IBM® Netezza® Analytics
Release 3.3.x.0

Matrix Engine Developer's Guide

Revised: Oct. 05, 2017

Part Number 00J2221-04 Rev. 8

Note: Before using this information and the product that it supports, read the information in Notices and Trademarks on page 17.

Contents

Preface

Audience for This Guide

The nzMatrix Engine is intended for developers and partners interested in working with large matrices on the IBM Netezza appliance. To work with nzMatrix, you should possess a thorough understanding of matrix concepts and linear algebra. You should also be familiar with basic operation and concepts of the IBM Netezza systems and software.

Purpose of This Guide

This guide provides an introduction to Netezza Matrix Engine concepts and usage examples for NZPLSQL stored procedures.

Symbols and Conventions

Note on Terminology: The terms *User-Defined Analytic Process (UDAP)* and *Analytic Executable (AE)* are synonymous.

The following conventions apply:

- Italics for emphasis on terms and user-defined values, such as user input.
- Upper case for SQL commands, for example, INSERT or DELETE.
- Bold for command line input, for example, **`nzsystem stop`**.
- Bold to denote parameter names, argument names, or other named references.
- Angle brackets (< >) to indicate a placeholder (variable) that should be replaced with actual text, for example, **`inza-<release_number>.zip`**.
- A single backslash ("\") at the end of a line of code to denote a line continuation. Omit the backslash when using the code at the command line, in a SQL command, or in a file.
- When referencing a sequence of menu and submenu selections, the ">" character denotes the different menu options, for example, **Menu Name > Submenu Name > Selection**.

If You Need Help

If you are having trouble using the IBM Netezza appliance, IBM Netezza Analytics or any of its components:

1. Retry the action, carefully following the instructions in the documentation.
2. Go to the IBM Support Portal at: http://www.ibm.com/support. Log in using your IBM ID and password. You can search the Support Portal for solutions. To submit a support request, click the **Service Requests & PMRs** tab.
3. If you have an active service contract maintenance agreement with IBM, you may contact customer support teams via telephone. For individual countries, please visit the Technical Support section of the IBM Directory of worldwide contacts (http://www14.software.ibm.com/webapp/set2/sas/f/handbook/contacts.html#phone)

Comments on the Documentation

We welcome any questions, comments, or suggestions that you have for the IBM Netezza documentation. Please send us an e-mail message at netezza-doc@wwpdl.vnet.ibm.com and include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

We appreciate your comments.

CHAPTER 1
Introduction to the Netezza Matrix Engine

The Netezza Matrix Engine (nzMatrix) is a parallelized linear algebra package. It provides a wide range of computations and manipulations that can be run on very large matrices. Using fully distributed and parallelized memory, processing, and database storage, nzMatrix performs computations comparable in size to the available aggregate RAM of the Netezza appliance. It scales to hundreds of gigabytes of RAM, hundreds of processing cores, and hundreds of terabytes of data storage. The Netezza Matrix Engine provides access to highly scalable matrix operations.

With nzMatrix, matrices are transparently stored in IBM Netezza Analytics, allowing matrix data transfer directly to and from the database and the distributed matrix engine.

Netezza Matrix Engine Functions

The Netezza Matrix Engine provides a wide variety of functions:

- Linear algebra functions
  - Matrix multiplication
  - Linear equation solving
  - Linear least squares
  - Matrix inversion
  - Eigenvalues and eigenvectors
  - Singular value decomposition
- Matrix element-wise, pair-wise, and reduction functions
- Inquiry and housekeeping functions
- Random number generators (RNGs)

Applications for nzMatrix

The Netezza Matrix Engine has many areas of application in business and research:

- Principal Component Analysis (PCA)
- Clustering and prediction
- Dimension reduction for data mining
- Customer segmentation, behavioral targeting, customer loyalty
- Social network analysis, such as marketing and counter-terrorism
- Unsupervised learning, latent semantic indexing
- Signal processing, image processing, data compression
- Recommender systems, such as the Netflix Prize
- Microarray Data Analysis, such as bioinformatics and genomics
- Mathematical finance

Understanding nzMatrix

nzMatrix Layered Architecture

nzMatrix has a layered architecture, consisting of an NZPLSQL layer with its API and a C++ layer, which is not exposed as an API. The C++ layer performs many of its functions by calling the Intel® MKL library, including PBLAS and ScaLAPACK routines. PBLAS and ScaLAPACK are proven libraries of highly scalable distributed matrix routines that rely on BLACS and MPI for interprocess communication. For more information on UDAPs, refer to the *User-Defined Analytic Process Developer's Guide*.

Matrix Storage and Access

The Netezza Matrix Engine stores data in the currently connected database using standard Netezza database tables beginning with the prefix **NZ_MAT_**. As a best practice, you should set privileges to hide these tables from direct access by standard users. The standard Netezza Analytics privilege-granting scripts, **create_inza_db.sh** and **create_inza_db_user.sh** are used to modify the permissions. While elevated privileges allow direct access to nzMatrix tables, it is not recommended because direct access could result in code incompatibility with future versions of nzMatrix or data corruption.

The **create_inza_db_user.sh** script grants users the necessary privileges to execute the nzMatrix stored procedures registered in the NZM database. It is recommended that you create additional databases for connection purposes, rather than using the NZM database.

The nzMatrix NZPLSQL stored procedures allow interaction with matrices. nzMatrix provides routines for creating matrices from and saving matrices to user-accessible Netezza database tables in Row/Column/Value format. nzMatrix also provides routines for listing and deleting matrices stored in the database as well as for creating identity matrices, random matrices, and others.

When a given database is shared, the matrix namespace is also shared. Therefore, if a matrix is created while connected to a shared database, that matrix is accessible to any user who has the proper privileges to work in the shared database. This is different than tables where, by default, users do not have visibility into other users' tables. Note that you cannot create a new matrix with the same name in the shared database without first deleting the previously created matrix.

nzMatrix General Limitations

nzMatrix uses the 64-bit double precision floating point approximate numeric data type for storage and computation of matrix element values. Row and column indices are stored as 32-bit integer values, allowing up to 2,147,483,647 rows and columns. Row and column indices begin at 1; a zero index value is not permitted.

nzMatrix computations are automatically distributed across the Netezza appliance. Computations that make use of PBLAS and ScaLAPACK are automatically queued FIFO and run one at a time. If a computational step that uses PBLAS or ScaLAPACK is being performed, any other users' attempts to do the same are queued.

To check which tasks are queued, if any, use the following SQL command:

```
CALL NZA..SP_MPI_STATS();
```

To abort a task, use the following SQL command, replacing the Job Task ID as needed:

```
CALL NZM..KILL_ENGINE(<job_task_ID>);
```

Calculations using PBLAS or ScaLAPACK consume S-Blade RAM for temporarily holding input matrices, intermediate work matrices, and result matrices. Each matrix element consumes 8 bytes. Exceeding available RAM may result in aborted computation and an "Out of memory" user error. Available RAM equals total RAM minus the RAM requirements of the Linux operating system, the Netezza software, and concurrent, unrelated queries.

For example, an IBM Netezza 1000-12 with 12 S-Blades and 16 GB of RAM per S-Blade, performing the matrix multiplication of two 65,000 x 65,000 element matrices, consumes its available RAM and takes approximately 1 hour. Performing the Singular Value Decomposition of a 45,000 x 45,000 element matrix also consumes the available RAM and takes approximately 7 hours. These are maximum limits and assume minimal concurrent queries.

Scalability

Memory requirements and computation time grow rapidly as matrix dimensions increase. The amount of memory required to hold a matrix with *n* rows and *n* columns is proportional to *n* squared, and the number of operations required to multiply, invert, or decompose such matrices is proportional to approximately *n* cubed.

Random Number Generators

Netezza Analytics provides a set of wrappers for the Intel® Math Kernel Library random number generators (RNGs). The SQL API provides a set of stored procedures that generate matrices filled with

pseudorandom values drawn from given probability distribution. See the *Netezza Matrix Engine Reference Guide* for a description of each stored procedure. See *Intel® Math Kernel Library Vector Statistical Library Notes* for more information on the RNGs.

Bulk Algorithms

From Version 3, IBM Netezza Analytics provides bulk algorithms that you can use to work on many smaller matrices in parallel.

The bulk algorithms are:

- Bulk matrix operations
- Bulk linear regression
- Bulk principal components analysis (PCA)

For detailed information about bulk algorithms, see the *IBM Netezza In-Database Analytics Developer's Guide*.

CHAPTER 2
Using nzMatrix: An Example

Enabling a Database for nzMatrix Use

To enable nzMatrix functionality, follow the instructions below:

1. Create a new Netezza Analytics-enabled database:

```
$ cd /nz/export/ae/utilities/bin
$ ./create_inza_db.sh MYDB
```

2. Connect to the database:

```
$ nzsql MYDB
```

3. Initialize the nzMatrix engine for the particular database:

```
CALL NZM..INITIALIZE();
```

Refer to the *IBM Netezza Analytics Administrator's Guide* for additional examples and detailed information on creating Netezza Analytics-enabled databases and setting database permissions.

nzMatrix Using the SQL Interface

This example demonstrates using nzMatrix by calling NZPLSQL stored procedures. It illustrates a wide variety of nzMatrix capabilities, ranging from simple matrix addition to complex linear algebra operations such as Singular Value Decomposition (SVD). The example is not intended to solve a specific issue, but rather to illustrate syntax and usage. Refer to the *Matrix Engine Reference Guide* for additional examples and detailed information on each stored procedure.

Note: nzMatrix does not overwrite existing matrices. Therefore, you may need to delete any pre-existing matrices before running this example and again after the example is complete. The DELETE_ALL_MATRICES() procedure deletes all matrices in the current database and should be used only if you are certain that the database does not contain matrices needed by other users. If the

database does contain matrices belonging to another user, use the Delete_MATRIX() procedure to delete matrices by name.

nzMatrix Run from Netezza SQL

This example can be run from the nzsql command line or from any other SQL-based application.

```
-- Note: All procedures return a value of true, except where noted.

-- Initialize nzMatrix
CALL NZM..INITIALIZE();

-- Uncomment the next line to delete all matrices in the current database
-- CALL NZM..DELETE_ALL_MATRICES();

-- Generate matrices
CALL NZM..UNIFORM('A', 5, 5); -- Uniformly distributed random numbers
CALL NZM..NORMAL('B', 5, 5); -- Normally distributed random numbers
CALL NZM..CREATE_RANDOM_MATRIX('A53', 5, 3); -- Same as UNIFORM
CALL NZM..CREATE_IDENTITY_MATRIX('A_IDENT', 5); -- Identity matrix
CALL NZM..CREATE_ONES_MATRIX('A_ONES', 5, 5); -- Matrix of ones

-- Set and Get the value of a matrix element
CALL NZM..SET_VALUE('A', 3, 2, 0.12345);
CALL NZM..GET_VALUE('A', 3, 2); -- Returns 0.12345

-- Bulk export of matrix data to a Netezza table
DROP TABLE MYTABLE; -- In case MYTABLE already exists
CALL NZM..CREATE_TABLE_FROM_MATRIX('A53', 'MYTABLE');
SELECT * FROM MYTABLE ORDER BY ROW, COL; -- Confirm contents of MYTABLE

-- Create a matrix from a table - bulk import of data
CALL NZM..CREATE_MATRIX_FROM_TABLE('MYTABLE', 'B53', 5, 3);
-- Confirm that B53 has been correctly created
CALL NZM.._TEST_DENSE_VALID('B53'); -- Returns true
CALL NZM..LIST_MATRICES(); -- Returns current list of matrices
CALL NZM..GET_NUM_ROWS('B53'); -- Returns 5
CALL NZM..GET_NUM_COLS('B53'); -- Returns 3

-- Element-by-element add, subtract, multiply, divide
CALL NZM..ADD('A', 'B', 'A_PLUS_B');
CALL NZM..SUBTRACT('A', 'B', 'A_MINUS_B');
CALL NZM..MULTIPLY_ELEMENTS('A', 'B', 'A_TIMES_ELEM_B');
CALL NZM..DIVIDE_ELEMENTS('A', 'B', 'A_DIV_ELEM_B');

-- Element-by-element comparisons
CALL NZM..EQ('A', 'B', 'A_EQ_B'); -- Equal
CALL NZM..NE('A', 'B', 'A_NE_B'); -- Not equal
CALL NZM..GT('A', 'B', 'A_GT_B'); -- Greater than
CALL NZM..GE('A', 'B', 'A_GE_B'); -- Greater than or equal
CALL NZM..LT('A', 'B', 'A_LT_B'); -- Less than
CALL NZM..LE('A', 'B', 'A_LE_B'); -- Less than or equal
CALL NZM..MAX('A', 'B', 'MAX_A_B'); -- Pair-wise maximum
CALL NZM..MIN('A', 'B', 'MIN_A_B'); -- Pair-wise minimum

-- Functions of matrix elements
```

```
CALL NZM..ABS_ELEMENTS('A','A_ABS'); -- Absolute value
CALL NZM..CEIL_ELEMENTS('A','A_CEIL'); -- Ceiling
CALL NZM..FLOOR_ELEMENTS('A','A_FLOOR'); -- Floor
CALL NZM..EXP_ELEMENTS('A','A_EXP'); -- Exponential
CALL NZM..LN_ELEMENTS('A','A_LN'); -- Natural logarithm
CALL NZM..LOG_ELEMENTS('A','A_LOG'); -- Base 10 logarithm
CALL NZM..INT_ELEMENTS('A','A_INT'); -- Truncate toward zero

-- Round to the specified number of decimal digits
CALL NZM..ROUND_ELEMENTS('A', 'A_ROUND', 2);

-- Show the contents of a matrix
CALL NZM..PRINT('A_ROUND', FALSE); -- Returns the contents of A_ROUND

-- Modulo (remainder) function
CALL NZM..MOD_ELEMENTS('A', 'A_MOD', 3);

-- Reductions
CALL NZM..RED_MAX('A'); -- Returns the maximum value in A
CALL NZM..RED_MIN('A'); -- Returns the minimum value in A
CALL NZM..RED_MAX_ABS('A'); -- Returns the maximum absolute value in A
CALL NZM..RED_MIN_ABS('A'); -- Returns the minimum absolute value in A
CALL NZM..RED_SUM('A'); -- Returns the sum of the values in A
CALL NZM..RED_SSQ('A'); -- Returns the sum of the squares of the values
in A
CALL NZM..RED_TRACE('A'); -- Returns the trace of A (the sum of the
diagonal
--                            elements)

CALL NZM..ANY_NONZERO('A'); -- Returns one if any of A's values are
nonzero.
--                            Otherwise, returns zero.
CALL NZM..ALL_NONZERO('A'); -- Returns one if all of A's values are
nonzero.
--                              Otherwise, returns zero.

-- Copy a submatrix
CALL NZM..COPY_SUBMATRIX('A', 'A_SUB', 2, 3, 1, 4);
CALL NZM..GET_NUM_ROWS('A_SUB'); -- Returns 2
CALL NZM..GET_NUM_COLS('A_SUB'); -- Returns 4

-- Transpose (interchange rows and columns)
CALL NZM..TRANSPOSE('A', 'A_TRANS');

-- Concatenate vertically (number of columns stays the same)
CALL NZM..CONCAT('A', 'B', 'A_B_CONCAT_V', 'V');
CALL NZM..GET_NUM_ROWS('A_B_CONCAT_V'); -- Returns 10
CALL NZM..GET_NUM_COLS('A_B_CONCAT_V'); -- Returns 5

-- Concatenate horizontally (number of rows stays the same)
CALL NZM..CONCAT('A', 'B', 'A_B_CONCAT_H', 'H');
CALL NZM..GET_NUM_ROWS('A_B_CONCAT_H'); -- Returns 5
CALL NZM..GET_NUM_COLS('A_B_CONCAT_H'); -- Returns 10

-- Kronecker product
CALL NZM..KRONECKER('A', 'B', 'A_B_KRON');
CALL NZM..GET_NUM_ROWS('A_B_KRON'); -- Returns 25
```

```
CALL NZM..GET_NUM_COLS('A_B_KRON'); -- Returns 25

-- Housekeeping
CALL NZM..MATRIX_EXISTS('A');
CALL NZM..DELETE_MATRIX('A');
CALL NZM..MATRIX_EXISTS('A'); -- Returns false

--
-- Linear algebra
--

CALL NZM..CREATE_RANDOM_MATRIX('A', 1000, 1000);

-- Matrix inversion
CALL NZM..INVERSE('A', 'A_INV');

-- Matrix multiplication
CALL NZM..GEMM('A', 'A_INV', 'A_A_INV');

-- Verify that A_INV is the inverse of A, within floating point precision
CALL NZM..CREATE_IDENTITY_MATRIX('I', 1000);
CALL NZM..SUBTRACT('A_A_INV', 'I', 'A_A_INV_MINUS_I');
CALL NZM..RED_MAX_ABS('A_A_INV_MINUS_I'); -- Returns a value close to
zero

-- Linear Equation Solving
-- Solve A X = RHS for X
CALL NZM..NORMAL('RHS', 1000, 10);
CALL NZM..SOLVE('A', 'RHS', 'X');

-- Verify that A X = RHS, within floating point precision
CALL NZM..GEMM('A', 'X', 'A_X');
CALL NZM..SUBTRACT('A_X', 'RHS', 'A_X_MINUS_RHS');
CALL NZM..RED_MAX_ABS('A_X_MINUS_RHS'); -- Returns a value close to zero

-- Linear Least Squares
-- For over/underdetermined systems A X_LLS = RHS,
-- find linear least squares solution X_LLS
CALL NZM..SOLVE_LINEAR_LEAST_SQUARES('A', 'RHS', 'X_LLS');

-- Singular Value Decomposition
CALL NZM..SVD('A', 'U', 'S', 'VT');

-- Create a diagonal matrix from the one-column matrix S
CALL NZM..VEC_TO_DIAG('S', 'SIGMA');

-- Verify that A = U SIGMA VT, within floating point precision
CALL NZM..GEMM('U', 'SIGMA', 'U_SIGMA');
CALL NZM..GEMM('U_SIGMA', 'VT', 'U_SIGMA_VT');
CALL NZM..SUBTRACT('A', 'U_SIGMA_VT', 'A_MINUS_U_SIGMA_VT');
CALL NZM..RED_MAX_ABS('A_MINUS_U_SIGMA_VT'); -- Returns a value close to
zero

-- Eigenvalues and eigenvectors of a symmetric matrix
-- Create a symmetric matrix
CALL NZM..TRANSPOSE('A', 'A_TRANSP');
CALL NZM..ADD('A', 'A_TRANSP', 'A_SYMM');
```

```
-- Compute Eigenvalues W and Eigenvectors Z
CALL NZM..EIGEN('A_SYMM', 'W', 'Z');
```

Note: In actual operation, the CALL NZM.._TEST_DENSE_VALID(); command used in the example outputs notice messages that provide some informational text on the tests it is performing. The call ultimately returns TRUE or FALSE, depending on whether the matrix passes all of the validity tests. The notice text was omitted from the example for clarity.

APPENDIX A
Notices and Trademarks

Notices

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant 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 character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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 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.

Licensees 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:
IBM Corporation
26 Forest Street
Marlborough, MA 01752 U.S.A.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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. (enter the year or years). All rights reserved.

Trademarks

IBM, the IBM logo, ibm.com and Netezza are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. 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 IBM trademarks is available on the Web at “Copyright and trademark information” at ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

NEC is a registered trademark of NEC Corporation.

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

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

Red Hat is a trademark or registered trademark of Red Hat, Inc. in the United States and/or other countries.

D-CC, D-C++, Diab+, FastJ, pSOS+, SingleStep, Tornado, VxWorks, Wind River, and the Wind River logo are trademarks, registered trademarks, or service marks of Wind River Systems, Inc. Tornado patent pending.

APC and the APC logo are trademarks or registered trademarks of American Power Conversion

Corporation.

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

Regulatory and Compliance

Regulatory Notices

Install the NPS system in a restricted-access location. Ensure that only those trained to operate or service the equipment have physical access to it. Install each AC power outlet near the NPS rack that plugs into it, and keep it freely accessible. Provide approved circuit breakers on all power sources.

Product may be powered by redundant power sources. Disconnect ALL power sources before servicing. High leakage current. Earth connection essential before connecting supply. Courant de fuite élevé. Raccordement à la terre indispensable avant le raccordement au réseau.

Homologation Statement

This product may not be certified in your country for connection by any means whatsoever to interfaces of public telecommunications networks. Further certification may be required by law prior to making any such connection. Contact an IBM representative or reseller for any questions.

FCC - Industry Canada Statement

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case users will be required to correct the interference at their own expense.

This Class A digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Cet appareil numérique de la classe A respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

CE Statement (Europe)

This product complies with the European Low Voltage Directive 73/23/EEC and EMC Directive 89/336/EEC as amended by European Directive 93/68/EEC.

Warning: This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

VCCI Statement

この装置は、情報処理装置等電波障害自主規制協議会 （VCCI） の基準に基づくクラス A 情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起越すことがあります。この場合には使用者が適切な対策を講ずるう要求されることがあります。