs all operations and messages to the log file. 1 Logs warnings and errors to the log file. 2 Logs only errors to the log file (default).
<i>-P</i>	Specifies the port that EMSRV is using. The default port number is 4800.

Example

Here is an example of the information that might be displayed in response to issuing `emadmin opts javateam` from a command prompt:

```
EMADMIN 7.0
Copyright (C) IBM Corporation 1989-2000
Server Type : EMSRV
Server Version : EMSRV 7.0 for Windows NT Apr 4 2000 15:36:56 (AEST)
=====
EMSRV Options for: javateam
-----
EMSRV 7.0 for Windows NT Apr 4 2000 15:36:56 (AEST) Options
Maximum number of concurrent connections = [256]
Working directory = [g:\emsrv623]
Password checking = [Disabled]
Logging level = [Error]
Log file name = [emsrv.log]
Allow connection to truncate libraries = [true]
Track EMSRV statistics = [true]
Track process file locking statistics = [false]
Process activity timeout value = [360] sec.
Sleep on lock value = [1000] msec.
Free disk space warning threshold = [10000] KBytes
Restrict libraries to local filesystems = [false]
-----
```

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Changing EMSRV settings (NetWare)

RELATED REFERENCES

The EMADMIN utility - overview
EMSRV startup parameters

The EMADMIN stat command

The **emadmin stat** command provides statistics for operations completed since the repository server was started. It also tells you what the current EMSRV working directory is.

Syntax

emadmin stat *host* [-P *portNumber*]

Parameter	Description
<i>host</i>	The host name or IP address of the server whose active connections you want to display. By default, this is the name of the host from which you are issuing the emadmin command.
-P	Specifies the port that EMSRV is using. The default port number is 4800.

Example

Here is an example of the information that might be displayed in response to issuing `emadmin stat teamserv` from a command prompt:

```
EMADMIN 7.0
Copyright (C) IBM Corporation 1989-2000
Server Type : EMSRV
Server Version : EMSRV 7.0 for Windows NT Apr 4 2000 15:36:56 (AEST)
=====
EMSRV Statistics for: teamserv
-----
EMSRV 7.0 for Windows NT Apr 4 2000 15:36:56 (AEST)
Total Connects: 27 Total Disconnects: 24
Total Opens: 9 Total Closes: 7
Active Locks 0 Unexpected Connection Closes: 0
Total Locks: 0 Total Unlocks: 0
Total Reads: 44 Total Writes: 0
Total Reads Failed On Lock: 0 Total Locks Failed On Lock: 0
Times Lock Limit Hit: 0
Total Requests Serviced: 282 Requests in last interval: 0
Largest Packet Sent: 6836 Largest Packet Received: 24
Server Has Been Alive For: 0 Days 19 Hours 53 Minutes 31 Seconds
Server Working Directory : g:\teamreps
-----
```

RELATED CONCEPTS

The repository server (EMSRV)

RELATED TASKS

Stopping the repository server
Stopping a client connection
Changing the EMSRV working directory

RELATED REFERENCES

The EMADMIN utility - overview

The EMADMIN stop command

The **emadmin stop** command allows you to do the following tasks:

- Stop a client's connection to the repository server
- Stop EMSRV

Syntax

```
emadmin stop host [-k connection_number] [-p password] [-P portNumber]
```

Parameter	Description
<i>host</i>	The host name or IP address of the server. By default, this is the name of the host from which you are issuing the emadmin command. If you do not use -k to specify a particular connection, the entire server will be shut down.
-k	The unique connection number that you want to terminate. (Use the emadmin list command for a list of connections.)
-p	The password of the EMSRV user (the account used to start EMSRV). If you do not provide the password, you will be prompted for it.
-P	Specifies the port that EMSRV is using. The default port number is 4800.

Example

To stop the repository server on an IP host called teamserv, which was started with the password "secret", issue the following command:

```
emadmin stop teamserv -p secret
```

RELATED CONCEPTS

The repository server (EMSRV)
EMSRV user

RELATED TASKS

Stopping the repository server
Stopping a client connection

RELATED REFERENCES

The EMADMIN utility - overview

Notices

Note to U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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.*

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan*

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 OR CONDITIONS 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.

Licenses of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*Lab Director
IBM Canada Ltd.
1150 Eglinton Avenue East
Toronto, Ontario M3C 1H7
Canada*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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 may 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.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1997, 2000. All rights reserved.

Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow the customer to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

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

- AIX
- AS/400
- DB2
- CICS
- CICS/ESA
- IBM
- IMS
- Language Environment
- MQSeries
- Network Station
- OS/2
- OS/390
- OS/400
- RS/6000
- S/390
- VisualAge
- VTAM
- WebSphere

Lotus, Lotus Notes and Domino are trademarks or registered trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Tivoli Enterprise Console and Tivoli Module Designer are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

Encina and DCE Encina Lightweight Client are trademarks of Transarc Corporation in the United States, or 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 and other countries.

ActiveX, Microsoft, SourceSafe, Visual C++, Visual SourceSafe, Windows, Windows NT, Win32, Win32s and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

Other company, product, and service names, which may be denoted by a double asterisk(**), may be trademarks or service marks of others.