WebSphere® eXtreme Scale provides the ability to secure access to protect data in the cache. You can enable security within the eXtreme Scale Grid, and integrate with external security providers. This presentation provides a brief summary of eXtreme Scale security features. You can get more detailed information on configuring security from the Administration Guide and the Programming Guide.

Agenda

- Security concepts
- Configuring grid security
- System extension points

This presentation will first provide an overview of basic security concepts. The majority of the presentation will cover the concepts and artifacts required to configure grid security. Finally, the presentation will introduce several system programming interfaces which allow you to extend and customize the grid's security operations.

Security concepts

- Trustable authentication
  - The ability to determine the identity of the requester
- Authorization
  - The ability to give permissions or grant access rights to the requester
- Secure transport
  - The safe transmission of data over a network

Any secure product must be able to reliably determine the identity of any principal. A principal can be an actual user, a client program, or another server in a distributed grid. Once a principal's identity has been verified, the product must provide some mechanism to determine what operations and data the principal is allowed to access. Finally, all communications betweens grid components must be secure to prevent unauthorized access to the data as it flows across the network.

Authentication

- Ensure a client is authentic
- WebSphere eXtreme Scale supports a distributed client server framework
  - A client server security infrastructure is in place to secure access to eXtreme Scale servers
- When authentication is required
  - Client must provide credentials to authenticate to the server
  - Client credential can be
    - User name and password pair
    - Client certificate
    - Kerberos ticket
    - Data that is presented in a format that is agreed upon by client and server

The primary goal of authentication is to ensure a client is who it claims to be. Numerous mechanisms exist to accomplish this. WebSphere eXtreme Scale provides easy integration with external security implementations. This external implementation must provide authentication and authorization services for the eXtreme Scale. The eXtreme Scale has plug-in points to integrate with a security implementation. WebSphere eXtreme Scale has been successfully integrated with Lightweight Directory Access Protocol (LDAP), Kerberos, WebSphere security, Tivoli® Access Manager, and Java™ Authentication and Authorization Service (JAAS). In addition, eXtreme Scale provides a flexible framework that allows you to provide your own authentication implementation.

Authorization

- Ensure client has permission to perform requested action
- Grid authorizations are based on subjects and permissions
- Supported authorization methods
  - Java Authentication and Authorization Services (JAAS)
  - Custom approach
    - Tivoli Access Manager (TAM) plug-in provided
    - Can create your own

Authorizations refers to the ability to determine what a user is allowed to do. For example, a given user can have permission to *retrieve* information in the grid but not to change it. In eXtreme Scale, resource authorizations are based on standard Javax security *Subject*s and Java security *Permission*s. eXtreme scale allows you to use any implementation of these standard interfaces.

WebSphere eXtreme Scale supports the Java Authentication and Authorization Service (JAAS) authorization model. With JAAS, permissions can be granted based on what code is running and on who is running it. If you do not want to use JAAS authorization eXtreme Scale allows you to implement your own authorization mechanism. eXtreme Scale provides a sample custom authorization plug-in for Tivoli Access Manager.

Authorization levels

- Package com.ibm.websphere.objectgrid.security
- **Map** (MapPermission)
  - Perform insert, read, update, evict, or delete operations on Maps
- **Grid** (ObjectGridPermission)
  - Perform object or entity queries and stream queries on ObjectGrid objects
  - Create a dynamic map
- **DataGrid agent** (AgentPermission)
  - Deploy DataGrid agents to an eXtreme Scale grid
  - Start an agent on the server side
- **Server side map authorization** (ServerMapPermission)
  - Replicate a server map to client side
  - Create or remove a dynamic index on the server map
- **Administration**
  - Perform administration tasks

The following authorizations can be given to a client or group.

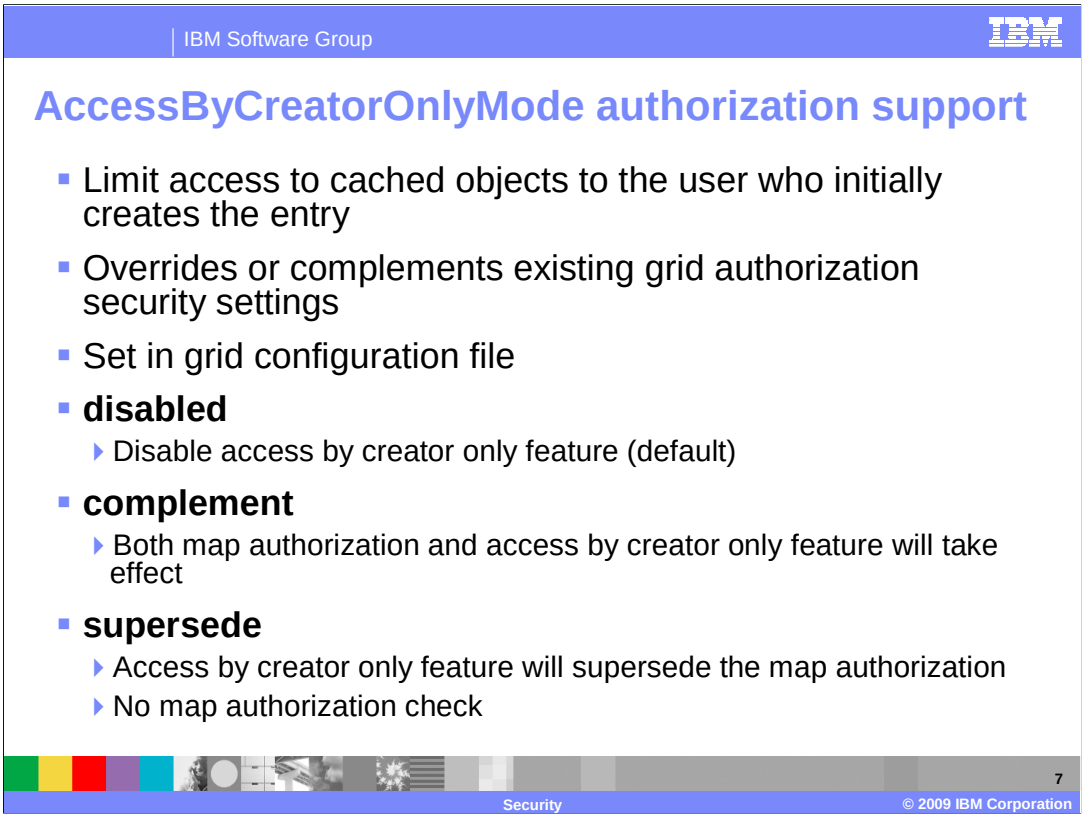
MapPermission defines a user's ability to create, retrieve, update, or remove data in individual ObjectMaps or JavaMaps.

Use the **ObjectGridPermission** to define permission to run object or entity queries, and to create a dynamic map from a template.

ServerMapPermission defines permissions for a client to replicate a server map to a near cache, and to create or remove dynamic indexes on the server.

The **AgentPermission** maps a user's permission to run specific server side MapGridAgents or ReduceGridAgents on remote maps.

Finally, you can grant a user permission to perform administrative actions using the Java Management Extensions (JMX) standard **MBeanPermission** since all administrative actions are done by MBean operations.

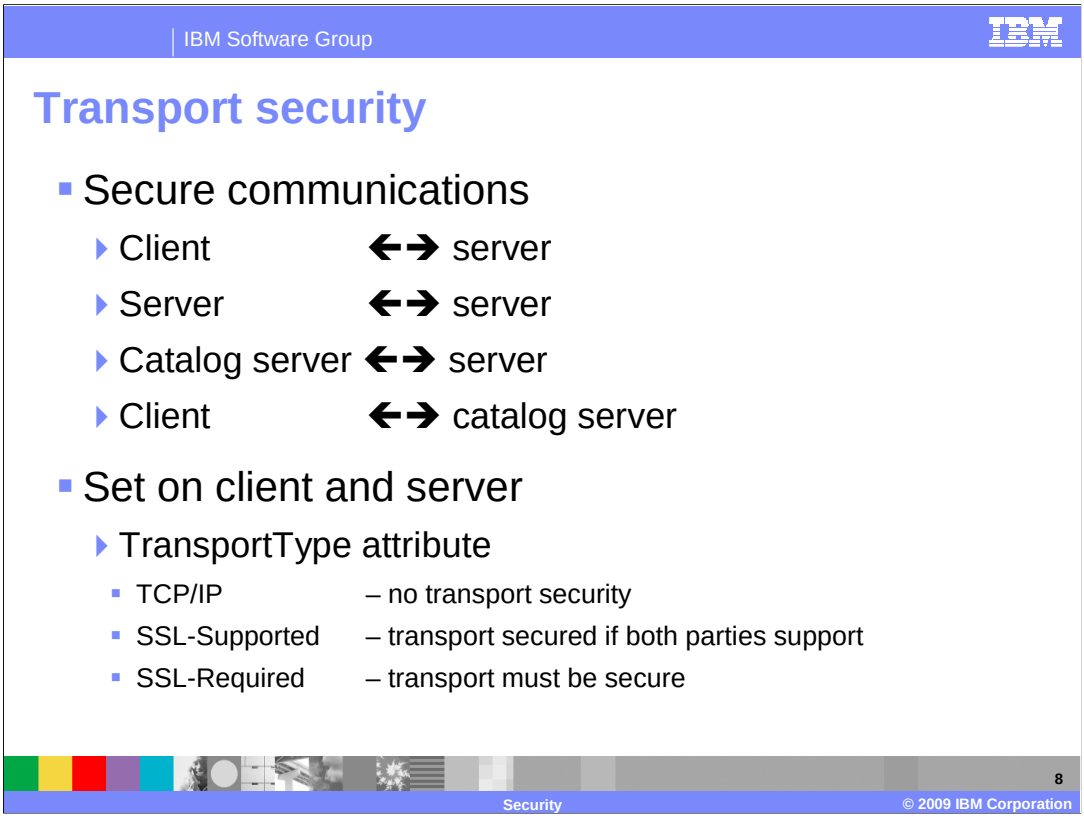


In addition, extreme scale allows you to restrict access to an object only allowing access by the creator of the entry. Access by creator only authorization ensures that only the user Principal who inserts an entry into the ObjectGrid map can access that entry. You can set this feature to either complement or override any other authorization.

When set to complement mode, both map authorization and access by creator only feature will take effect. This allows you to further limit the operations to the data. For example, you can use map permissions to specify that the creator can read but not invalidate the data, and access by creator only to ensure that only the creator can read, update, or delete the data.

When set to supersede, the access by creator only feature will completely replace any map authorizations.

Access by creator only is configured in the ObjectGrid configuration XML file or programmatically on the ObjectGrid object. This feature is disabled by default.



In a secure environment, it is critical to ensure that communications between components, clients and servers, are secure. That is, the actual messages passed between components must be secure from interception or modification.

WebSphere eXtreme Scale optionally supports TCP/IP's Transport Layer Security Secure Sockets Layer (TLS/SSL) for secure communication between clients and servers. When enabled, TLS/SSL provides secure communication between the client and server. The communication mechanism that is used depends on the value of the transportType parameter that is specified in the client and server configuration files.

If transport security is set to TCP/IP, communications are not encrypted. If set to SSL-Supported, communications are encrypted only if both components either support or require it. This is the default.

If transport security is set to SSL-Required, communications to this component *must* be encrypted. It the other component's transport security is set to TCP/IP an error will result and communications will fail.

Specify the transportType property in the client security configuration, the container server security configuration, and the catalog server security configuration

Inter-server grid security

- Server-to-server security
- Shared secret key
  - Server joining grid is challenged to present the secret string
  - Server only allowed to join if its secret key matches the one in the master server
- SecureTokenManager plug-in secures key before sending it
  - Default implementation provided
  - You can provide your own custom implementation

In a secure environment, a server must be able to check the authenticity of another server. WebSphere eXtreme Scale uses a shared secret key string, similar to a shared password, for this purpose.

All of the eXtreme Scale servers in a grid agree on a shared secret string. When a server joins the grid, the server is challenged to present the secret string. If the secret string of the joining server matches the one in the master server, then the joining server can join the grid. Otherwise, the join request is rejected.

Since sending a clear text secret is not secure, the eXtreme Scale security infrastructure provides a SecureTokenManager plug-in to allow the server to secure this secret before sending it. WebSphere eXtreme Scale provides a default implementation which uses a simple algorithm to encrypt and sign the secret. You can provide your own implementation if you need more robust security.

Section

Configuring grid security

This section will describe at a high level how to configure an eXtreme Scale grid for secure operations. More details are available in the WebSphere eXtreme Scale information center

Configuration files

- Grid configuration
- Security descriptor
- Server properties
- Client properties

There are four primary areas where eXtreme Scale security must be configured: the eXtreme Scale grid configuration file, a security descriptor, server properties, and client properties.

Though this presentation describes configuration through XML property files, security configuration is also possible through programming interfaces.

Grid configuration file

- SecurityEnabled
  - Enables security at the grid level
- AuthorizationMechanism
  - Sets the authorization mechanism for the element.
  - You can set the attribute to one of two values
    - AUTHORIZATION_MECHANISM_JAAS (default)
    - AUTHORIZATION_MECHANISM_CUSTOM
- PermissionCheckPeriod
  - How long (seconds) can a server cache a client's access permissions
- AccessByCreatorOnlyMode

The Grid configuration XML file provides the primary mechanism for defining authorization mechanism for objects stored in the grid. Security at the ObjectGrid level is not enabled by default. If you enable security, you can also specify which authorization mechanism to use: Java Authentication and Authorization Service or a custom implementation.

WebSphere eXtreme Scale is able to cache client authorizations to improve performance. The permission check period specifies how often the grid should recheck a client's permissions with the authorization provider. If set to zero, every get, put, update, remove, or evict method call asks the authorization mechanism, either JAAS authorization or custom authorization, to check if the current subject has appropriate permission. A value greater than zero indicates the number of seconds to cache a set of permissions before returning to the authorization mechanism to refresh.

You can also specify access by creator only authorization support to either complement or supersede other authorization settings for the grid.

Security descriptor file

- Configure deployment topology with security enabled
- Security attributes
  - Security
  - authenticator
  - systemCredentialGenerator

The security descriptor file describes the security properties that are common to all servers in the grid, including catalog servers and container servers. Use the security element to define common security attributes, such as whether security is enabled and whether single sign on should be used. The authenticator element specifies the implementation class used to authenticate clients to eXtreme Scale servers in the grid.

The systemCredentialGenerator element is used to set up a system credential generator. The CredentialGenerator object knows how to generate a valid client credential representing a client identity. Examples of a client credential include a user ID and password pair or a Kerberos ticket.

Server properties file

- Properties that define different settings for your server
  - trace settings
  - logging
  - security configuration
  - other
- Used by the catalog service and container servers
- Used to configure eXtreme Scale server security
- Single server property file to specify both basic and security properties

The server properties file contains several properties that define different settings for an individual server, such as trace settings, logging, and security configuration. The server properties file is used by the catalog service and container servers.

Server properties file security properties

- securityEnabled
  - Enables server security
  - Should match catalog server security descriptor setting
- credentialAuthentication
  - Never
  - Supported
  - Required
- Transport layer security settings
- SSL configuration properties

The server properties file allows you to enable or disable security for the server. The server securityEnabled setting should match the securityEnabled property specified in the security descriptor file that is provided to the catalog server.

The credentialAuthentication property indicates whether this server supports credential authentication. You can specify that a server does not support credential authentication, requires a credential, or will support credential authentication if the client also supports credential authentication. If CredentialAuthentication is not required and the user does not provide a credential, the user is considered "anonymous".

You also specify transport layer security in the server properties file. If a server supports or requires SSL transport protocol you must also include SSL configuration properties

Client properties file security properties

- authenticationRetryCount
  - Number of times that authentication is retried if the credential is expired
- credentialGeneratorClass

The client properties file includes the same security properties as the server properties file plus a few extra. The **authenticationRetryCount** property specifies how many times the runtime should attempt to revalidate an expired credential. If the credential is not valid, there is no retry. The **credentialGeneratorClass property** specifies the name of the credential generator class which is used to get credentials for clients.

Server properties file locations

- **objectGridServer.properties** file in the classpath
- System property that specifies a file in the system current directory
  - The file cannot be in the classpath:
  - **-Dobjectgrid.server.props=<*file_name*>**
- -serverProps parameter when you run the startOgServer command
  - **-serverProps <file_name>**
- As a programmatic override
  - ServerFactory.getServerProperties method
  - ServerFactory.getCatalogServerProperties methods
  - The data in the object is populated with the data from the properties files

You can specify the server properties file in one of these ways; a file named objectGridServer.properties in the class path; a system property, objectgrid.server.props, that specifies a the name of a file in the file system; by specifying the serverProps parameter to the startOgServer command; or programmatically using the ServerFactory.getServerProperties and ServerFactory.getCatalogServerProperties methods.

Specifying a setting by using one of the items later in the list overrides the previous setting. For example, if you specify a system property value for the server properties file, the properties in that file override the values in the objectGridServer.properties file that is in the classpath.

Starting a secure catalog server

- Security-related files used
  - Security descriptor file
  - Server properties file
  - Grid configuration file
- Secure catalog server options:
  - -clusterSecurityFile <security descriptor file>
  - -clusterSecurityUrl <security descriptor file URL>
    - security properties that are common to all servers
  - -serverProps <server properties file>
    - Includes server-specific security properties

Start a secure catalog service by supplying the security descriptor file, which describes the security properties common to all servers, and the server properties file, which contains the server-specific security properties.

Starting a secure container server

- Security-related files used
  - Server properties XML file
  - Grid configuration file
- Start a container using a server properties file.
  - -serverProps <server properties file>
    - Includes server-specific security properties
  - -objectGridFile <grid configuration file>
    - Includes grid-specific security properties
  - Common security settings retrieved from catalog server

Start a secure grid container server using a server properties file that includes server-specific security properties, and the grid configuration file with grid-specific security properties.

Starting a secure client

- Enable security in client properties file
  - Can also set programmatically using ClientSecurityConfiguration interface
- Specify the client properties file:
  - objectGridClient.properties file in the classpath
  - System property that specifies a file in the system current directory
    - -Dobjectgrid.client.props=<file_name>

An eXtreme Scale client can connect to a server securely using any ObjectGridManager “connect” method which takes a ClientSecurityConfiguration object.

You can specify the client properties file by including a file named objectGridClient.properties in the classpath, or by specifying a system property, objectgrid.client.props, that specifies the name of a file in the file system.

You can access the client’s security properties programmatically using the ClientClusterContext.getClientProperties methods. You cannot configure security properties with this method.

Local security

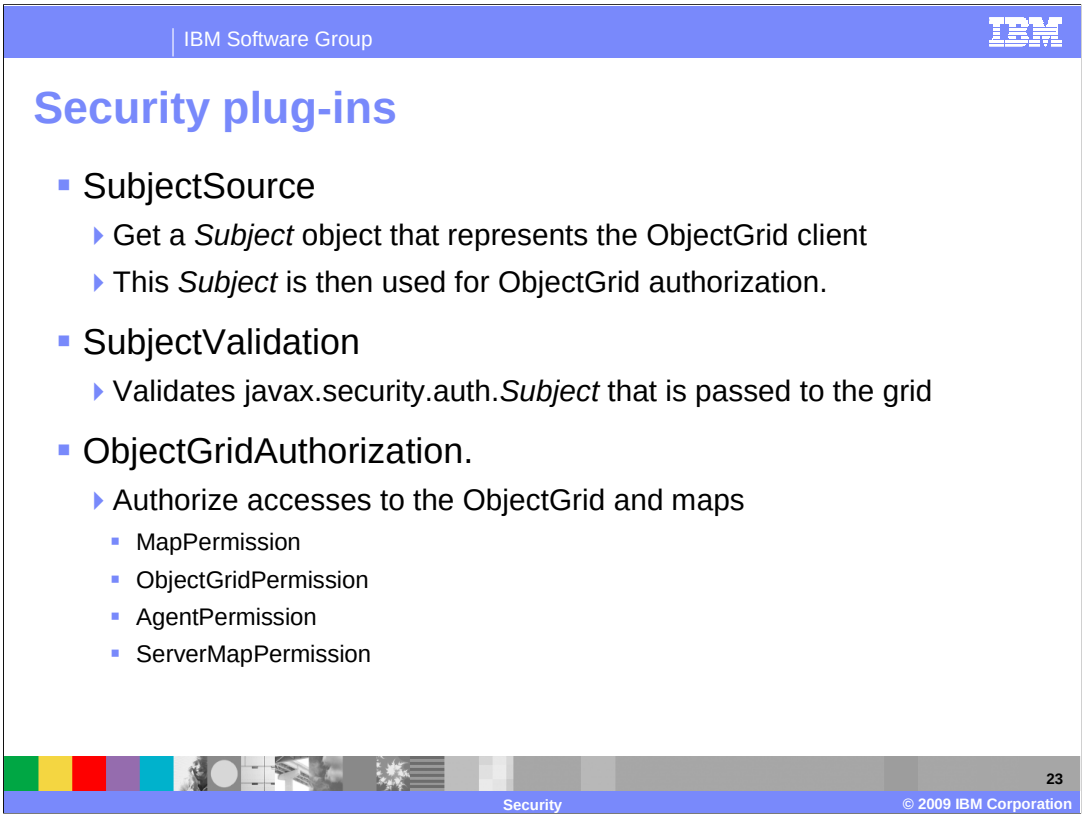
- Different from the distributed model
  - No client-server concept
  - Application directly instantiates and uses grid instance
- Authentication not supported
  - Application manages authentication
  - Passes authenticated Subject to grid
- Same authorization mechanism as client-server model
  - Grid configuration file defines security properties

Local eXtreme Scale security is different from the distributed model because the application directly instantiates and uses an ObjectGrid instance. Your application and eXtreme Scale instances coexist in the same Java virtual machine (JVM). In the local model your applications must manage their own authentication, then pass the authenticated Subject object to the local eXtreme Scale grid instance. The *authorization* mechanism used for the local eXtreme Scale programming model is the same as the client-server model.

Section

System extension points

This section will describe system interfaces which allow you to provide your own authentication and authorization implementations.



WebSphere eXtreme Scale provides several security endpoints to allow custom authentication mechanisms to be integrated with the eXtreme Scale grid.

The **SubjectSource** plug-in is used to get a Subject object, representing a grid client, from an eXtreme Scale environment. The Subject will then be used for eXtreme Scale authorization. This plug-in is used by the eXtreme Scale runtime, and is useful for an already authenticated client. The client can retrieve the authenticated Subject object and then pass it to the ObjectGrid runtime, avoiding another authentication.

The **SubjectValidation** plug-in can be used to validate that a Subject, either passed to the ObjectGrid or retrieved by the SubjectSource plug-in, is a valid Subject that has not been tampered with. You can use this plug-in if you do not trust the Subject object that is passed to a method.

SubjectSource and SubjectValidation are normally only used for local ObjectGrid where a Subject is not available from client server authentication. You can configure the SubjectValidation and SubjectSource plug-ins in the grid configuration XML file or through the ObjectGrid programming interface.

The **ObjectGridAuthorization** plug-in is used to authorize map access for the principals contained in a Subject object. eXtreme Scale provides two default implementations: One that uses the JAAS mechanism for authorization and another implementation that demonstrates the use of Tivoli Access Manager to manage authorizations.

Summary

- Authentication
- Authorization
- Secure transport
- System extension points
- Configuring grid security
  - Grid configuration
  - Security descriptor
  - Server properties
  - Client properties

In summary, WebSphere eXtreme Scale provides support for standard authentication and authorization mechanisms and transport security to ensure the security and integrity of your data. Flexible programming interfaces allow you to easily extend or replace existing security mechanism. eXtreme Scale security is configured either programmatically or by adding entries to XML-based properties files.

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_WXS70_Security.ppt

This module is also available in PDF format at: ../WXS70_Security.pdf

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

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

Tivoli WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

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

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

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.

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.