HP NonStop SQL/MX Release 3.1 Database and Application Migration Guide

Abstract

This manual explains how to migrate databases and applications from SQL/MX Release 2.3.x and SQL/MX Release 3.0 to SQL/MX Release 3.1, and how to manage different versions of NonStop SQL/MX.

Product Version

NonStop SQL/MX Release 3.1

Supported Release Version Updates (RVUs)

This publication supports J06.12 and all subsequent J-series RVUs and H06.23 and all subsequent H-series RVUs, until otherwise indicated by its replacement publications.

Part Number	Published
663853-001	October 2011

Document History

Part Number	Product Version	Published
540435-002	NonStop SQL/MX Releases 2.0, 2.1, 2.2	February 2006
540435-005	NonStop SQL/MX Release 2.0, 2.1, 2.2, 2.3, 2.3.1, 2.3.2, 2.3.3, 2.3.4	April 2010
666210-001	NonStop SQL/MX Release 3.0	June 2011
663853-001	NonStop SQL/MX Release 3.1	October 2011

Legal Notices

Printed in the US

What's New in This Manual

Manual Information

Abstract

This manual explains how to migrate databases and applications from SQL/MX Release 2.3.x and SQL/MX Release 3.0 to SQL/MX Release 3.1, and how to manage different versions of NonStop SQL/MX.

Product Version

NonStop SQL/MX Release 3.1

Supported Release Version Updates (RVUs)

This publication supports J06.12 and all subsequent J-series RVUs and H06.23 and all subsequent H-series RVUs, until otherwise indicated by its replacement publications.

Part Number	Published
663853-001	October 2011

Document History

Part Number	Product Version	Published
540435-002	NonStop SQL/MX Releases 2.0, 2.1, 2.2	February 2006
540435-005	NonStop SQL/MX Release 2.0, 2.1, 2.2, 2.3, 2.3.1, 2.3.2, 2.3.3, 2.3.4	April 2010
666210-001	NonStop SQL/MX Release 3.0	June 2011
663853-001	NonStop SQL/MX Release 3.1	October 2011

New and Changed Information

The following are the changes to the HP NonStop SQL/MX 3.1 Database and Application Migration Guide:

- Modified content in Considerations for upgrading to SQL/MX Release 3.1 on page 2-1
- Added content in Considerations for upgrading to SQL/MX Release 3.1 on page 2-1
- Added content related to New features in SQL/MX Release 3.1 on page 2-5.
- Modified information related to Versions of SQL/MX software components on page 3-7.
- Modified Fallback paths on page 5-1.

- Added Removing Security Administrator Grants on page A-1.
- Added Identifying Tables That Contain an IDENTITY Column on page B-1.
- Added Identifying tables where the ownership has changed on page C-1
- Made minor modifications throughout the manual.

HP NonStop SQL/MX Release 3.1 Database and Application Migration Guide

Index | Figures | Tables

Figures

Tables

About This Manual

This manual explains how to migrate databases and applications from SQL/MX Release 2.3.x and SQL/MX Release 3.0 to SQL/MX Release 3.1, and how to manage different versions of SQL/MX. NonStop SQL/MX is a relational database management system for the HP NonStop server and is based on the ANSI/ISO/IEC 9075:1999 SQL standard, commonly referred to as SQL:1999.

Throughout this manual, references to SQL/MX Release 2.x indicate SQL/MX Release 2.0, SQL/MX Release 2.1, SQL/MX Release 2.2, and subsequent releases until otherwise indicated in a replacement publication.

Audience

This manual is intended for database administrators (DBAs), HP support personnel, and others who plan to migrate or upgrade databases and database applications from SQL/MX Release 2.3.x and SQL/MX Release 3.0 to SQL/MX Release 3.1.

What This Manual Does Not Cover

- Migration from NonStop SQL/MP to NonStop SQL/MX.
- Migration from other database vendors to NonStop SQL/MX. For assistance, contact your service provider. For more information, see Assistance with migration on page 1-3.
- Migration of SQL/MX, SQL/MP, and Enscribe databases and applications from an HP NonStop S-series system to an HP Integrity NonStop NS-series system. For more information, see the *NonStop NS-Series Database Migration Guide*.
- Migration from Enscribe databases and applications to NonStop SQL/MX. An automated tool to convert Enscribe databases and applications to NonStop SQL/MX does not exist. The Escort SQL product converts Enscribe databases and applications to NonStop SQL/MP. For more information, see the *ESCORT SQL™ User's Guide*, which you can obtain from Carr Scott Software Inc. at http://www.carrscott.com/evaluation_prog.shtml. If there is sufficient market demand, Carr Scott Software Inc. might consider modifying Escort SQL to support direct Enscribe-to-SQL/MX migrations. If you or your customer deems this functionality to be critical to your project, send a note and background on the project to harry.scott@carrscott.com.
- Migration to Format 2 SQL/MP tables. For more information, see the *SQL/MP Installation and Management Guide*.

Organization

Section 1, Overview of SQL/MX Migration	Defines migration terminology, provides planning considerations, and describes where you can go for further assistance.
Section 2, Considerations for upgrading to SQL/MX Release 3.1	Discusses the factors that you must consider before upgrading to SQL/MX Release 3.1.
Section 3, Version management and interoperability	Describes SQL/MX releases, interoperability of SQL/MX releases, versions of SQL/MX software components, query plan versioning, and managing a mixed-node network.
Section 4, Upgrading to SQL/MX Release 3.1	Provides guidelines for upgrading to SQL/MX Release 3.1.
Section 5, Falling Back from SQL/MX Release 3.1	Provides guidelines for downgrading to an earlier version of SQL/MX.
Appendix A, Removing Security Administrator Grants	Provides example steps for removing any Security Administrator grants prior to falling back to an earlier release of SQL/MX.
Appendix B, Identifying Tables That Contain an IDENTITY Column	Provides example steps for removing any identifying tables that contain an IDENTITY column as an aid to falling back to an earlier release of SQL/MX.
Appendix C, Identifying tables where the ownership has changed	Provides examples to identify the list of objects in a catalog whose owners are different from their parent schema owners, and the steps to change the ownership of those objects.

Related Documentation

This manual is part of the HP NonStop SQL/MX library of manuals. The following table describes the list of manuals:

Introductory Guides

SQL/MX Comparison Guide for SQL/MP Users	Describes SQL differences between NonStop SQL/MP and NonStop SQL/MX.
SQL/MX Quick Start	Describes basic techniques for using SQL in the SQL/MX conversational interface (MXCI). Includes information about installing the sample database.

Reference Manuals

SQL/MX Reference Manual	Describes the syntax of SQL/MX statements, MXCI commands, functions, and other SQL/MX language elements.
SQL/MX Messages Manual	Describes SQL/MX messages.

SQL/MX Glossary	Defines SQL/MX terminology.

Installation Guides

SQL/MX Installation and Management Guide	Describes how to plan for, install, create, and manage a SQL/MX database. Explains how to use installation and management commands and utilities.
NSM/web Installation Guide	Describes how to install NSM/web and troubleshoot NSM/web installations.

Connectivity Manuals

SQL/MX Connectivity Service Manual	Describes how to install and manage the HP NonStop SQL/MX Connectivity Service (MXCS), which enables applications developed for the Microsoft Open Database Connectivity (ODBC) application programming interface (API) and other connectivity APIs to use NonStop SQL/MX.
SQL/MX Connectivity Service Administrative Command Reference	Describes the SQL/MX administrative command library (MACL) available with the SQL/MX conversational interface (MXCI).
ODBC/MX Driver for Windows	Describes how to install and configure HP NonStop ODBC/MX for Microsoft Windows, which enables applications developed for the ODBC API to use NonStop SQL/MX.

Migration Guides

HP NonStop SQL/MX 2.3.x to SQL/MX 3.0 Database and Application Migration Guide	Describes how to migrate databases and applications from SQL/MX 2.3.x to SQL/MX 3.0.
HP NonStop SQL/MP to SQL/MX Database and Application Migration Guide	Describes how to migrate databases and applications from SQL/MP to SQL/MX.
NonStop NS-Series Database Migration Guide	Describes how to migrate NonStop SQL/MX, NonStop SQL/MP, and Enscribe databases and applications to HP Integrity NonStop NS-series systems.

Data Management Guides

SQL/MX Data Mining Guide	Describes the SQL/MX data structures and operations to carry out the knowledge-discovery process.
SQL/MX Report Writer Guide	Describes how to produce formatted reports using data from a SQL/MX database.
DataLoader/MX Reference Manual	Describes the features and functions of the DataLoader/MX product, a tool to load SQL/MX databases.

Application Development Guides

SQL/MX Programming Manual for C and COBOL	Describes how to embed SQL/MX statements in ANSI C and COBOL programs.
SQL/MX Query Guide	Describes how to understand query execution plans and write optimal queries for a SQL/MX database.
SQL/MX Queuing and Publish/Subscribe Services	Describes how NonStop SQL/MX integrates transactional queuing and publish/subscribe services into its database infrastructure.
SQL/MX Guide to Stored Procedures in Java	Describes how to use stored procedures that are written in Java within NonStop SQL/MX.

Online Help

Reference Help	Overview and reference entries from the *SQL/MX Reference Manual*.
Messages Help	Individual messages grouped by source from the *SQL/MX Messages Manual.*
Glossary Help	Terms and definitions from the *SQL/MX Glossary*.
NSM/web Help	Context-sensitive help topics that describe how to use the NSM/web management tool.
Visual Query Planner Help	Context-sensitive help topics that describe how to use the Visual Query Planner graphical user interface.

The NSM/web and Visual Query Planner help systems are accessible from their respective applications. You can download the Reference, Messages, and Glossary online help from the $SYSTEM.ZMXHELP subvolume or from the HP Business Support Center (BSC). For more information about downloading online help, see the *SQL/MX Installation and Management Guide*.

Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

> This requirement is described under Backup DAM Volumes and Physical Disk Drives on page 3-2.

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
MAXATTACH
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

```
file-name
```

computer type. `Computer type` letters within text indicate C and Open System Services (OSS) keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
myfile.c
```

italic computer type. *`Italic computer type`* letters within text indicate C and Open System Services (OSS) variable items that you supply. Items not enclosed in brackets are required. For example:

```
pathname
```

[ ] Brackets. Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name
```

```
INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num  ]
   [ -num ]
   [ text ]
```

```
K [ X | D ] address
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }
```

```
ALLOWSU { ON | OFF }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address [ , new-value ]...

[ - ] {0|1|2|3|4|5|6|7|8|9}...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;

LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]...
```

!i and !o. In procedure calls, the !i notation follows an input parameter (one that passes data to the called procedure); the !o notation follows an output parameter (one that returns data to the calling program). For example:

```
CALL CHECKRESIZESEGMENT (  segment-id                    !i
                         , error        ) ;              !o
```

!i,o. In procedure calls, the !i,o notation follows an input/output parameter (one that both passes data to the called procedure and returns data to the calling program). For example:

```
error := COMPRESSEDIT ( filenum ) ;                      !i,o
```

!i:i. In procedure calls, the !i:i notation follows an input string parameter that has a corresponding parameter specifying the length of the string in bytes. For example:

```
error := FILENAME_COMPARE_ (  filename1:length          !i:i
                            , filename2:length ) ;      !i:i
```

!o:i. In procedure calls, the !o:i notation follows an output buffer parameter that has a corresponding input parameter specifying the maximum length of the output buffer in bytes. For example:

```
error := FILE_GETINFO_ (  filenum                       !i
                        , [ filename:maxlen ] ) ;       !o:i
```

Notation for Messages

This list summarizes the notation conventions for the presentation of displayed messages in this manual.

Bold Text. Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE
?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

Nonitalic text. Nonitalic letters, numbers, and punctuation indicate text that is displayed or returned exactly as shown. For example:

```
Backup Up.
```

lowercase italic letters. Lowercase italic letters indicate variable items whose values are displayed or returned. For example:

```
p-register
process-name
```

[ ] Brackets. Brackets enclose items that are sometimes, but not always, displayed. For example:

```
Event number = number [ Subject = first-subject-value ]
```

A group of items enclosed in brackets is a list of all possible items that can be displayed, of which one or none might actually be displayed. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
proc-name trapped [ in SQL | in SQL file system ]
```

{ } Braces. A group of items enclosed in braces is a list of all possible items that can be displayed, of which one is actually displayed. The items in the list can be arranged

either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
obj-type obj-name state changed to state, caused by
{ Object | Operator | Service }

process-name State changed from old-objstate to objstate
{ Operator Request. }
{ Unknown.          }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
Transfer status: { OK | Failed }
```

% Percent Sign. A percent sign precedes a number that is not in decimal notation. The % notation precedes an octal number. The %B notation precedes a binary number. The %H notation precedes a hexadecimal number. For example:

```
%005400

%B101111

%H2F

P=%p-register E=%e-register
```

Change Bar Notation

Change bars are used to indicate substantive differences between this manual and its preceding version. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

The message types specified in the REPORT clause are different in the COBOL85 environment and the Common Run-Time Environment (CRE).

The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

1 Overview of SQL/MX Migration

This section introduces the process of migrating to SQL/MX Release 3.1 and discusses the following topics:

Migration terminology

You must be familiar with the following terms, which are used throughout the manual:

Version. A product version or the version of a software component.

Release. The product version of NonStop SQL/MX. A SQL/MX release identifies a set of SQL/MX products as belonging to a given version of NonStop SQL/MX.

SQL/MX application. Uses SQL/MX syntax, is SQL-compiled using MXCMP, and can query SQL/MP and SQL/MX database objects.

SQL/MX Release 2.x. SQL/MX Release 2.0, SQL/MX Release 2.1, SQL/MX Release 2.2, and subsequent releases until otherwise indicated in a replacement publication.

SQL/MX Release 3.x. SQL/MX Release 3.0, SQL/MX Release 3.1, and subsequent releases unless otherwise specified.

Upgrade (also known as Migration). Moving from an older software version to a newer software version. For example, you upgrade to a newer incremental or maintenance release of SQL/MX, such as upgrading from SQL/MX Release 2.3.x to SQL/MX Release 3.1.

Downgrade (also known as Fallback). Reverting from a newer software version to an older software version. For example, you downgrade to an older incremental or maintenance release of SQL/MX, such as downgrading from SQL/MX Release 3.1 to SQL/MX Release 2.3.4.

Planning for migration

You must save a copy of the modules before a system upgrade. If you fall back to the current version at a later stage, you can reinstate this copy after fallback. This method is applicable only if the database layout after the fallback is similar to the layout before the system upgrade.

If the migration can be broken into phases, each phase must include its own complete plan:

Migration Plan	For more information, see...
Plan for migrating applications	• Section 4, Upgrading to SQL/MX Release 3.1
Plan for fallback	• Section 5, Falling Back from SQL/MX Release 3.1
Plan for updating disaster recovery or RDF configuration and procedures	• *SQL/MX Installation and Management Guide*
Plan for validating that migration is successful	• Guidelines for conducting the migration on page 1-2

Guidelines for conducting the migration

To ensure a successful migration, use the following guidelines:

- Conducting the initial migration in a test environment
- Runing baseline tests and gathering data at each migration phase
- Assessing the success of the migration

Conducting the initial migration in a test environment

To avoid disrupting the production database and daily business operations, isolate key applications and database objects to migrate and conduct the initial migration in a test environment. The test environment should simulate the production environment as much as possible.

Runing baseline tests and gathering data at each migration phase

Before you migrate, use Measure to obtain initial baseline data, such as average query run time, CPU time, and process time. At each phase of the migration, gather the same type of data and analyze it. For more information about Measure, see the *Measure User's Guide*.

Assessing the success of the migration

The data that you gather during each migration phase will help you determine the success of migration. Prepare to answer the following questions:

- What did functional, performance, and scalability testing reveal during migration?
- How can you use these results to improve and fine-tune the migration process before migrating applications and database objects in the production environment?

Assistance with migration

Specific strategies for migrating to SQL/MX Release 3.1 depend on your specific database environment and business needs. Without the help of migration experts and special tools, determining and implementing a migration strategy can be time-consuming and costly. The following HP services can help you determine and implement the best migration strategy for your business:

- Product support for NonStop servers
- NonStop solutions development and implementation

Product support for NonStop servers

The Global NonStop Solution Center (GNSC) is staffed by highly trained analysts who support NonStop systems. These analysts can help you migrate your systems. For more information, see the Web site

http://www8.hp.com/us/en/services/services-detail.html?compURI=tcm:245-808927&pageTitle=Product-Support-for-NonStop-Servers

NonStop solutions development and implementation

NonStop Solutions Development and Implementation (SDI) provides a porting and migration service to help you plan and implement application and database migration on your NonStop system. SDI works with you to develop a comprehensive plan for successfully migrating your applications and database while achieving high levels of availability and performance with minimal risk to business continuity.

The service plan identifies the changes required for system software, application and utility programs, development and test environments, database configuration, and database loading. The service plan includes a comprehensive report that describes the scope and strategy for migrating your applications and database, a detailed project plan, and a statement of work for services to help you implement the project plan.

For more information about this service, contact NonStop-SDI-Services@hp.com.

2 Considerations for upgrading to SQL/MX Release 3.1

If you are currently using SQL/MX, to obtain new functionalities, performance enhancements, and bug fixes, you might want to upgrade to a newer version.

This section discusses the factors that you must consider before upgrading to SQL/MX Release 3.1, and addresses the following topics:

Migration path

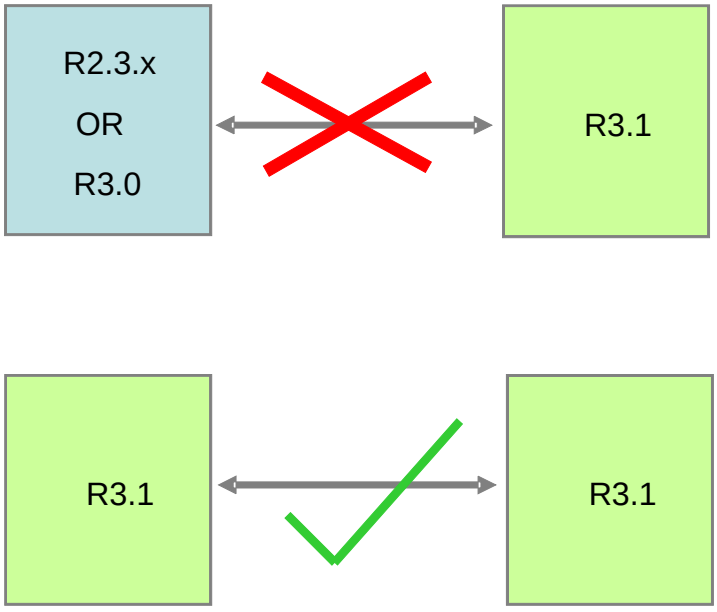
You can migrate from SQL/MX Release 2.3.x or SQL/MX Release 3.0 to SQL/MX Release 3.1, as shown in Figure 2-1.

Figure 2-1. Migration Path from SQL/MX 2.3.x or SQL/MX 3.0 to SQL/MX 3.1

Compatibility of SQL/MX 3.1 systems

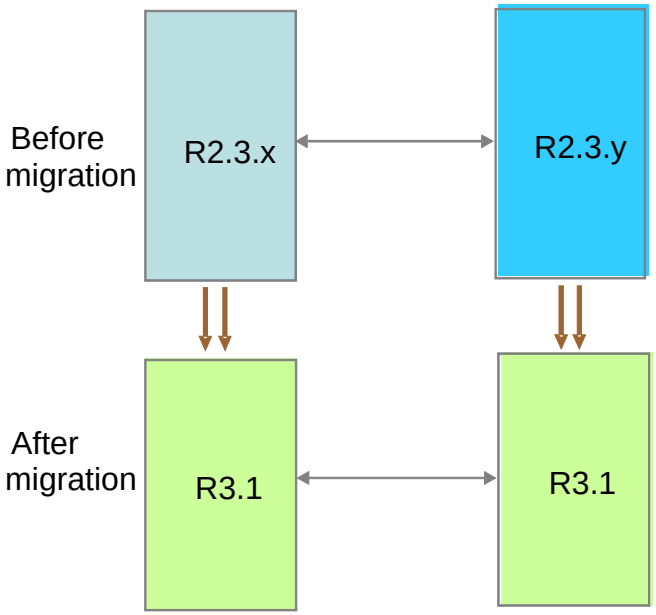
SQL/MX Release 3.1 can coexist only with other instances of SQL/MX Release 3.1 in a network, as shown in Figure 2-2.

Figure 2-2. Compatibility of SQL/MX releases in a network



If your network includes multiple versions of SQL/MX, such as SQL/MX Release 2.3.3 and SQL/MX Release 2.3.4, you must migrate all of them to SQL/MX Release 3.1 to ensure compatibility, as shown in Figure 2-3.

Figure 2-3. Compatibility of SQL/MX releases in a network after migration



Working with v3100, v3000, and v1200 objects in SQL/MX Release 3.1

In SQL/MX Release 3.1, a catalog can contain v1200, v3000, or v3100 schemas, but not a mixture of the versions.

A CREATE SCHEMA statement will create a new schema of the same version as those schemas in the catalog. For example, suppose that a catalog contains some v3000 schemas. The version of a new schema that is created in that catalog will be 3000.

By default, schemas created in new catalogs will be v3100. You do not have to perform additional steps to create v3100 schemas.

You must perform these additional steps only when a v1200 or v3000 schema is required in a new catalog that does not already contain any schema.

To create a v3000 schema in a new catalog, you must enter the following `control query default` command before you issue the CREATE SCHEMA statement:

```
control query default create_definition_schema_version '3000'
```

To create a v1200 schema in a new catalog, you must enter the following `control query default` command before you issue the CREATE SCHEMA statement:

```
control query default create_definition_schema_version '1200'
```

If the system schema version is 3000 or later, you cannot create a v1200 schema. After executing the `UPGRADE ALL METADATA` command, you cannot create v1200 schemas.

Irrespective of the schema version, no explicit action is needed to create or maintain individual database objects such as tables, indexes, and views.

For more information on commands, see SQL Statements in the *SQL/MX Release 3.1 Reference Manual*.

Conversions from numeric to char or varchar

SQL/MX Release 3.x reports 8402 errors during conversions from numeric to char or varchar types with truncation. In SQL/MX Release 2.x, such conversions truncate the data without any errors.

The following examples illustrate overflow errors generated in SQL/MX Release 3.x:

- Example 1

```
>>create table tnumeric (a numeric (18,6));

--- SQL operation complete.

>>

>>insert into tnumeric values (123456789012.345678);

--- 1 row(s) inserted.

>>

>>select cast(a * 10 as char(18)) from tnumeric;

*** ERROR[8402] A string overflow occurred during the
evaluation of a character expression. Conversion of Source
Type:DECIMAL SIGNED(REC_DECIMAL_LSE) Source
Value:0x2B000102030405060708090001020304050607080 to Target
Type:CHAR(REC_BYTE_F_ASCII).

--- 0 row(s) selected.

>>
```

- Example 2

```
>>create table tnumeric (a numeric (18,6));

--- SQL operation complete.

>>
>>insert into tnumeric values (3.0),(123456789012.345678);

--- 2 row(s) inserted.

>>
>>select cast(a * 10 as char(18)) from tnumeric;

(EXPR)
------------------

30.000000

*** ERROR[8402] A string overflow occurred during the
evaluation of a character expression. Conversion of Source
Type:DECIMAL SIGNED(REC_DECIMAL_LSE) Source
Value:0x2B000102030405060708090001020304050607080 to Target
Type:CHAR(REC_BYTE_F_ASCII).

--- 1 row(s) selected.

>>

>>select cast (max(a) as varchar(8)) from tnumeric;
*** ERROR[8402] A string overflow occurred during the
evaluation of a character expression. Conversion of Source
Type:LARGEINT(REC_BIN64_SIGNED) Source
```

```
Value:123456789012345678 to Target
Type:VARCHAR(REC_BYTE_V_ASCII).

--- 0 row(s) selected.

>>

>>select cast(a * a as char(18)) from tnumeric;

(EXPR)

------------------

9.000000000000

*** ERROR[8402] A string overflow occurred during the
evaluation of a character expression. Conversion of Source
Type:DECIMAL SIGNED(REC_DECIMAL_LSE) Source
Value:0x2B000105020401050708070503020308080306050207090608020
909070605020709060804 to Target Type:CHAR(REC_BYTE_F_ASCII).

--- 1 row(s) selected.

>>
>>update tnumeric set a = a + a
+>where cast( a * 20 as char (10)) > '5';

*** ERROR[8402] A string overflow occurred during the
evaluation of a character expression. Conversion of Source
Type:DECIMAL SIGNED(REC_DECIMAL_LSE) Source
Value:0x2B000204060901030507080002040609010305060 0 to Target
Type:CHAR(REC_BYTE_F_ASCII).

--- 0 row(s) updated.
>>
```

New features in SQL/MX Release 3.1

SQL/MX Release 3.1 introduces the following new database features:

- Inclusion of IDENTITY columns in tables
- Capability to change the ownership of database objects
- Capability to restrict catalog and schema creation to predefined set of users
- Capability to define security administrators who can manage access to database objects without having explicit access to the data in those objects.

For more information on the new features, see the *SQL/MX Release 3.1 Reference Manual*.

After upgrading to SQL/MX Release 3.1, you can use the new features on databases and applications created . You can also use the new features on older databases and applications, however, you must upgrade all the metadata before using them.

New metadata in SQL/MX Release 3.1

The change of ownership and separation of privileges features require new metadata tables to store privilege information. These new metadata tables are created when SQL is initialized on SQL/MX Release 3.1 or when the metadata is upgraded. The UPGRADE command creates the SYSTEM_SECURITY_SCHEMA in the system catalog, and then creates and initializes the tables in that schema. When you downgrade the metadata, the SYSTEM_SECURITY_SCHEMA and the tables it contains will be removed.

You can downgrade to v1200 and v3000 schemas.

Note. If the downgrade operation fails, you can use the RECOVER command to undo the effects of the failed downgrade. However, the contents of the tables in the SYSTEM_SECURITY_SCHEMA might be lost.

---

3 Version management and interoperability

This section covers these topics:

NonStop SQL/MX releases

A SQL/MX release identifies a set of SQL/MX products as belonging to a given version of NonStop SQL/MX. These subsections describe topics related to SQL/MX releases:

SQL/MX releases on systems running H-Series RVUs and J-Series RVUs

NonStop SQL/MX can be installed on HP Integrity NonStop NS-series systems and on HP Integrity NonStop BladeSystems. HP Integrity NonStop NS-series systems run H-series Release Version Updates (RVUs). HP Integrity NonStop BladeSystems and some rack-mounted servers run J-series RVUs. Each SQL/MX release is supported by specific release version updates (RVUs). For more information, see RVUs and supported SQL/MX releases on page 3-6.

Each system node can support only one release of NonStop SQL/MX at a time. Table 3-1 on page 3-2 shows the NonStop operating systems that support SQL/MX

Release 3.1.

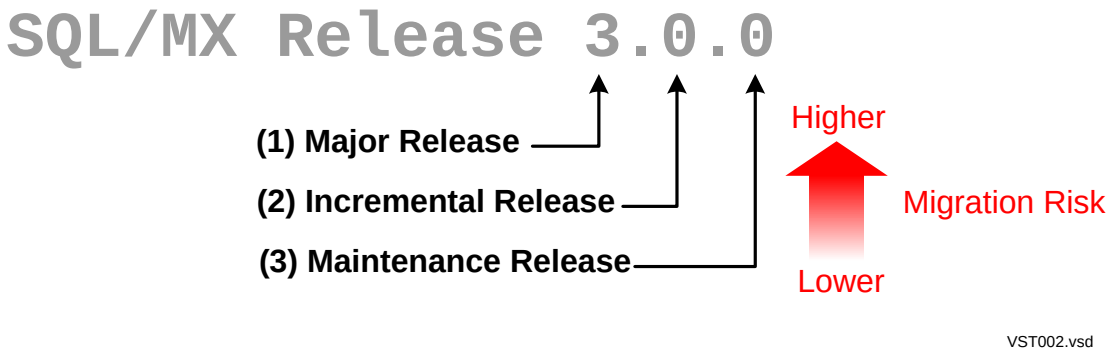
Table 3-1. SQL/MX releases on systems running H-Series or J-series RVUs

SQL/MX Release	Product Version Identifier	SPR Identifier	Availability	Date of Initial Release	Delivery	Supported RVUs
3.1	H31	N/A	General	10/2011	Scout or H06.23 SUT	H06.23, J06.12

Naming scheme for SQL/MX releases

NonStop SQL/MX contains many individual products, which are identified by T-numbers. A SQL/MX release name identifies a set of SQL/MX products as belonging to a given release of NonStop SQL/MX.

Each SQL/MX release is identified by a three-part release identifier. The following is an example of a release identifier:



1. The first number indicates a *major release*, which introduces major new functionality and a new SQL version.
2. The second number indicates an *incremental release*, which introduces an incremental change and some new functionality.
3. The third number indicates a *maintenance release*, which introduces bug fixes and minimal new functionality.

The migration risk increases from maintenance to major releases because major releases introduce significant new functionality that makes fallback more difficult, if implemented.

The first and second levels of the release name (for example, SQL/MX Release 3.0) are based on the product version identifier (for example, H30). Major and incremental releases typically change the product version of NonStop SQL/MX. Maintenance releases and time-critical fixes (TCFs) do not change the product version of NonStop SQL/MX.

Major and incremental releases

Major releases introduce significant new functionality. These releases occur less frequently and usually require you to take specific action to use the new functionality, such as perform an upgrade operation or alter existing applications.

Incremental releases introduce incremental changes and some new functionality, which does not significantly affect existing applications.

In this manual, the SQL/MX releases are identified by major and incremental release numbers such as SQL/MX Release 2.3.4 and SQL/MX Release 3.1.

Maintenance releases

Maintenance releases introduce bug fixes and minimal new functionality. These releases occur most frequently and rarely affect all the component products in the SQL/MX product set. This manual refers to maintenance releases only when they introduce new functionality that requires mentioning.

SQL/MX product T-Numbers and VPROC information

In a given SQL/MX release, participating SQL/MX products include the product version identifier in their T-numbers (for example, T1051H30).

The VPROC information of each SQL/MX product in a given release is:

```
TNNNNPNN_DDMMMYYYY_SPR_RRR_MMDD
```

where:

TNNNNPNN

is the product T-number (for example, T1051) and the product version identifier (*PNN*). The first letter *P* in *PNN* represents the RVU series to which the product version applies:

- H for HP Integrity NS-series systems, which run H-series RVUs
- J for HP Integrity NonStop™ BladeSystems, which run J-series RVUs

The next two digits *NN* represent the SQL/MX release number (for example, 30 for SQL/MX Release 3.0).

DDMMMYYYY

is the VPROC date (for example, 14FEB2011).

SPR

is the three-digit SPR identifier of the release (for example, ACF). For an initial PVU, this field is the same as the product version *PNN* (for example, H30).

RRR

is the three-digit SQL/MX release number (for example, 300 for SQL/MX Release 3.0.0).

MMDD

is the final build date for the product, including the month and day (for example, 1130 for November 30).

For example, this VPROC command displays information for the T1051 product that is associated with SQL/MX Release 3.0.0 for H-series RVUs:

```
T1051H30_14FEB2011_H30_300_1130
```

The SQL/MX Master Softdoc (T0650) for each SQL/MX release refers to the release name and VPROC information.

Product version and SPR identifiers

Each SQL/MX release is associated with a product version identifier and possibly an SPR identifier.

Each product version identifier (for example, “H30”) includes:

- Supported platform of the product using “H” for systems running H-series RVUs or J-series RVUs
- SQL/MX release number (for example, “30” for SQL/MX Release 3.0)

For an initial product version (PV) of NonStop SQL/MX, the product version identifier is not accompanied by an SPR identifier. For example, the initial PV of SQL/MX Release 3.0 on systems running H-series RVUs is referred to as “H30.”

For a product version update (PVU) of NonStop SQL/MX, the product version identifier is accompanied by an SPR identifier. For example, the PVU of SQL/MX Release 3.0 on systems running H-series RVUs is referred to as “H30^ACF” where “ACF” is the SPR identifier.

Availability of SQL/MX releases

Each SQL/MX release is made available to customers as a General Availability (GA) release, a Controlled Availability (CA) release, or an Early Adopter Program (EAP) release.

A GA release is available to any customer who orders the product and whose system meets the prerequisites for using the product.

A CA release limits the shipment of a product to a set of customers who agree to specific use conditions, which include migrating to a GA release when it is ready. For example, the SQL/MX DDL license (T0394) has controlled availability for current SQL/MX releases.

An EAP release introduces a new product to customers at a very early stage and is a type of test program conducted at a customer site.

For more information about these types of releases, see the *Managing Software Changes* manual.

Delivery of SQL/MX releases

Each SQL/MX release is delivered as a set of Software Product Revisions (SPRs). Some SQL/MX SPRs are first available for downloading from Scout for NonStop Servers and are later available on the next RVU's site update tape (SUT). Other SQL/MX SPRs are available only on specific RVU SUTs and are not available in Scout for NonStop Servers. Rarely, as is the case for SQL/MX Release 3.0 (H30), which is an Early Product Delivery (EPD), the SQL/MX SPRs must be ordered in Scout for NonStop Servers as a backup tape delivery.

For information about the delivery of SQL/MX Release 3.1, see Table 3-1 on page 3-2. To learn how to acquire a SQL/MX release that is not listed in those tables, check the SQL/MX Master Softdoc (T0650) for that SQL/MX release.

RVUs and supported SQL/MX releases

Each SQL/MX release is supported by specific Release Version Updates (RVUs). An RVU is a collection of compatible revisions of HP NonStop operating system software products that are shipped and supported as a unit.

Before downloading a set of SQL/MX SPRs from Scout for NonStop Servers, verify that the RVU of your system supports that release of NonStop SQL/MX. For more information, see Table 3-1 on page 3-2.

Table 3-2 lists the RVUs of NonStop systems and the SQL/MX releases that they support. You can install only one SQL/MX release on a system at a time.

Table 3-2. RVUs and Supported SQL/MX Releases

RVUs	Supported SQL/MX Releases
H06.23 and later J06.12 and later	SQL/MX Release 3.1

Interoperability of SQL/MX releases

SQL/MX 3.1 does not interoperate with other versions of SQL/MX in an EXPAND network. All systems in the EXPAND network must have SQL/MX 3.1 software installed.

Versions of SQL/MX software components

Each SQL/MX software component and persistent entity has a version identifier that identifies its software release and that can be understood by other SQL/MX releases. Versioned software components include the SQL/MX system software version (MXV).

Versioned persistent entities, which persist over time and across releases, include:

- Schemas that contain SQL/MX database objects
- SQL/MX database objects, such as tables, views, and indexes
- Compiled modules

NonStop SQL/MX uses version identifiers to determine if a software component or persistent entity is compatible with the SQL/MX release. Before using a software component or operating on a persistent entity, NonStop SQL/MX checks the version identifier of the software component or persistent entity. If the version identifier is compatible with the SQL/MX release, NonStop SQL/MX successfully proceeds with the operation. If the version identifier is incompatible with the SQL/MX release, NonStop SQL/MX fails the operation and returns a versioning error indicating an incompatibility.

The use of version identifiers enables interoperability between compatible versions and prevents interoperability between incompatible versions. For more information, see:

- Version identification of SQL/MX software components on page 3-7
- SQL/MX system software version (MXV) on page 3-8
- SQL/MX database object versions on page 3-8
- SQL/MX query plan and module versions on page 3-10

Version identification of SQL/MX software components

All version identification in SQL/MX is numeric. General availability (GA) and Controlled Availability (CA) software releases are assigned version numbers that are multiples of 20. General and controlled availability releases do not interoperate with Early Adopter Program (EAP) releases or other releases whose version numbers are not multiples of 20. Table 3-3 lists the software version identifiers for SQL/MX Release 3.1.

Table 3-3. Software version identifiers for SQL/MX Release 3.1

Version Item	SQL/MX release version identifier 3.1	For more information, see...
Current MXV	3100	SQL/MX system software version (MXV) on page 3-8
Earliest supported MXV	3100	
Current schema version and OSV	3100	SQL/MX database object versions on page 3-8
Earliest supported schema version	1200	
Default compiler version	3100	SQL/MX query plan and module versions on page 3-10
Current plan version	3100	
Earliest supported plan version	3000	

SQL/MX system software version (MXV)

The MXV provides a primary version identifier for the SQL/MX software installed on a node and provides a lower boundary, the earliest supported MXV, for all other versions that can be accepted by the node. All version numbers that are equal to or higher than the earliest supported MXV are compatible. All version numbers lower than the earliest supported MXV are incompatible.

The SQL/MX system software obtains all the earliest supported versions that correspond to the MXV. NonStop SQL/MX uses the earliest supported MXV to determine if a remote node is running a compatible earlier release of NonStop SQL/MX.

NonStop SQL/MX uses the earliest supported plan version to determine if a SQL/MX query execution plan or module from an earlier release of NonStop SQL/MX is compatible with the current version of NonStop SQL/MX. For more information, see SQL/MX query plan and module versions on page 3-10.

NonStop SQL/MX can also obtain the Object Schema Versions (OSVs) of SQL/MX database objects on which the system software operates. NonStop SQL/MX uses the OSVs to perform version checking. For example, when an object is opened, the SQL/MX executor compares the OSV for that object with the earliest supported schema version. If the object's OSV is earlier than the earliest supported schema version, a versioning error is generated. For more information, see SQL/MX database object versions on page 3-8.

SQL/MX database object versions

Database object versioning is a set of mechanisms that allow SQL/MX software to determine if the version of a SQL/MX database object is compatible with the MXV. Database object versions include the schema version, object schema version (OSV), and object feature version (OFV).

Schema version and Object Schema Version

A schema version is assigned to each user schema at creation time and is based on the version of the SQL/MX compiler that creates the schema. The Object Schema Version (OSV) of a SQL/MX database object is inherited from the schema where the object resides and determines if the object can be accessed by a particular version of NonStop SQL/MX.

NonStop SQL/MX uses the earliest supported schema version of the node to determine if it can access a SQL/MX database object in a schema that has a lower version. If the object's OSV is higher than or equal to the earliest supported schema version and lower than or equal to the MXV (which is often the same as the current schema version) of the node, NonStop SQL/MX accesses the object. Otherwise, NonStop SQL/MX returns a versioning error.

For SQL/MX Release 3.1, the schema version is 3100. For SQL/MX Release 3.0, the schema version is 3000. For SQL/MX Release 2.x, the schema version is 1200.

A database object can be accessed if all involved systems and schemas are version compatible.

Displaying the schema version

The following example displays the schema version of a specified schema, where `sch` is the schema name:

```
>>get version of schema sch;

VERSION: 3100

--- SQL operation complete.

>>
```

For more information, see SQL/MX Language Elements in the *SQL/MX Release 3.1 Reference Manual*.

Object Feature Version

The Object Feature Version (OFV) of a SQL/MX database object represents the lowest version schema that can accommodate the features used by that object. The OFV is computed from the actual features used by a database object and not from the version of the software that creates the database object.

Displaying the OSV and OFV

The following example displays the OSV and OFV of a specified database object, where the table name, *tlarge*, is a value that you specify:

```
>>get version of table tlarge;

OBJECT SCHEMA VERSION: 3100
OBJECT FEATURE VERSION: 3100

--- SQL operation complete.

>>
```

The following is an example of feature_version_info function:

```
>>select object_name, feature_version
from table (feature_version_info ('CATALOG', 'CATX', 1200));

OBJECT_NAME                                FEATURE_VERSION

-----------------------------------------  ---------------

CATX."schema x"."table with large key"     3100

CATX.SCHEMAY."table with bignum column"    3100
```

FEATURE_VERSION_INFO is a built-in table-valued function that returns feature version information for all user objects with an OFV higher than a given value, in a specified set of catalogs.

The MXCI SHOWLABEL command also displays OSV and the OFV.

SQL/MX query plan and module versions

Query plan versioning is a set of mechanisms that allow SQL/MX software to assign a version to query execution plans and modules and to determine if the version of a query execution plan or module is compatible with the MXV. Query plan versions include the plan version and module version.

Plan versions

The query execution plan of a SQL statement is assigned a plan version depending on the SQL/MX release that is installed on the node where the plan is compiled.

NonStop SQL/MX successfully compiles and executes a query execution plan if the plan version is higher than or equal to the earliest supported plan version and lower than or equal to the current plan version of the node where the plan is being executed. Otherwise, NonStop SQL/MX returns a versioning error.

Module versions

When you compile modules of embedded SQL/MX applications, the version assigned to a module is the same as the default compiler version of the node where you

compiled the module. The SQL/MX Release 3.1 compiler produces the module version 3100.

For a module to be executed on a node, the module version must be higher than or equal to the earliest supported plan version and lower than or equal to the current plan version. Otherwise, SQL/MX generates a versioning error.

Note. SQL/MX Release 3.1 supports v3100 and v3000 plans and modules. However, it does not support SQL/MX Release 2.3.x plans and modules. Before executing SQL/MX Release 2.3.x plans and modules on SQL/MX Release 3.1, you must recompile them.

Interoperability across schema versions

NonStop SQL/MX Release 3.1 is compatible with v1200 schemas, v3000 schemas, v3100 schemas, and the objects in such schemas on the local node and on remote compatible nodes in a network.

Limitations to schema version interoperability

- Schemas of different versions cannot coexist in the same catalog.
- Immediately after upgrading from SQL/MX Release 2.3.x or from SQL/MX Release 3.0 to SQL/MX Release 3.1, the version of the system schema and all user schemas is either 1200 or 3000. Executing the `UPGRADE ALL METADATA` command will upgrade all metadata for the system schema and the user schemas to v3100. After the metadata is upgraded to v3100, you cannot create v1200 schemas. However, using the `control query default` (`cqd`) command, you can create v3000 schemas in catalogs that do not already contain v3100 schemas:

  ```
  control query default create_definition_schema_version '3000'
  ```

- Database relationships cannot cross schema versions.

  For example, a view in a v1200 schema cannot reference a table in a v3100 schema.

  The following types of database objects are able to establish cross schema relationships:

  - views
  - triggers
  - RI constraints

  If you attempt to establish cross-schema relationships between schemas of different versions, the attempt will be rejected and an error is generated.

4

Upgrading to SQL/MX Release 3.1

This section addresses the following topics:

Planning for fallback

Before you install SQL/MX Release 3.1, plan for a fallback to an earlier version. To fall back to versions earlier than SQL/MX Release 2.3.4, you must first fall back to SQL/MX Release 2.3.4 and then to the required SQL/MX 2.x version.

To fall back to a previous version of SQL/MX, see Section 5, Falling Back from SQL/MX Release 3.1.

Note. When using new features, if you encounter an issue that is specific to SQL/MX Release 3.1, discontinue the use of those features instead of falling back from that version.

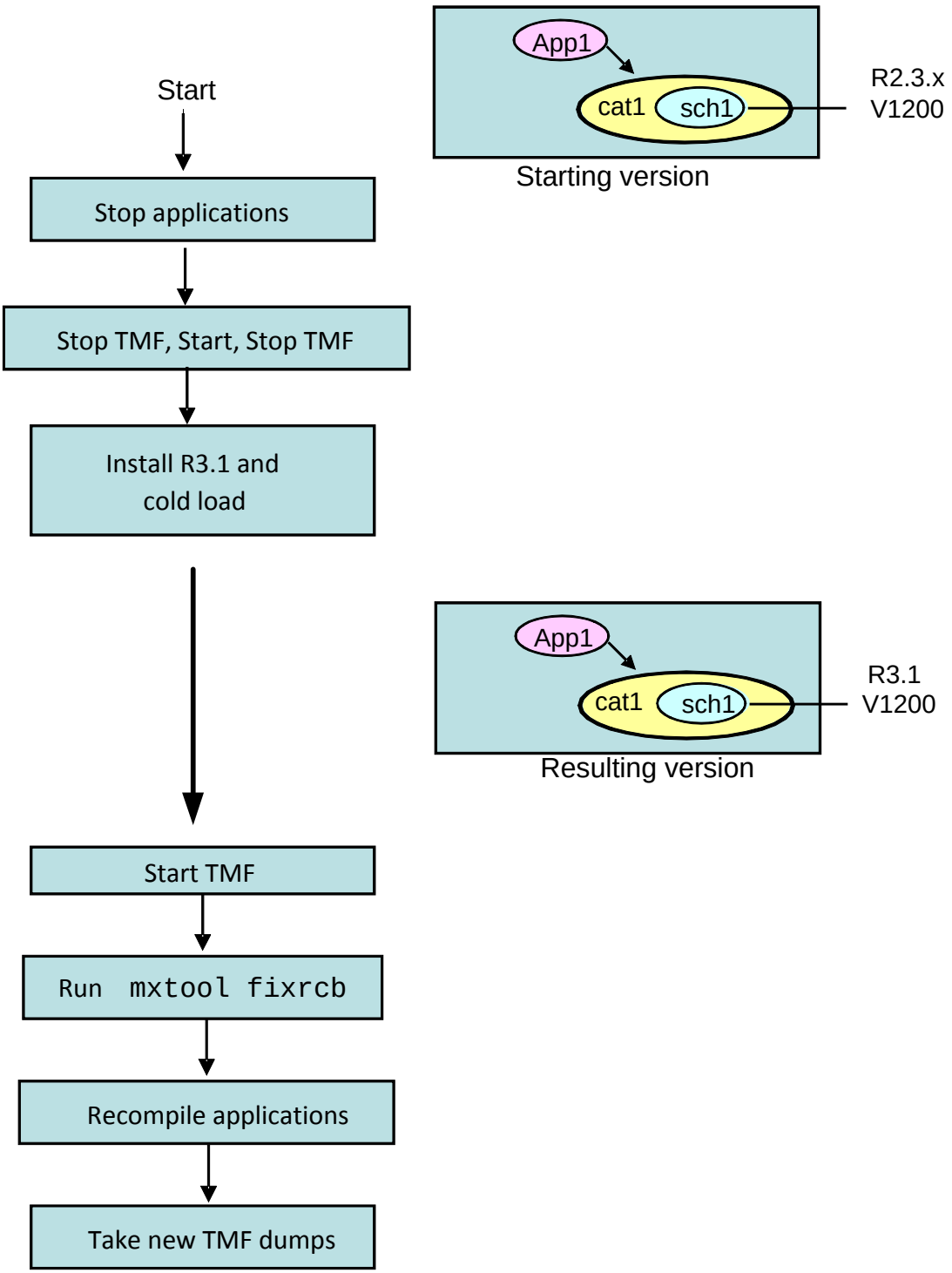
Upgrading from SQL/MX Release 2.3.x to SQL/MX Release 3.1

This section describes the sequence and the procedures to upgrade from SQL/MX Release 2.3.x to SQL/MX Release 3.1.

Upgrade sequence

To upgrade from SQL/MX Release 2.3.x to SQL/MX Release 3.1, you must follow the sequence shown in Figure 4-1.

Figure 4-1. Sequence to upgrade from SQL/MX 2.3.x to SQL/MX 3.1



Preinstallation

Complete the following prerequisites:

1. Stop all applications that access the database.
2. Stop the HP NonStop Transaction Monitoring Facility (TMF) twice with no database activity between the two stops:

```
TMF 1> STOP TMF
       <stop tmf output>
TMF 2> START TMF
       <start tmf output>
TMF 3> STOP TMF
       <stop tmf output>
```

To avoid volume recovery when TMF is started after the system upgrade, you must perform these steps. TMF recovery of SQL/MX database objects will access the RCBs of the affected objects. TMF shutdown must be done, because the RCBs of the affected objects are not immediately compatible after an upgrade to SQL/MX Release 3.1.

Installing SQL/MX Release 3.1

1. Install the Site Update Tape (SUT) for the required RVU. Determine the required RVU for a SQL/MX Release 3. See Table 3-1 on page 3-2. To install an RVU, see the *H06.nn* or *J06.nn Software Installation and Upgrade Guide*.
2. Install any necessary Independent Software Products, such as the NonStop Server for Java, the JDBC/MX Driver for NonStop SQL/MX, and RDF.
3. Install, configure, and then start TMF, which must be running on the system node where you install NonStop SQL/MX. For more information, see the *TMF Planning and Configuration Guide*.

Accessing the database

After the initial system load on the target release, RCBs are still incompatible. To enable regular access to database objects, you must execute an explicit RCB fixup by using the following command:

```
mxtool fixrcb
```

FIXRCB is an OSS command-line utility run from `mxtool`. It performs a RCB fixup for all the required database objects in catalogs that have an automatic reference on the local system. The local super ID must execute the `mxtool fixrcb` command. This command regenerates RCBs for SQL/MX objects in all catalogs that were originally created on the system.

If the system upgrade is performed for a clustered or networked environment, you must upgrade all affected systems simultaneously and run the `mxtool fixrcb` command on each involved system.

An error might occur while running the `mxtool fixrcb` command due to one of the following reasons:

- An involved node has an incompatible version.
- A user other than the local super ID performed the operation.

Note. The FIXRCB operation does not handle the objects in catalogs that have a manual reference on the local system. RCB fixup for objects in such catalogs must be performed on the system where the automatic reference is located.

The FIXRCB operation is available on systems running J06.12 or later J-series RVUs, or H06.23 or later H-series RVUs, and on fallback SPR (H06.21-ANC).

To verify the database recoverability after a system upgrade to SQL/MX Release 3.1, you must take a full set of online dumps after executing the `mxtool fixrcb` command successfully.

Note. Recovery on SQL/MX Release 3.1 by using pre-SQL/MX Release 3.1 TMF dumps has some restrictions because of the changes made by `fixrcb`. HP recommends that you take new TMF online dumps when the SQL database is ready (after running `mxtool fixrcb`).

Migrating the database after system upgrade

The following describes the SQL/MX Release 3.1 features and limitations related to migration of the database, which includes physical data files and all related objects.

- Databases created on SQL/MX Release 2.x (v1200 schemas) are supported.
- An upgrade of v1200 schemas to v3100 is supported.
- The v3100 schemas in new catalogs are supported without having to upgrade v1200 metadata from SQL/MX 2.x version.
- New SQL/MX Release 3.1 features are supported with v3100 metadata only. The features are not supported in v1200 and v3000 schemas. For information on how to use the new SQL/MX Release 3.1 features, see Upgrading the database and using new features on existing and new databases on page 4-8.
- New schemas in an existing, non-empty catalog will be of the same version as the existing schemas in the catalog. The version of a new schema in an empty catalog will be 3100, unless the `CREATE_DEFINITION_SCHEMA_VERSION CQD` is set to 1200.

Migrating applications after system upgrade

Embedded SQL-based programs, ODBC and JDBC-based programs that use Module File Caching (MFC) can create modules. The embedded modules that were generated on SQL/MX Release 2.3.4 or earlier are no longer compatible. Therefore, all such modules must be compiled after the system is upgraded to SQL/MX Release 3.1 and before the applications are started.

SQL/MX Release 3.1 enables upgrading of applications from an SQL/MX Release 2.3.x system to a SQL/MX Release 3.1 system without requiring you to recreate and reload the database. You do not need to preprocess, compile the host language, and link when you migrate from SQL/MX Release 2.3.x to SQL/MX Release 3.1. However, SQL/MX Release 3.1 does not support user modules and query plans from SQL/MX 2.3.x or earlier versions. You must to recompile all the existing embedded modules so they can be compatible with the enhanced SQL/MX Release 3.1 executor.

Note. Recompiling programs might result in different execution plans, causing a change in performance.

After installing SQL/MX Release 3.1, you must reformat the RCBs for all database objects and recompile all embedded modules before accessing the database. After reformatting the RCBs, you can retain the existing SQL/MX Release 2.3.x metadata (v1200) that is in use or upgrade all v1200 metadata to SQL/MX Release 3.1 (v3100).

Note. SQL/MX Release 2.3.x metadata (v1200) for existing applications and data objects in existing catalogs can coexist on a single system with SQL/MX Release 3.1 metadata (v3100) for objects in newly created catalogs. To use version 3100 metadata for existing objects, you must upgrade all existing v1200 metadata and recompile the embedded applications.

After a system fallback, before accessing the database, you must reformat the RCBs for all database objects. You might need to preprocess, compile, and link when you fall back from SQL/MX Release 3.1. The Fallback SPR is provided for SQL/MX Release 2.3.4 (ANC SPR on H06.21). To fall back to an earlier release of SQL/MX 2.x, you must perform the fallback in the following sequence:

1. Fall back to SQL/MX Release 2.3.4.
2. Fall back to the required SQL/MX 2.x version.

Upgrading from SQL/MX Release 3.0 to SQL/MX Release 3.1

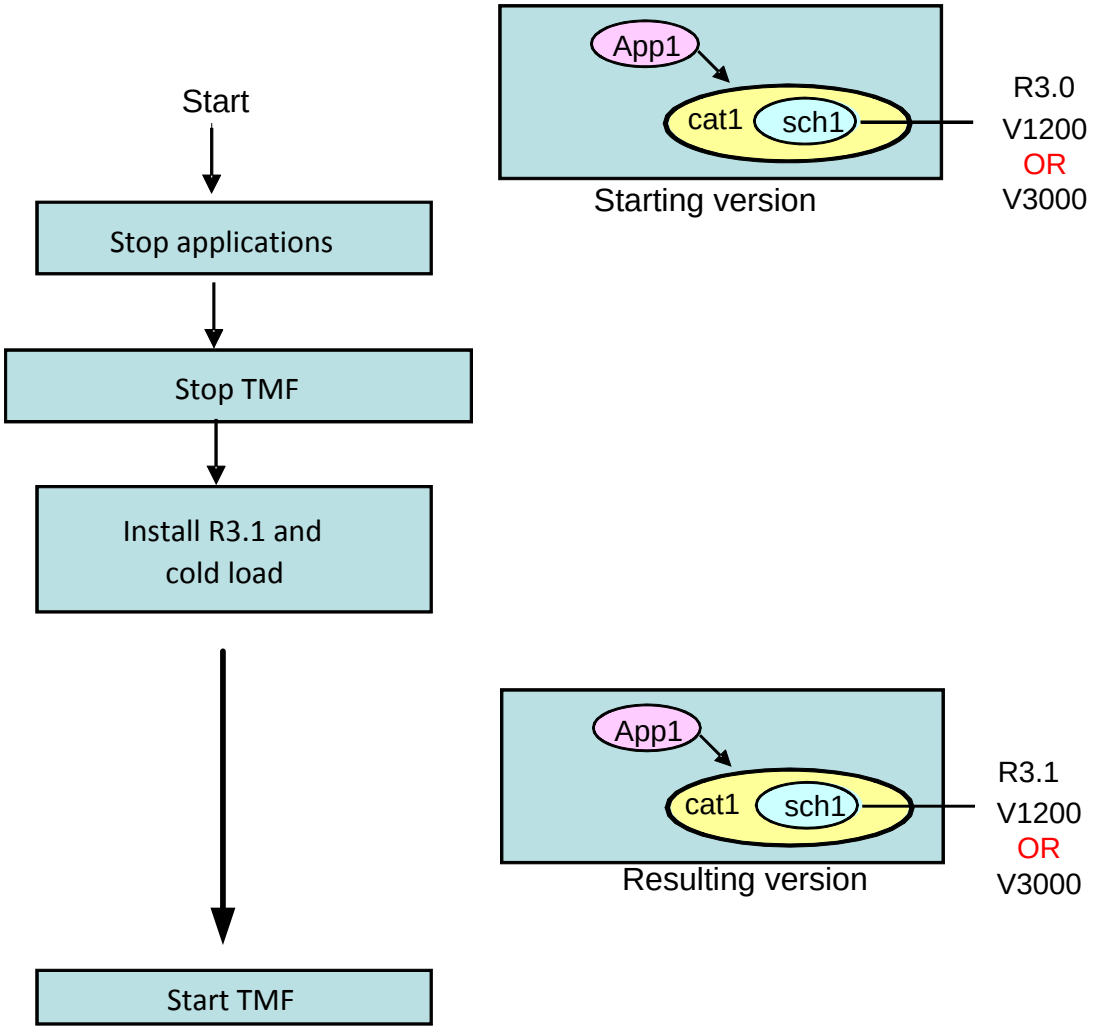
This section describes the sequence and the procedures to upgrade from SQL/MX Release 3.0 to SQL/MX Release 3.1.

Upgrade sequence

To upgrade from SQL/MX Release 3.0 to SQL/MX Release 3.1, you must follow the sequence shown in Figure 4-2.

Note. The steps to upgrade from SQL/MX Release 3.0 to SQL/MX Release 3.1 are simpler than the corresponding steps to upgrade from SQL/MX R2.3.x.

Figure 4-2. Sequence to upgrade from SQL/MX 3.0 to SQL/MX 3.1



Preinstallation

Before upgrading, complete the following steps:

1. Stop all applications that access the database.

2. Stop the HP NonStop Transaction Monitoring Facility (TMF).

Note. When upgrading from SQL/MX Release 3.0 to SQL/MX Release 3.1, you do not have to stop TMF twice before the installation.

Installing SQL/MX Release 3.1

For information on how to install SQL/MX Release 3.1, see Installing SQL/MX Release 3.1 on page 4-3.

Accessing the database

After the initial system load on the target release, the database is ready for use.

Note. When upgrading from SQL/MX Release 3.0 to SQL/MX Release 3.1, you do not have to use the `mxtool fixrcb` command, and recompile user application modules.

Migrating the database after system upgrade

The following describes the SQL/MX Release 3.1 features and limitations related to migration of the database, which includes physical data files and all related objects.

- Databases created on previous releases of SQL/MX (v1200 schemas, v3000 schemas) are supported.
- An upgrade of older version metadata (v1200 schemas, v3000 schemas) to v3100 is supported.
- v3100 schemas in new catalogs are supported without having to upgrade existing metadata.
- New SQL/MX Release 3.1 features are supported with v3100 metadata only. They are not supported in v1200 or v3000 schemas. For information on how to use the new SQL/MX Release 3.1 features, see Upgrading the database and using new features on existing and new databases on page 4-8.
- New schemas in an existing, non-empty catalog will be of the same version as the existing schemas in that catalog. The version of a new schema in an empty catalog will be 3100, unless the `CREATE_DEFINITION_SCHEMA_VERSION CQD` is set to 1200 or 3000.

Note. A system that is upgraded from SQL/MX Release 3.0 might have v1200 schemas - or a combination of v1200 and v3000 schemas - where that system previously was upgraded from SQL/MX Release 2.3.x to SQL/MX Release 3.0, and no metadata upgrade was performed.

Migrating applications after system upgrade

SQL/MX Release 3.1 supports Application Migration Without Recompilation (AMWR). After migrating applications from SQL/MX Release 3.0 to SQL/MX Release 3.1, you can:

- Execute the user modules and SQL queries compiled in SQL/MX Release 3.0 without any change.
- Use the RCBs generated on SQL/MX Release 3.0 without any change.
- Omit running the `mxtool fixrcb.` command.

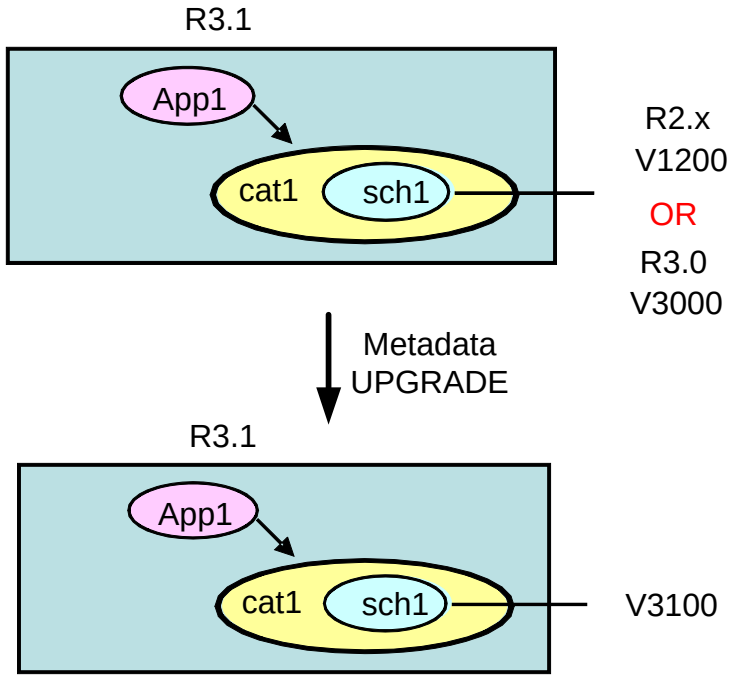
Note. An automatic recompilation of a SQL query with plan version 3000 will result in a query with plan version 3100.

Upgrading the database and using new features on existing and new databases

After upgrading to SQL/MX Release 3.1 using the relevant upgrade sequence, you can use the SQL/MX Release 3.1 features on existing and new databases. The scenarios in this section describe how you can use the new features.

- Scenario 1 - New features offered in SQL/MX Release 3.1 on existing database and applications: To use new features offered in SQL/MX Release 3.1 on any existing database and applications, you must upgrade all the metadata.

Figure 4-3. Using new features of SQL/MX Release 3.1 on existing databases

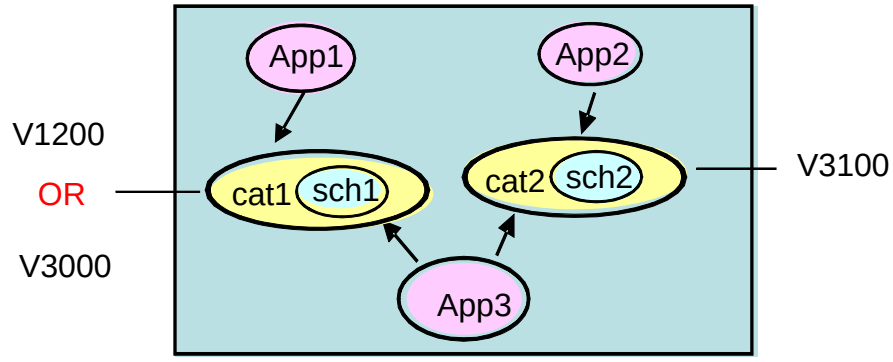


- Scenario 2 - New features offered in SQL/MX Release 3.1 on new databases and applications: After upgrading the system, newly created databases and applications can use all the SQL/MX Release 3.1 features. Additionally, the newly

created databases and applications can coexist with the databases and applications from the earlier release.

Note. Scenario 2 does not involve a metadata upgrade.

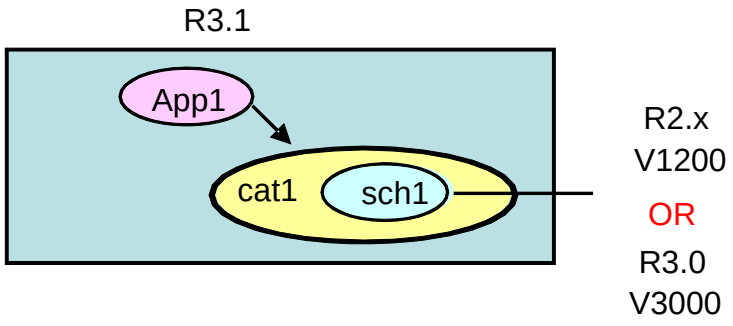
Figure 4-4. Coexistence of new and old databases



- Scenario 3 - New features offered by SQL/MX Release 3.1 are not required: If you do not want to use any of the new features offered by SQL/MX Release 3.1 on existing databases, no further changes are required after upgrading.

Note. Scenario 3 does not involve a metadata upgrade.

Figure 4-5. New features of SQL/MX Release 3.1 not required



Implementing the new functionality

If there is a chance of falling back to an earlier version, you must defer the implementation of the new functionality until you are sure that fallback is unlikely.

5 Falling Back from SQL/MX Release 3.1

This section describes falling back from SQL/MX Release 3.1 to SQL/MX Release 3.0 and to SQL/MX Release 2.x.

Note. When using new features, if you encounter an issue that is specific to SQL/MX Release 3.1, discontinue the use of those features instead of falling back from that version.

This section addresses these topics:

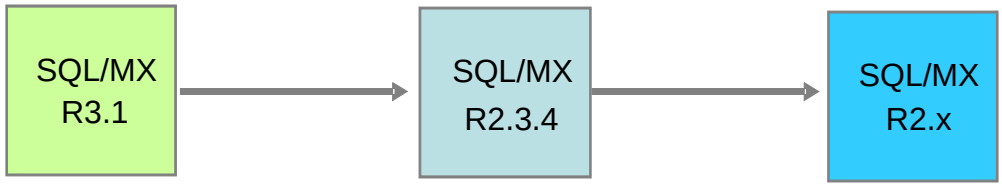
Fallback paths

You can fall back from SQL/MX Release 3.1 to SQL/MX Release 2.3.4 or SQL/MX Release 3.0, as shown in Figure 5-1.

Figure 5-1. Falling Back from SQL/MX 3.1 to SQL/MX 2.3.4 or SQL/MX 3.0

To fall back from SQL/MX Release 3.1 to a supported SQL/MX 2.x version earlier than SQL/MX Release 2.3.4, you must first fall back to SQL/MX Release 2.3.4 and then to the required version, as shown in Figure 5-2.

Figure 5-2. Falling back to a SQL/MX 2.x version earlier than SQL/MX 2.3.4

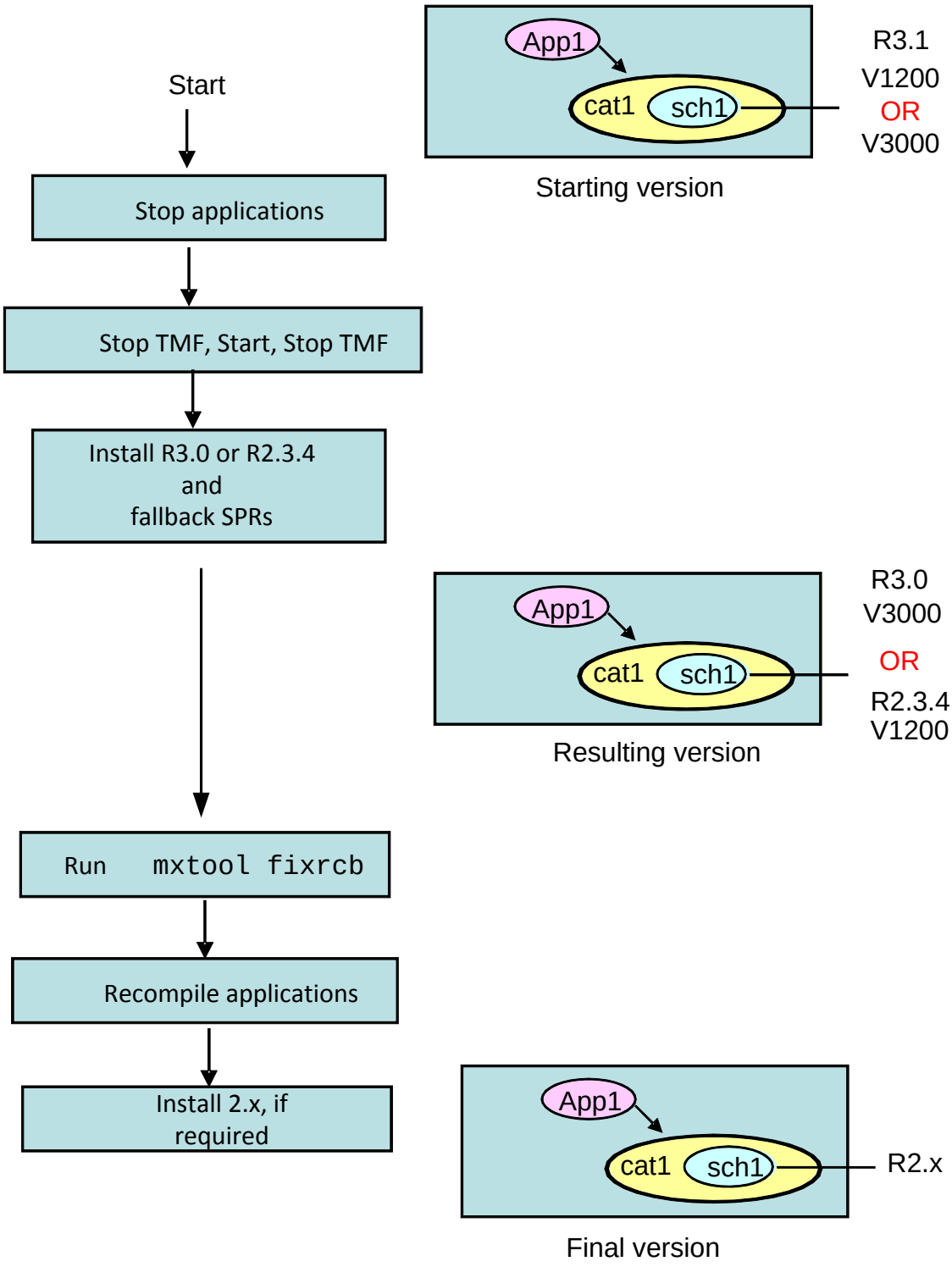


Fallback sequence

Based on your requirements, you must fall back in one of the following ways:

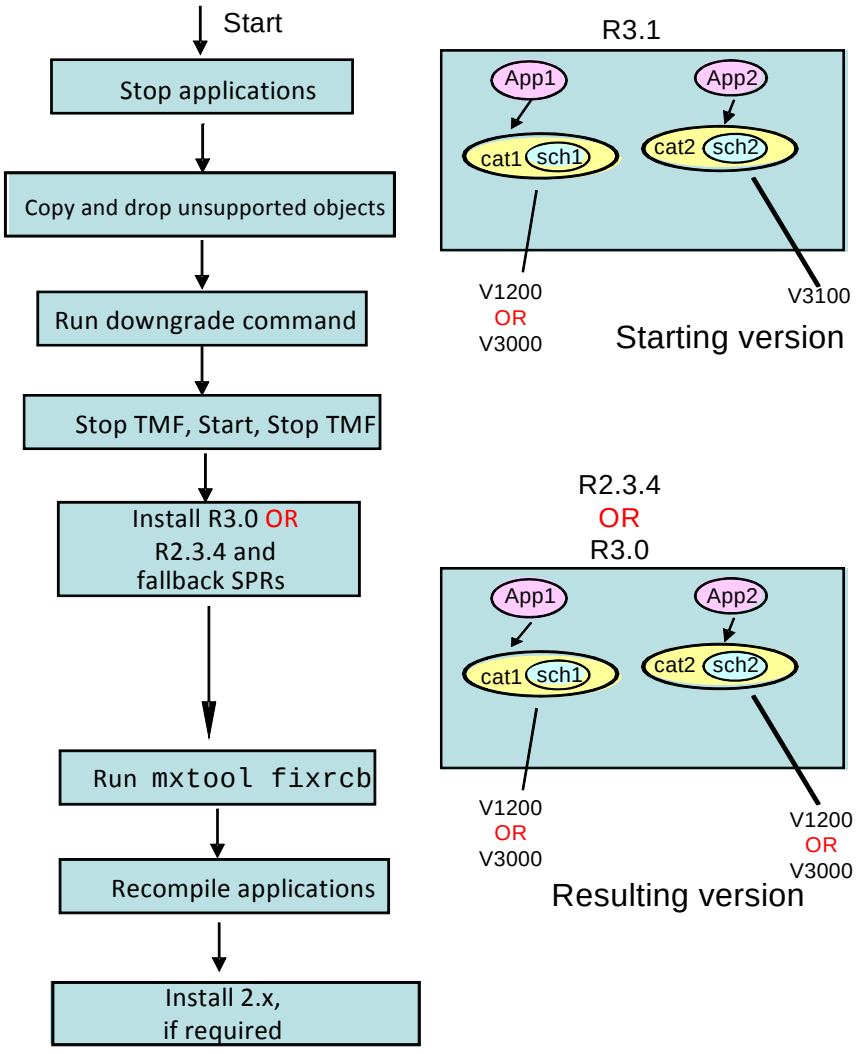
- If you have not upgraded the metadata to SQL/MX Release 3.1 and have not created any new databases, you must fall back as shown in Figure 5-3.

Figure 5-3. The v3000 or v1200 objects are not upgraded and new objects are not created on SQL/MX 3.1



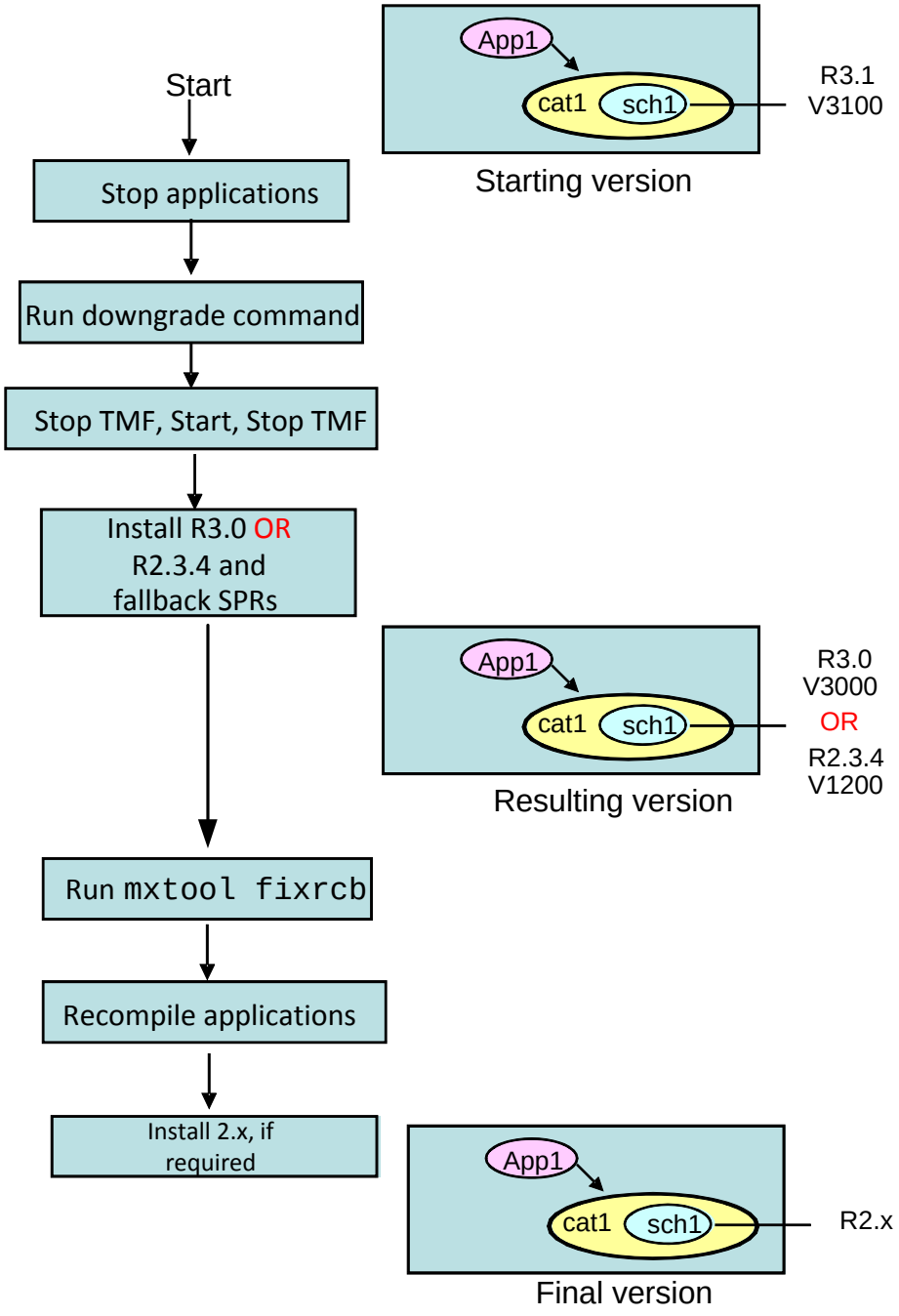
- If you have upgraded the metadata to v3100 and created objects using new features, you must fall back as shown in Figure 5-4.

Figure 5-4. New database objects are created on SQL/MX Release 3.1



- If you have upgraded your metadata from v1200 or v3000 to v3100, you must fall back as shown in Figure 5-5.

Figure 5-5. The v1200 or v3000 objects are upgraded to v3100



Fallback considerations

You must consider the following points before falling back:

- Applications that are compiled using the SQL/MX Release 3.1 compiler must be preprocessed, recompiled, and then linked with an earlier release of the SQL/MX compiler after the fallback.

Note. Recompiling programs might result in different execution plans, causing a change in performance.

- The online dumps taken on SQL/MX Release 3.1 cannot be used by TMF recovery on SQL/MX Release 3.0 or on SQL/MX Release 2.x. HP recommends that you take new full TMF online dumps immediately after fallback.
- If security administrators have granted any privileges on database objects after upgrading to SQL/MX Release 3.1, you should consider revoking those privileges before downgrading. Although the privileges granted by security administrators to database objects will apply in earlier SQL versions, attempts to modify those privileges through REVOKE on those objects will fail because the security administrator grants will be viewed as dependent privileges by earlier SQL versions. You can address this issue in one of the following ways:

  1. Prior to falling back to an earlier SQL version, all grants performed by security administrators must be revoked. For steps to identify and revoke these grants, see Removing Security Administrator Grants on page A-1. HP recommends this method unless you want to retain the existing privileges (for example, for operational reasons), in which case the following method should be considered.
  2. After falling back, the owner of each object affected by security administrator grants must GRANT, to the security administrator username (for example, “SECADMIN.USER1”), the same privileges, WITH GRANT OPTION, that the security administrator had granted while acting as a Security Administrator (GRANT ALL ...WITH GRANT OPTION can be used). Alternatively, the super ID can be used to perform the GRANT instead of the object owner. This action connects the "orphaned" security administrator grants to the owner. The reconnected grants can then either be left in place, or removed by the owner (or the super ID) using the REVOKE statement.

- Databases created after upgrading to SQL/MX Release 3.1 can include an IDENTITY column. You cannot remove an IDENTITY column from a table. You must remove all tables with IDENTITY columns before downgrading to an earlier version. You can save the data in tables with a similar layout, but without the IDENTITY column, and retrieve after a downgrade. However, using the saved tables instead of those with IDENTITY columns will require changes to the applications. For steps to identify tables that contain an IDENTITY column, see Identifying Tables That Contain an IDENTITY Column on page B-1.

Preinstallation

You must complete the following steps before installing SQL/MX Release 3.0 or SQL/MX Release 2.x:

Migrating the Database for a System Fallback

You might have to use the DOWNGRADE utility when you perform a system fallback from SQL/MX Release 3.1 to SQL/MX Release 3.0 and to SQL/MX Release 2.3.4.

Consider the following scenarios for system fallback:

- Executing the DOWNGRADE command is not necessary in the following scenarios:
  - The fallback version is SQL/MX Release 3.0 and the database contains v3000 and v1200 schemas only.
  - The fallback version is SQL/MX Release 2.3.x and the database contains v1200 schemas only.
- If you are using new database features for some objects in v3100 schemas, you must perform the following steps:
  - Use the FEATURE_VERSION_INFO function to obtain the name and object type of database objects that use new features. Such objects prevent a downgrade operation. For more information, see SQL/MX Functions and Expressions in the *SQL/MX Reference Manual*.
  - Convert the old unsupported objects to the corresponding copy objects without new features.
  - Drop the old unsupported objects before executing the DOWNGRADE command.
  - Downgrade to the required version. For information on how to use the downgrade utility, see SQL/MX Utilities in the *SQL/MX Release 3.1 Reference Manual*.

Stopping TMF

Before a system fallback, stop TMF twice with no database activity between the two stops:

```
TMF 1> STOP TMF
       <stop tmf output>
TMF 2> START TMF
       <start tmf output>
TMF 3> STOP TMF
       <stop tmf output>
```

These steps are necessary to avoid volume recovery when TMF is started after the system fallback. TMF recovery of SQL/MX database objects will access the RCBs of the affected objects. You must shut down TMF because the RCBs of the affected

objects are not immediately compatible after a change from SQL/MX Release 3.1 to an earlier release.

Note. Do not fall back from SQL/MX Release 3.1 immediately after a system crash. In case of a system crash, you must restart the system with the version that was running at the time of the crash, start TMF, and then shut down TMF twice before falling back.

Installing SQL/MX Release 3.0

To fall back to SQL/MX Release 3.0, complete the following steps:

1. Install SQL/MX Release 3.0. For information on installing SQL/MX Release 3.0, see Installing NonStop SQL/MX in the *SQL/MX Installation and Management Guide*.
2. Install the fallback SPRs referred to in the SQL/MX Release 3.1 softdoc.
3. Perform the required changes to your system after falling back. See Changes required after falling back on page 5-9.
4. Perform the operations described in Accessing the database on page 5-8.
5. Perform the operations described in Migrating applications after system fallback on page 5-9.

Installing SQL/MX Release 2.3.4

To fall back to SQL/MX Release 2.3.4, complete the following steps:

1. Install SQL/MX Release 2.3.4. For information on installing SQL/MX Release 2.3.4, see Installing NonStop SQL/MX in the *SQL/MX Installation and Management Guide*.
2. Install the fallback SPRs referred to in the SQL/MX Release 3.1 softdoc.
3. Perform the required changes to your system after falling back. See Changes required after falling back on page 5-9.
4. Perform the operations described in Accessing the database.
5. Perform the operations described in Migrating applications after system fallback on page 5-9.

Accessing the database

After the initial system load on the target fallback release, RCBs are still incompatible. Shutting down TMF twice prior to the fallback simply prevents TMF volume recovery from encountering those RCBs after the fallback. To enable regular access to database objects, run the `mxtool fixrcb` command before any other SQL activity.

This command regenerates RCBs for SQL/MX objects in all catalogs that were originally created on the system.

If the system fallback is performed for a clustered or networked environment, you must fall back all affected systems simultaneously and must run the `mxtool fixrcb` command on each involved system.

To verify the database recoverability after a system fallback, you must take a full set of online dumps after executing the `mxtool fixrcb` command successfully. For more information on the `mxtool fixrcb` command, see MXCI Commands in the *SQL/MX Reference Manual*.

Installing SQL/MX Release 2.x

To install SQL/MX Release 2.x, complete the following steps:

1. Install SQL/MX 2.x. For information on installing SQL/MX Release 2.x, see Installing NonStop SQL/MX in the *SQL/MX Installation and Management Guide*.
2. Perform the operations described in Migrating applications after system fallback.

Migrating applications after system fallback

The modules that were generated on SQL/MX Release 3.1 are no longer compatible after a system fallback to an earlier release. Therefore, all such modules must be recompiled after system fallback.

An alternate method is to save a copy of the modules before system upgrade and then reinstate that copy after fallback. This method is applicable only if the database layout after the fallback is similar to the layout before the system upgrade.

Changes required after falling back

Falling back might require changes to some or all of these corequisite product versions, particularly if you fall back to an earlier RVU:

- RDF (Independent Product)
- NonStop Server for Java (Independent Product)
- JDBC Driver for SQL/MX (Independent Product)
- DP2
- TMF
- Measure
- Backup and Restore 2

For more information, see the *H06.nn* or *J06.nn Release Version Update Compendium*.

A Removing Security Administrator Grants

In the following steps, queries create a command file containing SQL REVOKE statements targeting all grants that were performed by security administrators. You can then use the command file as input to mxci to remove the grants. To effectively use these queries, carefully read and follow the directions embedded within the comments as some variables must be modified prior to execution. These queries must be run against version 3100 metadata.

```
-- Execute all of the following from within mxci.
-- Set the pattern $$GRANTS_TABLE$$ and $$COMMANDS_TABLE$$ to two different
-- fully qualified names of non-existent tables in schemas in which the user
-- executing the queries may create tables. The catalog(s) and schema(s)
-- containing the tables must already exist.
-- Set pattern $$SYSTEM$$ to the name of the system on which the queries are
-- executing.
-- set pattern $$COMMANDS_FILE$$ to the name of an OSS file to which to write
-- the revoke commands.

set pattern $$GRANTS_TABLE$$ CAT.SCH.SA_GRANTS;
set pattern $$COMMANDS_TABLE$$ CAT.SCH.SA_COMMANDS;
set pattern $$SYSTEM$$ YOURSYS;
set pattern $$COMMANDS_FILE$$ obey.txt;

create table $$GRANTS_TABLE$$( table_uid largeint not null, grantor int,
grantee int, privilege_type char(2), column_number int, catalog_name char(
128) );

create table $$COMMANDS_TABLE$$( command varchar( 512 ) heading ' ' );

-- All of the following statements between the "--Begin Repeat" and
-- "--End Repeat" comments must be executed for every catalog on the system,
-- changing the following "set catalog"and "set param" statement arguments
-- to match the next catalog in sequence for each iteration.

-- Begin Repeat for every catalog on the system...

-- Set catalog to <catalog name>
-- Set param ?CATALOG_NAME to '<catalog name>'
set catalog YOURCAT;
set param ?CATALOG_NAME 'YOURCAT';

insert into $$GRANTS_TABLE$$
  select
tbl_privileges.table_uid,tbl_privileges.grantor,tbl_privileges.grantee,
            tbl_privileges.privilege_type, -1, ?CATALOG_NAME
  from  definition_schema_version_3100.tbl_privileges as tbl_privileges
  where tbl_privileges.grantor != -2
            and not exists
 ( select
     tbl_privileges2.table_uid,tbl_privileges2.grantor,
     tbl_privileges2.grantee,tbl_privileges2.privilege_type
from
     definition_schema_version_3100.tbl_privileges as tbl_privileges2
 where
   (tbl_privileges.grantor = tbl_privileges2.grantee
     or
   tbl_privileges2.grantee = -1
   )
     and
   tbl_privileges2.table_uid = tbl_privileges.table_uid
     and
   tbl_privileges.privilege_type = tbl_privileges2.privilege_type
```

```
     and
   tbl_privileges2.is_grantable = 'Y'
);
insert into $$GRANTS_TABLE$$
  select
col_privileges.table_uid,col_privileges.grantor,col_privileges.grantee,
    col_privileges.privilege_type, col_privileges.column_number,
?CATALOG_NAME
  from definition_schema_version_3100.col_privileges as col_privileges
  where col_privileges.grantor != -2
    and not exists
  ( select
col_privileges2.table_uid,col_privileges2.grantor,col_privileges2.grantee,
    col_privileges2.privilege_type,col_privileges.column_number
    from  definition_schema_version_3100.col_privileges as col_privileges2
    where
     (col_privileges.grantor = col_privileges2.grantee
       or
      col_privileges2.grantee = -1
      )
        and
      col_privileges2.table_uid = col_privileges.table_uid
        and
      col_privileges.privilege_type = col_privileges2.privilege_type
        and
     col_privileges.column_number = col_privileges2.column_number
        and
     col_privileges2.is_grantable = 'Y'
);

insert into $$COMMANDS_TABLE$$
select distinct
   'REVOKE ' ||
   case privilege_type
      when 'D' then 'DELETE'
      when 'E' then 'EXECUTE'
      when 'I' then 'INSERT'
      when 'R' then 'REFERENCE'
      when 'S' then 'SELECT'
      else 'UPDATE'
   end ||
   case grants.column_number
      when -1 then ''
      else '(' || trim( cols.column_name ) || ')'
   end ||
   ' ON ' ||
   trim( catsys.cat_name ) ||
   '.' ||
   trim( schemata.schema_name ) ||
   '.' ||
   trim( objects.object_name ) ||
   ' FROM "' ||
   user( grantee ) ||
   '";'
from $$GRANTS_TABLE$$ as grants,
     nonstop_sqlmx_$$SYSTEM$$.system_schema.schemata as schemata,
     nonstop_sqlmx_$$SYSTEM$$.system_schema.catsys as catsys,
     definition_schema_version_3100.cols as cols,
     definition_schema_version_3100.objects as objects
where grants.catalog_name = ucase( ?CATALOG_NAME )
        and
      schemata.cat_uid = catsys.cat_uid
        and
      objects.schema_uid = schemata.schema_uid
        and
```

```
      objects.object_uid = grants.table_uid
        and
      cols.object_uid = grants.table_uid
and
      (grants.column_number = -1 OR cols.column_number =
grants.column_number);


-- End Repeat for every catalog on the system...

-- Execute the following three statements only after repeating the above loop
-- for every catalog in the system...

log $$COMMANDS_FILE$$ clear;
select * from $$COMMANDS_TABLE$$;
log;

-- Exit mxci and logon as a security administrator.
-- Reenter mxci and obey the output file ($$COMMANDS_FILE$$) created above to
-- revoke the security administrator grants.
-- For example (and assuming $$COMMANDS_FILE$$ was set to “obey.txt”):
-- mxci
-- Hewlett-Packard NonStop(TM) SQL/MX Conversational Interface 3.1
-- (c) Copyright 2003, 2004-2011 Hewlett-Packard Development Company, LP.
-- >>obey obey.txt;
```

B Identifying Tables That Contain an IDENTITY Column

The following steps use queries to extract names of tables with an IDENTITY column. These queries must be run against version 3100 metadata.

```
-- Execute all of the following from within mxci.
-- Set pattern $$SYSTEM$$ to the name of the system on which the queries are
-- executing.

set pattern $$SYSTEM$$ YOURSYS;

-- All of the following statements between the "--Begin Repeat" and
-- "--End Repeat" comments must be executed for every catalog on the system,
-- changing the following "set catalog" statement argument to match the next
-- catalog in sequence for each iteration.

-- Begin Repeat for every catalog on the system...

-- Set catalog to <catalog name>
set catalog YOURCAT;

Select
         obj.object_name,schm.schema_name
from definition_schema_version_3100.objects obj,
         definition_schema_version_3100.sg_usage sgu,
         nonstop_sqlmx_$$SYSTEM$$.system_schema.schemata schm
where obj.object_uid = sgu.using_object_obj_uid and obj.schema_uid =
schm.schema.uid;

-- End Repeat for every catalog on the system...
```

C Identifying tables where the ownership has changed

If the ownership of database objects is changed after upgrading to SQL/MX Release 3.1, you must revert the changes before fallback, because earlier releases do not support this feature. To revert the ownership, the schema owner must be an existing Guardian user.

To revert the changes prior to fallback, complete the following steps for each user catalog on the system:

1. To identify the list of objects in a catalog whose owners are different from their parent schema owners, run the following query:

```
SELECT
SCHEMA_OWNER,
OBJECT_OWNER,
CAST(TRIM(S.SCHEMA_NAME)||'.'||TRIM(O.OBJECT_NAME) AS
CHAR(45)) AS "OBJECT_NAME",
OBJECT_TYPE as TYPE
FROM
<CATNAME>.DEFINITION_SCHEMA_VERSION_3100.OBJECTS O,
NONSTOP_SQLMX_<NODENAME>.SYSTEM_SCHEMA.SCHEMATA S
WHERE
O.SCHEMA_UID = S.SCHEMA_UID AND
O.OBJECT_OWNER <> S.SCHEMA_OWNER;
```

   The following is a sample output:

```
SCHEMA_OWNER OBJECT_OWNER OBJECT_NAME TYPE
------------ ------------ ----------- ----

65535        44801        SCH.V1      VI
44801        44802        USER1.T1    BT

--- 2 row(s) selected.
```

2. Logon as the schema owner or security administrator or super ID (if Super ID is part of the Security Administrator Group (SAG) or if the SAG does not exist).
3. Based on the object type, run the corresponding GIVE commands on the objects to change their ownership to the schema owner. For example, if the schema and

object owners are as described in step 1, and the catalog name is 'CAT', complete the following steps:

1. Logon as SUPER.SUPER (whose Guardian ID is 65535), the schema owner for schema 'SCH'.
2. Run the following command for the view SCH.V1:

   ```
   GIVE VIEW CAT.SCH.V1 to "SUPER.SUPER";

   --- SQL operation complete.
   ```

3. Logon as SQL.USER1 (whose Guardian ID is 44801), the schema owner for schema 'USER1'.
4. Run the following command for table USER1.T1:

   ```
   GIVE TABLE CAT.USER1.T1 to "SQL.USER1";

   --- SQL operation complete.
   ```

Index

O

P

Q

R

S

T

U

V