This presentation introduces IBM Rational ClearCase.

Course objectives

- These topics are covered in this module:
  - IBM Rational ClearCase's implementation of software configuration management (SCM)
  - ClearCase roles
  - ClearCase terminology
  - Developer's workflow

This module begins with an introduction of software configuration Management, or SCM, then covers the various roles that exist in a ClearCase environment and how they interact, followed by a review of basic ClearCase objects and terms. The last topic covers the typical workflow of developing software in ClearCase.

ClearCase implementation of SCM

- Version control
  - Directories and files
- Workspace management
  - Isolation and parallel development
- Project history
  - Tracks all project changes
- Developer workflow
  - Accurately reproduces builds and releases

A software configuration management system, or SCM, is essential for controlling the numerous directories and files produced by the many people working on a project. This control helps avoid confusion among team members, controls cost, and ensures that project artifacts (both files and directories) are not in conflict due to:

Simultaneous Update. For instance, if two or more team members work separately on the same file but attempt to update at the same time, the last one to make changes overwrites the work of the previous updater.

Limited Notification. For instance, if a given problem is fixed in one or more files shared by several developers, some of the developers are not notified of the change or changes.

Multiple Versions. Most large programs are developed in evolutionary releases. For instance, many different versions of the same file could exist -- one release could be in customer use, another in test, and a third in development. If problems are found in any one of the versions, fixes need to be propagated to all versions. If changes are not carefully controlled and monitored, the result can be confusion, costly fixes, and unnecessary re-work.

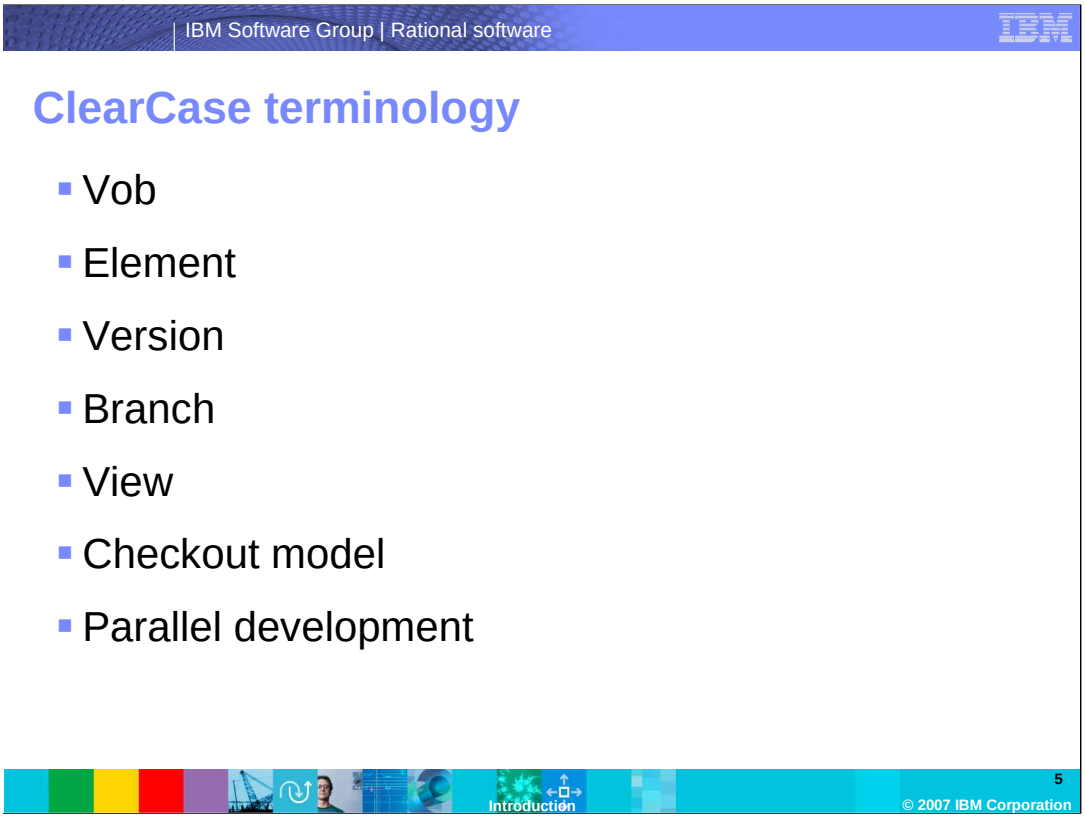
ClearCase roles

- **ClearCase developer**
  - Set up workspace
  - Make changes
  - Integrate changes
  - Update workspace
- **ClearCase configuration manager**
  - Establish CM policies
  - Write CM plan
  - Design and set up CM environment
  - Assign and schedule work
  - Monitor project status

- **ClearCase integrator**
  - Create integration workspace
  - Create baselines
  - Build components
- **ClearCase administrator**
  - Set up hardware environment
  - Implement development environment
  - Maintain hardware and development environments

There are a number of tasks that must be performed in a ClearCase environment. ClearCase roles group these tasks, as shown here, and do not correspond to individuals. In some organizations, one person may fill more than one role.

The ClearCase Developer role can set up workspaces, make changes, integrate those changes, and update workspaces. The ClearCase integrator role can create an integration workspace, create baselines, and build components. The ClearCase Configuration Manager role (or, CM), can establish CM policies, write the CM plan, design and set up the CM environment, assign and schedule work, and monitor project status. The last role, ClearCase Administrator role can set up the hardware environment, implement the development environment, and maintain the hardware and development environments.



For the first-time user of IBM Rational ClearCase, the terminology can sometimes be a bit overwhelming, but do not be intimidated by it. For every tool there is a learning curve, some steeper than others. The first step is to start with terminology and jargon used with ClearCase. This section will cover basic ClearCase concepts including: vobs, elements, versions, branches, views, and various checkout models.

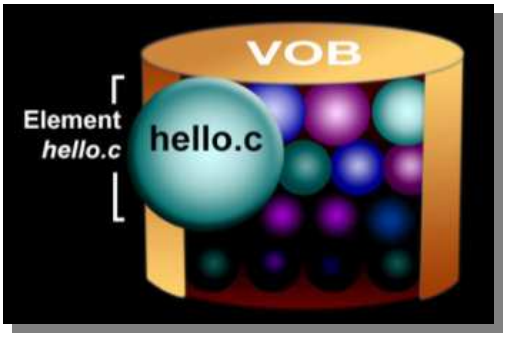
VOB

- A permanent, read-only data repository that:
  - Stores files, directories, and metadata
  - Stores version-controlled data
  - Displays its contents as files in a file system
  - Stores anything that can be represented as a file or directory
  - Can be replicated in two or more sites

A ***VOB*** or Versioned Object Base is a permanent, read-only repository that stores: versions of file elements, directory elements, derived objects (or, executables), version-controlled data, meta data associated with these objects, and virtually anything that can be represented as a file or directory.

Element

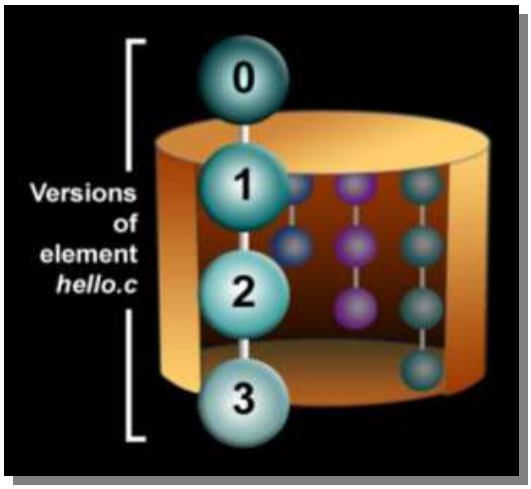
- A file or a directory, under source control, stored in a ClearCase VOB
- Can be any object that can be stored in a native file system, including:
  - Source files
  - Directories
  - Binary files
  - Object libraries
  - Documents



Elements can be either files or directories, under source control that are stored in a ClearCase VOB. A file element can be any object that can be stored in a native file system, including – source files, directories, binary files, object libraries, and documents.

Version

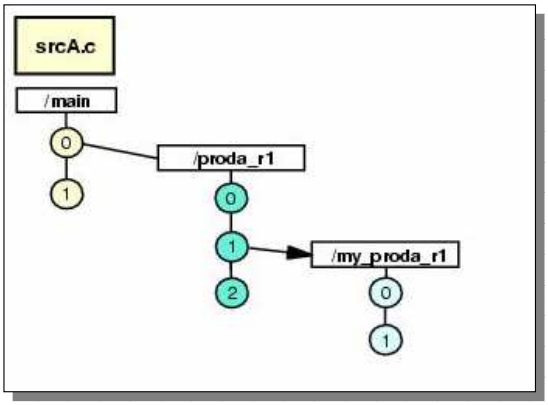
- An element consists of a set of versions, organized into a version tree
- Each version represents one revision of a file under source control
- Versions are displayed in a workspace or view



A ***version*** denotes a particular revision of an element. An element consists of a set of versions, organized into a version tree. Each time you revise and check in a file or directory, ClearCase creates a new version of it. The versions of an element are organized into a version tree structure as shown above – with numbered versions 0 through 3 of the element called “hello.c”.

Branch

- A ***branch*** is an object that specifies a sequence of versions of an element.
- Each element has a main branch
- Branching is done in order to provide work areas for developers

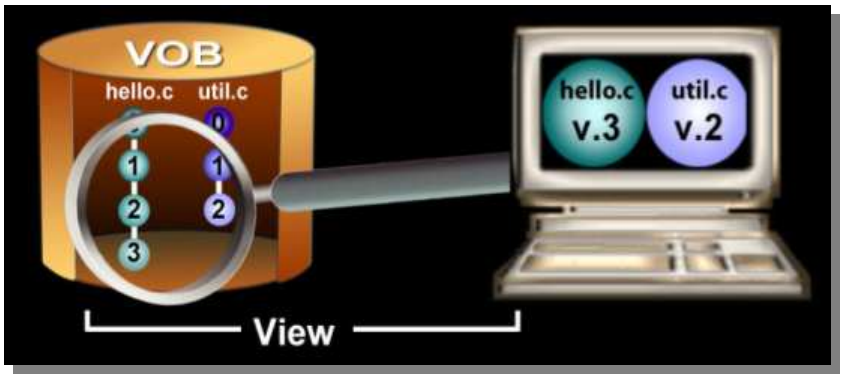


The concept of ***branches*** and branching is an important topic in ClearCase. A branch is an object that specifies a sequence of versions of an element. Branching is done in order to provide work areas for development. It is not uncommon to have dozens of branches in an elements version tree. Note that the ClearCase naming convention for branches is to use lower case alphabetic names. Because branching is done on a very low level, it also becomes very important to find a branching strategy that works from all level of perspectives and abstractions.

Some general rules that apply for branching to help with maintenance. First, the main branch should not be used for development work, but should be reserved for major milestones or released versions. Second, all major development is done on branches.

View

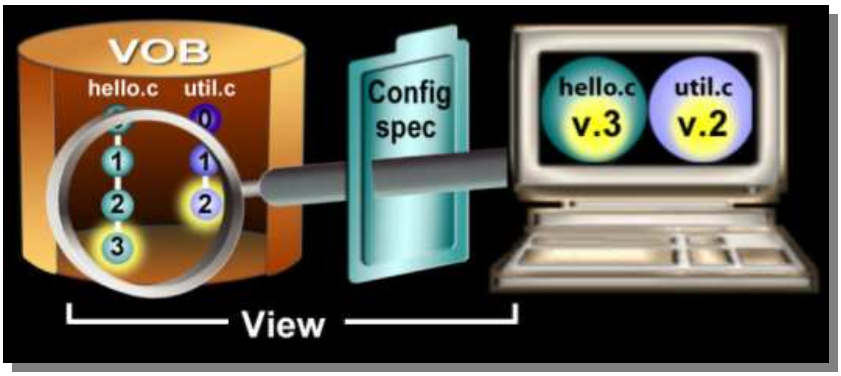
- A ClearCase mechanism that allows users access to versions of elements in VOBs
- An isolated workspace for a user or a group
- Enables users to work in parallel



A ***view*** is a ClearCase object that provides a work area for one or more users to edit and create versions, compile them into object modules, format them into documents, and so on. Users in different views can work on the same files without interfering with each other. To access elements in a VOB you have to use a view.

View and configuration specification

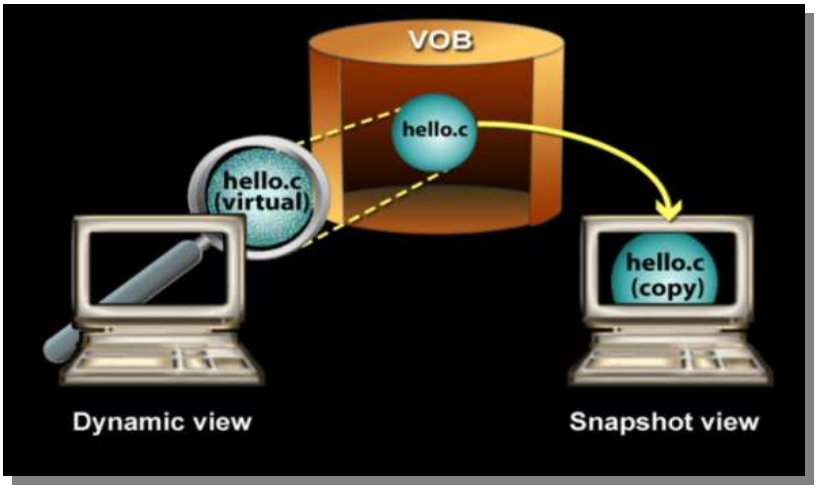
- For each view, a configuration specification is a set of ordered rules used to select [at most] one version of each element
- A configuration specification determines which versions of an element are visible in the view



A set of rules called a ***configuration specification***, determines which files are visible in a view. A view's configuration specification selects one version from the element's version tree and displays it in the view. You can determine what elements are visible to the view by adjusting the configuration specification in ClearCase. In this example, only the versions highlighted above are visible in the view.

View

- View types
  - Dynamic view
  - Snapshot view



There are basically two types of ClearCase views - dynamic and snapshot.

A ***dynamic*** view is a type of view that is always current with the VOB. Dynamic views use the ClearCase Multi version file system (or, MVFS) to create and maintain a directory tree that contains versions of VOB elements.

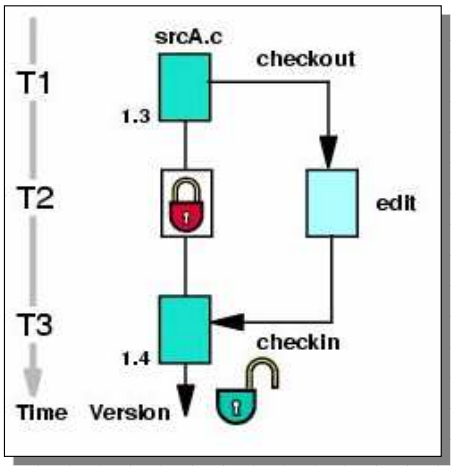
A ***snapshot*** view is a view that contains copies of ClearCase elements and other file system objects in a directory. You use an update tool to keep the view current with the VOB.

Use dynamic views when you want to access elements in repositories without copying them to your computer, or when you want the view to reflect changes made by other team members at all times.

Use snapshot views when you are using Windows® 95, Windows 98, Windows Millennium Edition, or ClearCase LT, which do not support dynamic views, or when you want to work disconnected from the network.

Check-out and check-in

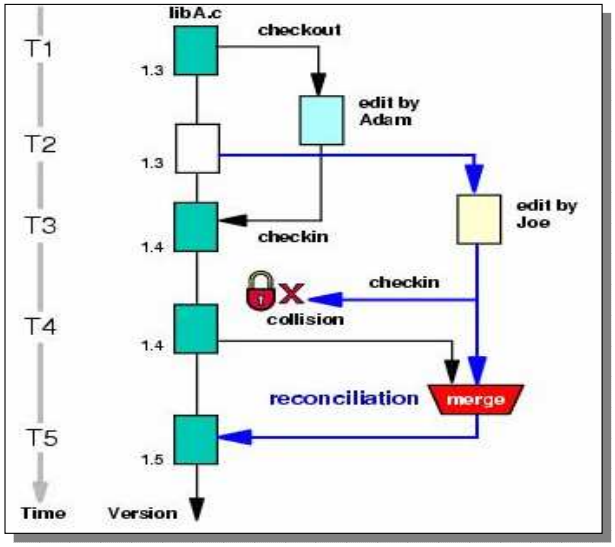
- A ***checkout*** enables you to have a private and editable copy of a specific element in your view
- When a file is checked in it is placed in the VOB
- If the checkout is cancelled, there is no change to the VOB



Check-out and check-in is a two part process that extends a branch of an element's version tree with a new version. The first part of the process, check-out, expresses your intent to create a new version at the current end of a particular branch. A ***checkout*** enables you to have a private and editable copy of a specific element in your view. Note that if the checkout is cancelled, there is no change to the VOB. The second part, check-in, completes the process by creating a new version in the VOB.

Reserved check-out

- A ***reserved*** check-out locks an element
- Guarantees a user's right to check in a new version
- Others must perform a merge if they want to save their checked out versions

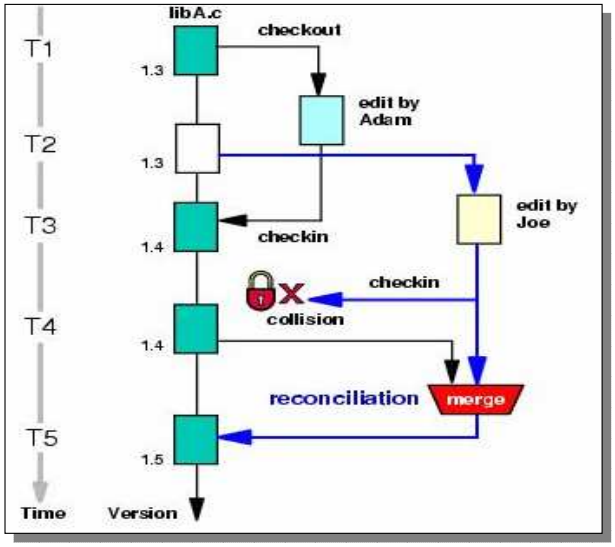


In ClearCase, users can use a check-out model where the checked out element is locked during the checkout. This is known as a ***reserved*** checkout model. The reserved check-out model guarantees a user's right to check in a new version of an element. If there are multiple users editing the same version of an element, they may come into conflict, or, collision with other checked-out versions. In this environment, performing a check-out does not guarantee you the right to perform a subsequent check-in as many users can check out the same version of an element, as long as they are working in different views. At most, one of these can be a reserved check-out. An unreserved check-out affords no such guarantee. If users have unreserved check-outs of the same version in different views, the first user to perform a check-in wins. Other users must perform a merge if they want to save their checkout-out versions.

This kind of model works fine in serial development, where the likelihood of more than one person checking out a specific element at the same time is very small. However, when doing concurrent or parallel development, this kind of checkout model does not work, and something else is needed. In concurrent development, as supported by ClearCase, you need a checkout model that allows for more than one person to check out a specific element at the same time.

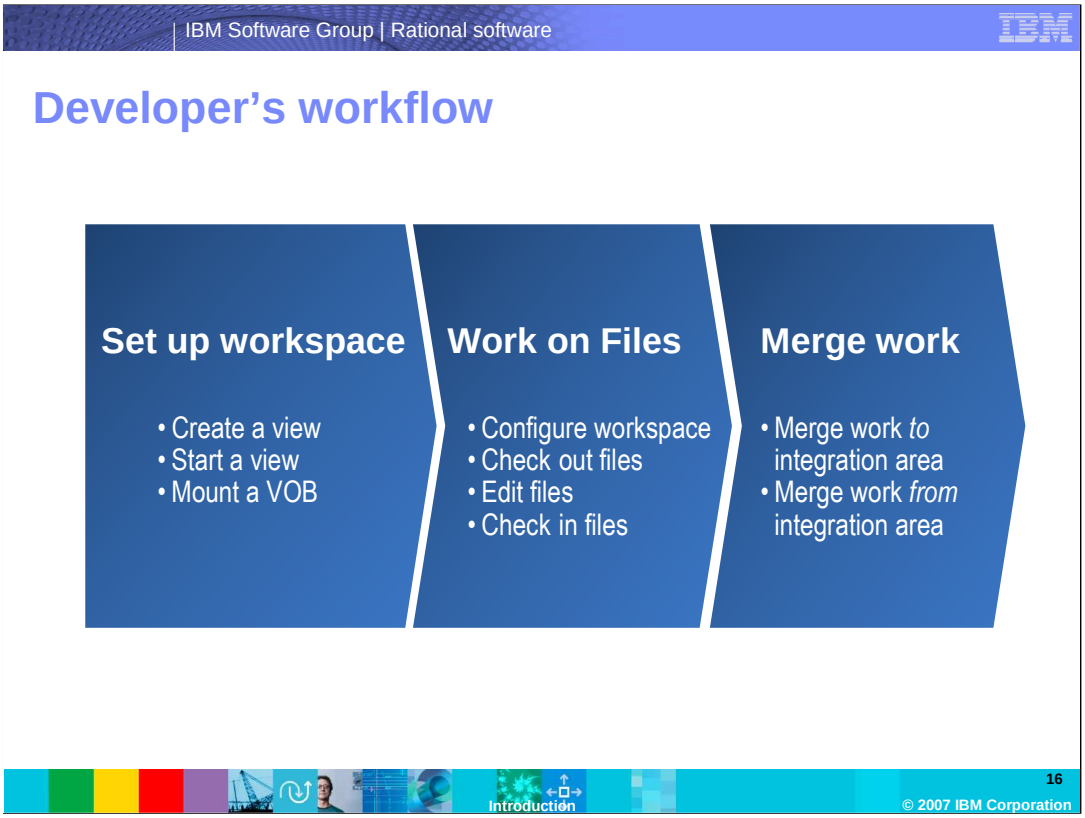
Parallel development

- Concurrent changes made to individual files
- Capability to merge the changes



Parallel development is defined as concurrent changes made to individual files by two or more individuals. Parallel development also includes the capability to merge the changes that have been made in parallel. The combination of configuration specification rules, views, VOBs, and branches provide the basis for parallel development in ClearCase.

In the example shown here, when Adam's checkin is complete (version 1.4), Joe can proceed with his checkin attempt. The conflict occurs here because the same checked out version of the element has already been checked in by Adam, the tool will detect a *collision* situation and will force Joe to perform a *reconciliation* (of his edited version 1.3 and Adam's version 1.4). If the checked out element is a textual file, the reconciliation can probably be performed with the help of a merge tool.



Now that you have the ClearCase basics, roles, and terminology down, this section will describe the developer's workflow in ClearCase. Using the flow above, starting with setting up the workspace, then working on files, and finishing up with merging your work, you can repeat the workflow to ensure success when getting started with ClearCase.

Setting up a workspace

The first step is to set up a workspace in ClearCase. You can create a view, start and view, and mount a VOB. You create a view that has a configuration specification that will select versions of elements that the developer needs to use.

Work on files

Once a view is created and a VOB is mounted, the developers can start working on the files. This step includes: configuring the workspace, check-out files, edit those files, and then checking them back in. These files in the VOB would be modified by checkout/checkins. New files would be created, placed under source control, and then modified by checkout/checkins.

Merge Work

The final step is to make your work available to other developers. This is accomplished by merging the latest version on your branch with the latest version on the main or integration branch. If you need other developer's work on your branch, you would merge their version from the main branch to your branch. This flow from your branch to the main and information from the main branch to yours allows many developers to work in parallel.

Module summary

- In this module, you have learned about:
  - ClearCase and SCM
  - ClearCase roles
  - ClearCase terminology - Views and vobs
  - Developer workflow

Contributors: Paul Doucette and Marcus Matic

In summary, this module covered a variety of topics important for you to know when using IBM Rational ClearCase, including: Software Configuration Management basics, ClearCase roles and tasks, ClearCase terminology including Views, VOBS, and branches, and an introduction to developing in ClearCase. If you would like to find out more information on IBM Rational ClearCase or other Software Configuration Management solutions, visit www.ibm.com/software/rational.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_RCCv7_Module1_IntroCC.ppt

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

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

ClearCase IBM Rational

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation in the United States, Other Countries, or both.

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

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. 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 in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.