Reference Guide

Integrated Library System

System Management

VIRTUA ILS – INTEGRATED LIBRARY SYSTEM

System Management

Reference Guide

Version 16.1

October 2017

1701 Kraft Drive
Blacksburg, Virginia 24060
U. S. A.
Phone 800.858.8857
E-mail: info@iii.com

Table of Contents

TABLE OF CONTENTS I

TABLE OF FIGURES VI

TABLE OF FIGURES VI

1. INTRODUCTION 1

1.1 ABOUT THE SYSTEM MANAGEMENT GUIDES 1
1.2 USING THIS GUIDE 2

2. DIRECTORY STRUCTURE OF THE VIRTUA SERVER INSTALLATION 3

2.1 VIRTUA DIRECTORY STRUCTURE: APPLICATIONS SOFTWARE 3
2.1.1 VIRTUA DIRECTORY DESCRIPTIONS - APPLICATIONS 4
2.2 VIRTUA DIRECTORY STRUCTURE: ORACLE DATABASE FILES 6
2.2.1 ABOUT ORACLE TABLESPACES AND DATA FILES 7
2.2.2 VIRTUA DIRECTORY DESCRIPTIONS - ORACLE 7

3. ADMINISTRATIVE TIPS FOR WORKING WITH UNIX® 12

3.1 ABOUT ENVIRONMENT VARIABLES 12
3.1.1 GETTING THE VALUE OF ENVIRONMENT VARIABLES 13
3.1.2 SETTING THE VALUE OF ENVIRONMENT VARIABLES 13
3.1.3 UNSETTING THE VALUE OF ENVIRONMENT VARIABLES 14
3.1.4 CREATING A SCRIPT TO SET ENVIRONMENT VARIABLES 14
3.2 RUNNING SCRIPTS AND EXECUTABLES 15
3.2.1 RUNNING SCRIPTS AND EXECUTABLES AS BACKGROUND PROCESSES 15
3.2.2 ADDING ORACLE ENVIRONMENT VARIABLE INFORMATION TO A SCRIPT 16
3.3 REDIRECTING OUTPUT 16
3.3.1 REDIRECTING MESSAGES FROM STDOUT 17
3.3.2 REDIRECTING MESSAGES FROM STDERR 17
3.3.3 REDIRECTING MESSAGES FROM STDOUT AND STDERR 18
3.4 DETECTING PROCESSES 18
3.5 KILLING PROCESSES 19
3.6 WORKING WITH CRON JOBS 20
3.6.1 VIEWING CRON JOBS 20
3.6.2 CREATING A CRON JOB 21
3.7 FINDING TEXT IN FILES 22

4. ADMINISTRATIVE TIPS FOR WORKING WITH ORACLE 24

4.1 WORKING WITH THE VIRTUA/ORACLE DATABASE 24
4.1.1 RECOMMENDED KNOWLEDGE FOR WORKING WITH THE DATABASE 25
4.1.2 MAKING CHANGES IN THE DATABASE 25
4.2 PERFORMING BASIC TASKS IN THE VIRTUA/ORACLE DATABASE 25
4.2.1 STARTING UP AND SHUTTING DOWN THE DATABASE 26
4.2.2 IDENTIFYING THE DATABASES THAT ARE RUNNING 27
4.2.3 IDENTIFYING THE ORACLE VERSION 27
4.2.4 IDENTIFYING ACTIVE DATABASE CONNECTIONS 28
4.2.5 VIEWING INFORMATION ABOUT TABLES 28
4.2.6 WORKING WITH THE ORACLE LISTENER 29
4.2.7 CHANGING PASSWORDS FOR ORACLE USERS 30
4.2.8 REFRESHING PERMISSIONS FOR VIRTUA ORACLE USERS 31
4.2.9 WORKING WITH TABLESPACES AND DATA FILES 32
4.2.10 GATHERING STATISTICS 33
4.2.11 CREATING A DIRECTORY OBJECT 34

5. WORKING WITH THE PSDRIVER.EXE PROGRAM 35

5.1 RUNNING PSDRIVER.EXE 35
5.1.1 RUNNING PSDRIVER.EXE DIRECTLY FROM THE COMMAND LINE 36
5.1.2 RUNNING PSDRIVER.EXE FROM A SCRIPT 36
5.2 LOGGING SERVER OUTPUT 37
5.2.1 SETTING THE LOG LEVEL 38
5.2.2 USING SYSLOG TO LOG MESSAGES 39
5.3 IDENTIFYING PSDRIVER.EXE PROCESSES 39
5.4 STOPPING A PSDRIVER.EXE PROCESS 40
5.5 WORKING WITH USER LIMIT CONSTRAINTS 41
5.5.1 VIEWING A COMPLETE LIST OF CONNECTIONS 41
5.6 WORKING WITH THE LICENSE KEY 43
5.6.1 DETERMINING THE VIRTUA RELEASE 43
5.6.2 VERIFYING THE LICENSE KEY AND EXPIRATION DATE 44
5.6.3 UPDATING THE LICENSE KEY 44

6. DATABASE BACKUPS 45

6.1 IMPORTANCE OF DATABASE BACKUPS 46
6.2 OPTIONS FOR DOING DATABASE BACKUPS 47
6.2.1 ARCHIVE LOGGING 47
6.2.2 COLD BACKUPS 51
6.2.3 HOT BACKUPS 52
6.3 VALIDATING BACKUP FILES 56
6.3.1 VALIDATING DATA FILES 56
6.4 MANAGING THE PHYSICAL MEDIA OF BACKUP FILES 58
6.5 INNOVATIVE RECOMMENDATIONS 59

6.5.1 USE ARCHIVE LOGGING 59
6.5.2 TAKE REGULAR BACKUPS 59
6.5.3 TEST YOUR BACKUP FILES 59
6.5.4 RETAIN PAST BACKUP FILES 60
6.5.5 STORE MULTIPLE COPIES OF BACKUP FILES 60
6.5.6 PRACTICE RECOVERING FROM BACKUP 60

7. EXPORTING AND IMPORTING A VIRTUA DATABASE USING DATA PUMP 61

7.1 EXPORTING THE DATABASE 61
7.2 IMPORTING THE DATABASE 62

8. COPYING PRODUCTION DATA TO THE TEST DATABASE 64

8.1 BEFORE YOU BEGIN . . . 64
8.2 CREATING A TEST DATABASE FROM THE PRODUCTION DATABASE 65
8.2.1 CREATING A BLANK TEST DATABASE 65
8.2.2 DETERMINING THE AMOUNT OF DISK SPACE NEEDED 66
8.2.3 COPYING DATA FROM THE PRODUCTION DATABASE TO THE TEST DATABASE 66
8.2.4 MAKING THE DATABASE COPY READY FOR USE 67

9. CHARACTER MAPPING 69

9.1 DETERMINING WHICH INPUT AND OUTPUT MAPS ARE LOADED 69
9.2 LOADING CHARACTER MAPS 70
9.3 CONVERTING THE CHARACTER SET OF A TEXT FILE 72

10. APPENDIX A - SCRIPTS AND EXECUTABLES 74

11. APPENDIX B - ENVIRONMENT VARIABLES USED BY VIRTUA 102

11.1 EXE_DIR 103
11.2 ORACLE_HOME 103
11.3 ORACLE_SID 104
11.4 NLS_LANG 104
11.5 PATH 104
11.6 VIRTUA_USER 105
11.7 VIRTUA_PASSWORD 105
11.8 V_LOG_LEVEL 105
11.9 V_LOG_TYPE 106
11.10 V_LOG_TYPE_PARAM 106
11.11 VTLS_TEMP 107
11.12 IIRS_SYS 107
11.13 EDIFACT_NEW_LINE 107

11.14 CLAIM_BATCH_LEVEL **108**
11.15 CLAIM_BATCH_TYPE **108**
11.16 ADD_AAP_LINKS_ONLY **109**
11.17 LC_TIME **109**
11.18 LC_ALL (LOCALE ENVIRONMENT VARIABLE) **109**
11.19 KEYWORD_INDEXING_OFF **110**
11.20 HEADING_INDEXING_OFF **110**

12. APPENDIX C - MAPPING CODES **111**

12.1 SORT MAPS **111**
12.2 INPUT MAPS **112**
12.3 OUTPUT MAPS **113**

13. APPENDIX D - CHARACTER SORTING AND MAPPING **115**

13.1 INTRODUCTION TO HEXADECIMAL SORTING **115**
13.2 UNDERSTANDING HEXADECIMAL NUMBERS **116**
13.2.1 COUNTING 116
13.2.2 COMPARING 118
13.2.3 DETERMINING THE CHARACTER SORT ORDER USING HEXADECIMAL NUMBERS 119
13.3 DETERMINING THE SORT ORDER OF A CHARACTER **121**
13.4 USING SORT MAPS **122**
13.4.1 UNDERSTANDING SORT MAPS 124
13.4.2 LOADING SORT MAPS 131
13.4.3 AFTER RUNNING LOADSORTMAP.SH 132

14. APPENDIX E - TROUBLESHOOTING SCRIPTS **133**

14.1 WORKING WITH REDO LOG GROUPS **133**
14.1.1 SWITCHING THE CURRENT REDO LOG GROUP 133
14.1.2 DISPLAYING THE REDO LOG SWITCHES 133
14.2 REPORTING ON PERFORMANCE PROBLEMS **134**
14.3 CHECKING ORACLE FOR BLOCK CORRUPTION **134**

15. APPENDIX F - CHANGES IN THIS GUIDE **135**

15.1 CHANGES FOR VERSION 16.1 **135**

INDEX **136**

INDEX **136**

Table of Figures

Figure 5-1. Virtua Client - Licensed User Limit Reached 41
Figure 13-1 Counting in a Base-10 System 117
Figure 13-2. Counting in a Base-16 System 118
Figure 13-3. Sort Map - Polish.mu 123
Figure 13-4. Unicode Code Chart - Hebrew Character Set 126
Figure 13-5. List of Code Charts at Unicode.org 128
Figure 13-6. Unicode Code Chart 129
Figure 13-7. Sort Map 130
Figure 13-8. Unicode Code Chart - Basic Latin 131

1. Introduction

The *System Management Reference Guide* provides information about system administration tasks for the Virtua™ ILS – Integrated Library System. Information in this guide is intended for system administrators who feel comfortable performing tasks that directly modify the contents of the Virtua database.

Most of the tasks documented in this guide alter system-required data or files. If you are not sure whether you need to perform a task documented in this guide, contact an Innovative customer services representative. Additionally, if you do not feel comfortable performing a task, or if you are unaware of the consequences of performing a task, contact Innovative.

This chapter covers the following topics:

⇒ About the System Management Guides
⇒ Using this Guide

1.1 About the System Management Guides

In addition to this reference guide, there are five subsystem-specific System Management user's guides:

- *Virtua System Management: Acquisitions and Serials User's Guide*
- *Virtua System Management: Cataloging User's Guide*
- *Virtua System Management: Circulation User's Guide*
- *Virtua System Management: OPAC User's Guide*
- *Virtua System Management: Reporting User's Guide*

Each of these guides contains instructions for running scripts and executables for the associated subsystem. To determine which guide documents a desired script or executable, refer to the table in the appendix "Documented Scripts and Executables" in this document.

1.2 Using this Guide

The *Virtua System Management Reference Guide* contains information related to Virtua system administration tasks. Use the list below and the table of contents to locate specific information in this guide.

For . . .	**See . . .**
An overview of the Virtua directory structure	Chapter 2
Administrative tips for working with UNIX	Chapter 3
Administrative tips for working with Oracle	Chapter 4
Instructions for working with the program **psdriver.exe**	Chapter 5
Information about performing database backups	Chapter 6
Instructions for using Data Pump to export and import a database or schema	Chapter 7
Information about copying production data to the test database	Chapter 8
Information about Character Mapping	Chapter 9
A list of scripts and executables discussed in the Virtua user documentation	Appendix A
Important information about the environment variables used by Virtua	Appendix B
A list of input, output, and sort maps used by Virtua	Appendix C
An overview of character sorting and mapping	Appendix D
Scripts used for troubleshooting	Appendix E
A list of changes that were made in the guide since the last version	Appendix F

2. Directory Structure of the Virtua Server Installation

This chapter outlines the default directory structure of the Virtua installation on the server. The Virtua system is separated into two distinct directory structures: 1) Oracle database files, which are located under the **/usr/vtls/clasXX** directory and 2) Applications software, which is located under **/usr/vtls/virtua**. If your server has a non-standard installation of Virtua, your directory structure may differ. If you make changes to the default directory structure, Innovative cannot guarantee that your system will function properly.

This chapter covers the following topics:

⇒ Virtua Directory Structure: Oracle Database Files
⇒ Virtua Database Directory Structure

2.1 Virtua Directory Structure: Applications Software

Here is the structure of the default installation of the applications software for the Virtua database. A few of these directories are discussed in detail below

/usr/vtls/virtua	- Base directory
/ora11	- Oracle server home
/Apache/Apache/htdocs	- Document root of the Apache Web Server installation.
/r_xx_x	- Main directory of the Virtua server software.
/sql	- Directory that contains Virtua SQL scripts.
/src	- Directory that contains Virtua scripts and executables.
/lang	- Directory that contains files related to character mapping.
/webrpt_xx_x	- Main directory for InfoStation.
/perl	- Main directory for the Perl 11 installation.

/perl11	- Main directory for the Perl 11 installation.
/yaz	- External library for Perl modules.
/expat	- External library for Perl modules.
/marcConversion	- Directory that contains tools for converting MARC 21 to UNIMARC and vice versa.
/mutt	- Directory that contains the e-mail client.
/xpdf	- Directory that contains tools for converting PDF files to UTF-8.
/antiword	- Directory that contains tools for converting Word files to UTF-8.

Note:

- For information about the directory structure of the Chamo installation, see the *Chamo Administration Guide*.
- For information about the directory structure of the InfoStation installation, see the *InfoStation User's Guide*.

2.1.1 Virtua Directory Descriptions - Applications

The sql, src, and lang directories contain files that are likely to be of interest to most users. These directories are discussed briefly below.

2.1.1.1 sql

The **sql** directory of your Virtua installation contains SQL scripts that are used to access, and, in some cases, modify the data in the database. Some of these scripts are designed to be run from the SQL*Plus prompt, while others are called by other scripts or executables, which are documented in the appropriate System Management guide. Most of the scripts in this directory make significant changes to the data in your database.

Warning: You may run, modify, or delete any script in this directory ONLY if your institution has its *own* Oracle license (as opposed to an embedded Oracle license), in which case, your institution is authorized to use Oracle utilities directly to perform queries and other commands on your database.

2.1.1.2 src

The **src** directory of your Virtua installation contains scripts and executables that are used in the Virtua system. Some of these programs are designed to be run from the command line, while others are called by other scripts or executables. Many of these programs make significant changes to the system.

Warning: Do NOT run, modify, or delete any program in this directory unless you are explicitly instructed to do so by Innovative staff or documentation.

Most scripts and executables that are available for customer use are documented in a System Management guide. For a list of available scripts and executables, see Appendix A: "Scripts and Executables" in this document.

2.1.1.3 lang

The **lang** directory of your Virtua installation contains files that are used for character mapping functions. The following subdirectories are included in this directory:

- **output** - Contains the output character maps used by Virtua.
- **input** - Contains the input character maps used by Virtua.
- **sort** - Contains the sort maps used by Virtua.

For information about working with character maps, see the chapter "Character Mapping" in this document.

2.2 Virtua Directory Structure: Oracle Database Files

Here is the structure of the default installation of Oracle database files for the Virtua server software. In the following sections, we provide more detailed descriptions of these directories

/usr/vtls/clas01

/initvtls.ora	Initialization parameters for Oracle
/lob	Large (binary) object data files
/data	Data files for tables
/indx	Data files for indexes
/arch	Archive redo log files
/migrate	Upgrade scripts and log files
/rdo1	First set of redo log files
/rdo2	Second set of redo log files
/system	Administrative data files (system, temp, users)
/intr	InterMedia (keyword index) data files
/undo	Files to undo changes to the database
/ctl01	Control files
/ctl02	Control files (copy of ctl01, should be stored on different physical disk)
/ctl03	Control files (copy of ctl01, should be stored on different physical disk from ctl01 and ctl02)

2.2.1 About Oracle Tablespaces and Data Files

A tablespace is a container for segments, or database objects, such as tables and indexes. A database consists of one or more tablespaces, each made up of one or more data files, which are files that are part of an Oracle database. Data files are used to store data, including user data and undo data, and are grouped together into tablespaces. Tables and indexes are created within a particular tablespace.

When a new Virtua database is created, it will have 11 to 13 tablespaces by default (depending on your system, the Large Tables tablespace and Large Indexes tablespace may or may not be included). Virtua is moving toward a ONE tablespace design. Effective as of Virtua release 2010.2 and Oracle 11*g*, all *new* tables and indexes are added to a single tablespace called Small Tables.

If you run the script **CheckTablespaceFileSize.sh** (see section "Checking Tablespaces and Data File Sizes"), you will be able to see the list of tablespaces in the database, along with the data files that each contains and their sizes. For your reference, here are the tablespaces that are created:

SYSTEM
USERS
INTERMEDIA
LOB_DATA
MEDIUM_INDEXES
MEDIUM_TABLES
SMALL_INDEXES
SMALL_TABLES
SYSAUX
UNDO
TEMP

2.2.2 Virtua Directory Descriptions - Oracle

2.2.2.1 initvtls.ora

Purpose: The **initvtls.ora** file contains initialization parameters for Oracle. These parameters are loaded each time Oracle is started.

Warning: Do NOT add, delete, or modify parameters in this file unless you are certain of what you are doing. Some parameters in this file are crucial to the operation of the database. If you change them to an incorrect value, the consequences can be as severe as the corruption of all of the data in the database.

2.2.2.2 lob

File Type: Data files

Purpose: The **lob** directory contains data files that store records in Binary Large Object (BLOB) format. In the Virtua system, these files are generally used to store MARC data.

2.2.2.3 data

File Type: Data files

Purpose: The **data** directory contains general data files used in the Virtua system. This directory contains the following subdirectories:

- **sm1** - Contains the small tablespaces data files.
- **md1** - Contains the medium tablespaces data files.
- **lg1** - Contains the large tablespaces data files.

Depending on your configuration, it is possible that you will not have medium and/or large tablespaces.

2.2.2.4 indx

File Type: Index files

Purpose: The **indx** directory contains files that store indexes used in Virtua. This directory contains the following subdirectories:

- **sm1** - Contains the small indexes data files.
- **md1** - Contains the medium indexes data files.
- **lg1** - Contains the large indexes data files.

Depending on your configuration, it is possible that you will not have medium and/or large indexes.

2.2.2.5 arch

File Type: Archive log files

Purpose: The **arch** directory is available for archive logs. However, it is not required that you configure Oracle to write archive logs to this directory. But wherever you configure Oracle to write archive logs, it should be on a separate disk from the data files. For information about archive logging, see the chapter "Database Backup" in this document.

2.2.2.6 migrate

File Type: N/A

Purpose: The **migrate** directory is used for database migration activity.

2.2.2.7 rdo1

File Type: Redo log files

Purpose: The **rdo1** directory is where Oracle will write redo logs in a default installation of Virtua. The information that is written to this directory is duplicated (duplexed) in the directory **rdo2**. This directory should exist on a separate physical disk from the data files and from the **rdo2** directory.

2.2.2.8 rdo2

File Type: Redo log files

Purpose: The **rdo2** directory is where Oracle will write redo logs in a default installation of Virtua. The information that is written to this directory is duplicated (duplexed) in the directory **rdo1**. This directory should exist on a separate physical disk from the data files and from the **rdo1** directory.

2.2.2.9 system

File Type: System files

Purpose: The **system** directory contains files that are required by Oracle to process basic functions. These files are . . :

- **sys_01.dbf** - System tablespace file.
- **temp_01.dbf** - Temporary tablespace file.
- **users_01.dbf** - Users tablespace file.

2.2.2.10 intr

File Type: InterMedia files

Purpose: The **intr** directory stores InterMedia data.

2.2.2.11 undo

File Type: Data files

Purpose: The **undo** directory stores the database's undo logs.

2.2.2.12 ctl01

File Type: Control file

Purpose: The **ctl01** directory stores the control file. The information that is written to this directory is duplicated (multiplexed) in the directories **ctl02** and **ctl03**. This directory should exist on a separate physical disk from the **ctl02** and **ctl03** directories.

2.2.2.13 ctl02

File Type: Control file

Purpose: The **ctl02** directory stores the control file. The information that is written to this directory is duplicated (multiplexed) in the directories **ctl01** and **ctl03**. This directory should exist on a separate physical disk from the **ctl01** and **ctl03** directories.

2.2.2.14 ctl03

File Type: Control file

Purpose: The **ctl03** directory stores the control file. The information that is written to this directory is duplicated (multiplexed) in the directories **ctl01** and **ctl02**. This directory should exist on a separate physical disk from the **ctl01** and **ctl02** directories.

3. Administrative Tips for Working with UNIX®

This chapter provides basic tips for performing system administration tasks on your server. The instructions in this chapter assume that you have experience working with your server from the command line. Tasks that you should be able to complete before you continue with this chapter include but are not limited to…

- Navigating the file system.
- Copying and moving files.
- Modifying text files.
- Setting environment variables
- Running scripts and executables.

This chapter contains the following topics:

⇒ About Environment Variables
⇒ Running Scripts and Executables as Background Processes
⇒ Redirecting Output
⇒ Detecting Processes
⇒ Killing Processes
⇒ Working with Cron Jobs
⇒ Finding Text in Files

3.1 About Environment Variables

Environment variables specify information about the working environment in the current UNIX session. These variables are accessed by programs such as Virtua for use when executing functions. Additionally, some of these variables are available from the command line.

Many environment variables are set automatically when you begin a new UNIX session, but you can change their values as needed.

Note: Environment variables need to be set for *each* database.

Important: It is imperative that you are aware of the variables that are set for your system. The environment variables EXEC_DIR, ORACLE_SID, and NLS_LANG are crucial to the proper operation of your system. If you set an environment variable incorrectly, you could inadvertently run the wrong program or even modify data on the wrong database.

For a list of environment variables used by Virtua, see the appendix "Environment Variables Used by Virtua" in this guide. We recommend that you review this list and check the values of these variables before performing tasks in Virtua.

3.1.1 Getting the Value of Environment Variables

You can get the value of any environment variable by using the following command:

```
echo $[variable name]
```

Where **[variable name]** is the name of the environment variable. For example, to get the value of the environment variable EXE_DIR, you would type:

```
echo $EXE_DIR
```

The system displays the value of the variable. If the system returns no information, the environment variable you specified is not set.

3.1.2 Setting the Value of Environment Variables

You can set the value of any environment variable by using the following command:

```
export [variable name]=[value]
```

Where . . .

- **[variable name]** is the name of the environment variable.
- **[value]** is the value to which you want to assign the environment variable.

For example, to set the value of the environment variable ORACLE_SID to vtls01, you would type:

```
export ORACLE_SID=vtls01
```

This variable will remain set until you change it or end your session.

3.1.3 Unsetting the Value of Environment Variables

You can unset the value of any environment variable by using the following command:

```
unset [variable name]
```

Where **[variable name]** is the name of the environment variable.

For example, to unset the value of the ADD_AAP_LINKS_ONLY environment variable, type:

```
unset ADD_APP_LINKS_ONLY
```

The variable will remain unset until you define a new value for it.

In most cases, you will not need to unset your environment variables. Sometimes, however, environment variables work as on/off switches, enabling or disabling certain functionality. Unsetting the variable is the equivalent of turning the functionality off. Additionally, some environment variables use default values if they are not set. Unsetting the variable gives you an easy way of enabling the default functionality.

Note: If you want to *reset* the value of an environment variable, you do NOT need to unset it first. Rather, you can use the instructions provided in the previous section to set a new value. The new setting will override any existing setting.

3.1.4 Creating a Script to Set Environment Variables

The script **CreateEnvVariableScript.sh** extracts the environment variables from your database, and creates a new script that will set those environment variables. This dynamically created script can then be called at the beginning of other scripts to set the environment variables to the identified values.

It is important to use this program if you run scripts and executables as background or scheduled processes (see "Adding Oracle Environment Variable Information to a Script.") The script is also useful to restore your desired environment variables when you upgrade Virtua or after you change any existing environment variables.

The script that is created after running **CreateEnvVariableScript.sh** will have the format:

Env4${ORACLE_SID}_scripts.sh

(See the section "ORACLE_SID" for information about this variable.)

To ensure that the correct Oracle environment variables are set before a script is run,

1. Log in to your server as the **dbadmin** user.
2. Type: **`CreateEnvVariableScript.sh`**
3. Press Enter.

The **CreateEnvVariableScript.sh** creates a script that can be embedded in another script. The name of the new script will be output to the screen.

When you call this script from within another script, the correct environment variables will automatically be set before the remainder of the script is run.

See the section "Adding Oracle Environment Variable Information to a Script" for more information.

3.2 Running Scripts and Executables

In the Virtua system guides, for the sake of simplicity, we instruct you to run most scripts and executables from the prompt after you to log into the server as the **dbadmin** user. This instruction works fine as long as the folder the script exists in is defined within the $PATH environment variable AND is defined in the $PATH environment variable *before* any other folder that may contain the same variable. That said, it is always best to run scripts from inside of the $EXE_DIR and to type "./" as a prefix to the name of the script. In this way, you are ensured of running the correct script and the correct version of the script.

3.2.1 Running Scripts and Executables as Background Processes

Many of the scripts and executables you will need to run take a long time to complete. For these programs, you may find it necessary to run them as background processes. The basic format for running a script or executable as a background process is:

(nohup [program information] &)

Where **[program information]** is . . .

- The name of the script or executable that you want to run.
- Any command options and input or output options that you need to specify.

Below is an example of the program **KeywordIndex.exe** run as a background process:

```
(nohup KeywordIndex.exe -s5743 &)
```

3.2.2 Adding Oracle Environment Variable Information to a Script

To ensure the correct environment variables are set before a script is run as a background process (or is called by **crontab**; see **"**Working with Cron Jobs"), run **CreateEnvVariablesScript.sh** and insert the name of the script created by **CreateEnvVariablesScript.sh** into the third line of the target script.

To set environment variables at the beginning of a script,

1. Run **CreateEnvVariableScript.sh** as described in the section, "Creating a Script to Set Environment Variables." Make note of the name of the script created by **CreateEnvVariableScript.sh**.

2. Use VI or your desired text-editing program to modify the desired script, inserting the script name into the script.

The next time you call the script, the correct environment variables for that database will be set before the rest of the script is run.

Note: If you are not comfortable modifying scripts but still want to include environment variables at the beginning of one or more scripts, contact Innovative Customer Support for assistance.

3.3 Redirecting Output

When you run most scripts or executables, messages are displayed on the screen. You have the option of redirecting these messages to a file.

Programs use the following two methods for writing messages:

- Standard Out (**stdout**) - Outputs diagnostic messages, log information, or other relevant information.
- Standard Error (**stderr**) - Outputs error messages.

3.3.1 Redirecting Messages from stdout

To redirect messages from **stdout** to a file, use the following format:

[program information] > [filename]

Where . . .

- **[program information]** is . . .
  - The name of the script or executable that you want to run.
  - Any command options and input or output options that you need to specify.
- **[filename]** is the name of the file to which you want to write information.

Here is an example of a command to run the program **create_intermedia.pl** with **stdout** messages redirected to a file named **intermedia_data.txt**:

```
create_intermedia.pl > intermedia_data.txt
```

Tip: If you do not want to save messages to a file, you can redirect them to **/dev/null**. Doing so will prevent the messages from being logged.

3.3.2 Redirecting Messages from stderr

To redirect messages from **stderr** to a file, use the following format:

[program information] 2> [filename]

Where . . .

- **[program information]** is . . .
  - The name of the script or executable that you want to run.
  - Any command options and input or output options that you need to specify.
- **[filename]** is the name of the file to which you want to write information.

Below is an example of a command to run the program **Re_CreateHeadingSort.sh** with **stderr** messages redirected to a file named **Re_CreateHeadingSort.err**:

```
Re_CreateHeadingSort.sh 2> Re_CreateHeadingSort.err
```

Tip: If you do not want to save messages to a file, you can redirect them to **/dev/null**. Doing so will prevent the messages from being logged.

3.3.3 Redirecting Messages from stdout and stderr

To redirect messages from both **stdout** AND **stderr** to a file, use the following format:

[program information] > [filename] 2>&1

Where . . .

- **[program information]** is . . .
  - The name of the script or executable that you want to run.
  - Any command options and input or output options that you need to specify.
- **[filename]** is the name of the file to which you want to write information.

Below is an example of a command to run the program **KeywordIndex.exe** with both **stdout** and **stderr** messages redirected to a file named **keywordLog.log**:

```
KeywordIndex.exe > keywordLog.log 2>&1
```

Tip: If you do not want to save messages to a file, you can redirect them to **/dev/null**. Doing so will prevent the messages from being logged.

3.4 Detecting Processes

You can check to see if a program is running by using the **ps** command. This command returns a list of processes that match the criteria you specify.

To check the status a program, type:

```
ps -ef | grep [program name]
```

Where **[program name]** is the name of the program. You do not need to specify the entire program name.

For example, to see if the program **KeywordIndex.exe** is running, type:

```
ps -ef | grep KeywordIndex.exe
```

The server returns a row of information for each instance of the program that is running. Below is an example of the information returned when the server finds the process you requested.

dbadmin 18109 1 13 10:11 pts/24 00:05:19 KeywordIndex.exe

You can determine when the process was started by checking the time listed first on the line. In this example, the process was started at 10:11.

Many times, the server will also return a line that lists the process resulting from the command you entered to check for a process. When this happens, the row will include the word **grep** before the program name. In the example below, the first row lists the **KeywordIndex.exe** process, and the second row lists the process that is looking for **KeywordIndex.exe**.

dbadmin 18109 1 13 10:11 pts/24 00:04:39 KeywordIndex.exe
dbadmin 18115 17986 0 10:16 pts/24 00:00:00 grep KeywordIndex.exe

If the server does not find an instance of **KeywordIndex.exe** running, it will display no rows other than the **grep** process.

dbadmin 18109 1 13 10:16 pts/24 00:00:00 KeywordIndex.exe

3.5 Killing Processes

If you know the process ID of a program, you can use the **kill** command to end the process.

Note: The procedure for killing a **psdriver.exe** process differs slightly from the one described here. For information on killing a **psdriver.exe** process, see the section "Stopping a psdriver.exe Process" in this document.

To kill a process,

1. Use the **ps** command for the program that you want to end. For information about using the **ps** command, see the section "Detecting Processes" in this document.

   In the second column of the information returned by the **ps** command, the process number is listed.

2. At the prompt, type: **`kill [process #]`**

Where **[process #]** is the process number of the program you want to kill.

3. Press Enter.

Tip: If this command does not successfully kill the process, you can try it again with the **-9** option. This option specifies that the process should be ended without waiting to complete an action.

3.6 Working with Cron Jobs

This section describes procedures for setting up and running cron jobs. A cron job uses the UNIX **cron** daemon to schedule an action to run at a given interval. You can set up cron jobs to perform tasks that need to be run periodically, such as batch jobs or database backups.

Cron jobs are unique to each UNIX user. You must log in as the **dbadmin** user to create cron jobs that are related to Virtua.

3.6.1 Viewing Cron Jobs

To view the cron jobs for the dbadmin user,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: **`crontab -l`**
3. Press Enter.

The crontab entries display in the following format:

[Minute] [Hour] [Day of month] [Day of Week] [Command]

Where . . .

- **[Minute]** designates the minute at which the command is executed.
- **[Hour]** designates the hour at which the command is executed.
- **[Day of month]** designates the numerical day of the month on which the command is executed.
- **[Day of Week]** designates the numerical day of the week on which the command is executed (0 = Sunday and 6 = Saturday).

- **[Command]** is the command that will be executed. Enter the command the same as you would from the command prompt. You must specify the full path of the program and any other files used in the command.

Each value is separated by a single tab. If, for a time category, you want to specify all values, insert an asterisk (*****).

Below are three crontab entries:

30 0 * * /usr/vtls/virtua/r_2014_2/src/startps 1111
0 2 * 6 /usr/vtls/clas01/scripts/coldBackup.ksh
30 3 1 * /usr/vtls/clas01/scripts/mMaint.ksh > /usr/vtls/clas01/tmp/mMaint.log

The first entry specifies that the program **/usr/vtls/virtua/ r_2014_2/src/startps** runs every day at 12:30 A.M. This entry also specifies a command line parameter of **1111**.

The second entry specifies that the script **/usr/vtls/clas01/scripts/coldBackup.ksh** runs every Saturday at 2:00 A.M.

The third entry specifies that the script **/usr/vtls/clas01/scripts/mMaint.ksh** runs on the first day of each month at 3:30 A.M. This entry also specifies that the script writes **stdout** messages to the file **/usr/vtls/clas01/tmp/mMaint.log**.

Note: InfoStation uses cron jobs to schedule reports. Do NOT directly add, modify, or delete crontab entries for InfoStation reports. Instead, use the InfoStation interface to work with report scheduling. InfoStation crontab entries call the program **wrapper.pl** somewhere in the command.

3.6.2 Creating a Cron Job

To create a cron job,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: **`crontab -e`**

   The crontab entries for the **dbadmin** user appear.
3. On a new line, enter a new cron job in the following format:

[Minute] [Hour] [Day of month] [Day of Week] [Command]

Where . . .

- **[Minute]** designates the minute at which the command is executed.
- **[Hour]** designates the hour at which the command is executed.
- **[Day of month]** designates the numerical day of the month on which the command is executed.
- **[Day of Week]** designates the numerical day of the week on which the command is executed (0 = Sunday and 6 = Saturday).
- **[Command]** is the command that will be executed. Enter the command the same as you would from the command prompt. You must specify the full path of the program and any other files used in the command.

Separate each value with a single tab. If, for a time category, you want to specify all values, insert an asterisk (*****).

Note: Some scripts require that the cron job running the script use space delimiters instead of tab delimiters. Please check the relevant Virtua documentation to determine whether this is the case for a given script.

4. Save your changes.

3.7 Finding Text in Files

To scan a directory for files with a particular text string, use the following command:

find [directory] -name "[filename]" | xargs grep "[text]"

Where . . .

- **[directory]** specifies the directory that will be scanned. This command is recursive, meaning that all subdirectories of the directory you choose will be included in the scan. To specify the current directory, type**.**
- **[filename]** specifies the filename of the files that will be scanned. Use an asterisk to specify a wildcard.
- **[text]** specifies the text that you want to find.

For example, to search in the current directory (and all subdirectories) for text *sidebar* in every file with the extension *.html*, type:

find . -name "*.html" | xargs grep "sidebar"

4. Administrative Tips for Working with Oracle

For Virtua to run, the Virtua/Oracle database must be open and in good working order. It is important that you understand basic administration tasks that are required to ensure the proper operation of Oracle. This chapter describes some of those basic Oracle administration tasks. (For information on taking backups of the database, see the chapter "Database Backups" in this guide.) However, to keep your system in proper working order, we recommend that your knowledge of Oracle exceed what is described in this chapter.

Important: Unless your institution has its *own* Oracle license, your institution is not authorized to use Oracle utilities directly to perform queries and other commands on your database. If your institution has an embedded Oracle license only, you must contact Innovative to perform any tasks involving SQL*Plus or other Oracle utilities.

This chapter discusses the following topics:

⇒ Working with the Virtua/Oracle Database
⇒ Performing Basic Tasks

4.1 Working with the Virtua/Oracle Database

Your database is one of the most important parts of the Virtua system. The information stored in the database is crucial to the operation of the system. It is important that you approach any database-related task with great caution. It is not difficult to delete, change, or corrupt data inadvertently.

Note: Given the importance of your data, you must have in place an effective backup strategy. For information about backing up your database, see the chapter "Database Backups" in this document.

4.1.1 Recommended Knowledge for Working with the Database

To work with the Virtua/Oracle database it is important that, at a minimum, you . . .

- Understand the concept of a relational database system.
- Have a working knowledge of the UNIX operating system.
- Know how to run and manage scripts and executables on the server.

In addition to this technical knowledge, it is equally important that you possess a sense of caution when working with the database and understand the consequences of any actions.

4.1.2 Making Changes in the Database

The easiest way to make your Virtua system unusable or to irreversibly corrupt data is to alter or delete tables, data, or indexes without explicit instructions from Innovative. Even if a field in a table appears to correspond directly to data that appears in the Virtua Client or Chamo, it may be only part of the information that is used in the function.

Do NOT directly alter data in the Virtua database. Always use a client program or a server script or executable to make changes to the database.

4.2 Performing Basic Tasks in the Virtua/Oracle Database

This section covers the following basic Virtua/Oracle database administration tasks:

- Starting and stopping the database
- Identifying the databases that are running
- Identifying the version of Oracle that is running
- Identifying active database connections
- Working with the Oracle Listener
- Changing the password for Oracle users
- Working with tablespaces and data files
- Gathering statistics
- Creating a directory object

4.2.1 Starting Up and Shutting Down the Database

This section provides instructions for starting up and shutting down the database. Keep in mind that when the database is down, you will not be able to use Virtua.

Note: If your installation is a two-tiered setup, it is likely that your databases will be on one server and your applications (Virtua, Chamo, etc.) on another. The startup and shutdown procedures, described below, need to be performed on the *database* server.

4.2.1.1.1 Using *startdb* to Start the Virtua/Oracle Database

To use the startdb script to start the database,

1. Log in to your server as the **dbadmin** user.
   The system prompts you for the database number.
2. Enter the database number. When you do this, the environment variable is automatically set for you.
3. At the prompt, type: **`startdb`**
4. Press Enter.

   Oracle opens the database.

4.2.1.1.2 Using *stopdb* to Stop the Virtua/Oracle Database

To use the stopdb script to stop the database,

1. Log in to your server as the **dbadmin** user.
   The system prompts you for the database number.
2. Enter the database number. When you do this, the environment variable is automatically set for you.
3. At the prompt, type: **`stopdb`**
4. Press Enter.

   The script executes the **shutdown immediate** command. Oracle will close, dismount, and shut down the database without waiting for users to disconnect. Virtua will not be available until you restart the database.

4.2.2 Identifying the Databases that Are Running

To identify databases that are running,

1. At the prompt, type: **`ps -ef | grep ora_`**
2. Press Enter.

The system returns a list of processes that begin with **ora_**. Any process that ends in a system identifier such as *vtls99* indicates an instance of an Oracle database that is running.

Below is an example of a group of processes returned which indicate that the database *vtls99* is running.

```
oracle   24698     1  0 Apr25 ?        00:00:00 ora_pmon_vtls99
oracle   24700     1  0 Apr25 ?        00:00:01 ora_dbw0_vtls99
oracle   24702     1  0 Apr25 ?        00:00:03 ora_lgwr_vtls99
oracle   24704     1  0 Apr25 ?        00:00:14 ora_ckpt_vtls99
oracle   24706     1  0 Apr25 ?        00:00:02 ora_smon_vtls99
oracle   24708     1  0 Apr25 ?        00:00:00 ora_reco_vtls99
oracle   24710     1  0 Apr25 ?        00:00:01 ora_snp0_vtls99
oracle   24712     1  0 Apr25 ?        00:00:00 ora_arc0_vtls99
```

4.2.3 Identifying the Oracle Version

If you are unsure of which version of Oracle is running you can use the script **DisplayOracleRelease.sh**.

1. At the prompt, type: **`DisplayOracleRelease.sh`**
2. Press Enter.

This script displays to the screen the version of Oracle that is running on the current database; for example:

```
11.2.0.3.0
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit
Production
```

Note: If "Enterprise Edition" is not displayed in the output, then it is to be assumed that the Standard Edition of Oracle is being run.

4.2.4 Identifying Active Database Connections

To identify active database connections,

1. At the prompt, type: **`DisplayActiveDBConnections.sh`**
2. Press Enter.

Virtua displays a list of active database connections, which will be in the following format:

```
The following is a list of Virtua components and programs currently
connected to the database:

Program Connected              OS Process ID    Connection time
------------------------------ ---------------  -------------------------
SQL Developer                  3764             24-MAR-2014 12:33:35z
```

4.2.5 Viewing Information About Tables

Some Oracle tables may have options enabled to modify the types of access permitted to the database. The scripts **DisplayNoLoggingTables.sh** and **DisplayParallelTables.sh** identify tables in certain states. In most cases, you will run these scripts only when Innovative personnel request you do so.

4.2.5.1 Displaying Tables that Disallow Logging

The script **DisplayNoLoggingTables.sh** displays tables that have been started with the NOLOGGING option. This option can be useful for speeding up the data load or migration process. However, after such operations, *no* Virtua tables should be left in this state.

The script **DisplayNoLoggingTables.sh** can identify tables that need to be restarted in logging mode.

To display the tables that are in NOLOGGING mode,

1. At the prompt, type: **`DisplayNoLoggingTables.sh`**
2. Press Enter.

Virtua displays a list of tables running in NOLOGGING mode.

Important: If the script displays any tables as being in NOLOGGING mode, you MUST contact Innovative Customer Support immediately. Tables in NOLOGGING mode will be unrecoverable if there is a problem that requires recovery.

4.2.5.2 Displaying Tables Running in Parallel Mode

The script **DisplayParallelTables.sh** displays database tables that have been created or modified with the PARALLEL option, which is sometimes used to speed up the process of a data load or migration. However, after such operations, *no* Virtua tables should be left in this state.

You can run **DisplayParallelTables.sh** with the command-line option **update** to stop all database tables from running in PARALLEL mode.

1. At the prompt, type: **`DisplayParallelTables.sh`**
2. Press Enter.

Virtua displays a list of tables running in PARALLEL mode.

If there are tables running in PARALLEL mode,

3. (*Optional*) At the prompt, type: **`DisplayParallelTables.sh update`**
4. Press Enter.

All tables in the database are removed from PARALLEL mode.

Note: If this script displays any tables as PARALLEL degree greater than 1, that table may be using an excessive amount of resources, and it is recommended to take the table out of PARALLEL mode. You can change all tables to not be in PARALLEL mode by running this script with the "update" parameter.

4.2.6 Working with the Oracle Listener

Applications on servers other than the one on which your Virtua/Oracle database resides connect to the database via the Oracle Listener. To allow connections from remote servers and PCs, the Oracle Listener must be started on the database server. Once started, the Oracle Listener "listens" for requests from remote connections.

4.2.6.1 Starting the Oracle Listener

To start the Oracle Listener on the host server,

1. Log in to your database server as the **dbadmin** user.
2. At the server prompt, type: **`OracleListener.sh start`**
3. Press Enter.

4.2.6.2 Stopping the Oracle Listener

To stop the Oracle Listener on the host server

1. Log in to your database server as the **dbadmin** user.
2. At the server prompt, type: **`OracleListener.sh stop`**
3. Press Enter.

4.2.6.3 Displaying the Status of the Oracle Listener

To display the status of the Oracle Listener on the host server,

1. Log in to your database server as the **dbadmin** user.
2. At the server prompt, type: **`OracleListener.sh status`**
3. Press Enter.

4.2.7 Changing Passwords for Oracle Users

Innovative provides the script **ChangeOracleUserPassword.sh** to change the password for any Oracle user. This script must be run on the host database server. It can and should be used to change the password of Oracle users such as *dbadmin*, *reporter*, *syscli*, or *chamo*.

To change the password for an Oracle user,

1. Log in to your server as the **dbadmin** user.
2. Type:

```
ChangeOracleUserPassword.sh <Oracle user> <new password>
```

3. Press Enter.

The password for the specified Oracle user is changed.

Note: In addition to Virtua-specific Oracle users, your Oracle database includes users that have permission to view and alter your Virtua tables. For security reasons, it is important that the default password for each of these users is changed upon installation. For the initial installation, Innovative will change the default passwords for you, but you are responsible for changing the passwords for these users at regular intervals.

The Oracle users with access to the Virtua tables are:

- **sys** - An administrative user for Oracle. The default password for **sys** is **change_on_install**
- **system** - An administrative user for Oracle. The default password for **system** is **manager**
- **ctxsys** - A user created for the interMedia application. The default password for **ctxsys** is **ctxsys**

While you should keep these passwords secure, Innovative will periodically need to access your database using these Oracle users to maintain and troubleshoot your system.

4.2.8 Refreshing Permissions for Virtua Oracle Users

The script **RefreshSyns.sh** refreshes the permissions of Virtua Oracle users, such as the **dbadmin** or **chamo** user. These are refreshed automatically upon any upgrade or migration, but Innovative personnel may instruct you to run this script in certain circumstances.

To refresh the permissions for Virtua Oracle users,

1. Log in to your server as the **dbadmin** user.

2. Type **`RefreshSyns.sh`**

3. Press Enter.

The permissions for Virtua's Oracle users are refreshed.

4.2.9 Working with Tablespaces and Data Files

Virtua comes with scripts to give you the ability to review, analyze, and manipulate the tablespaces and their associated data files in your Oracle database (for background information about tablespaces and data files, see the section "About Oracle Tablespaces and Data Files"). Each of the scripts are described below.

4.2.9.1 Checking Tablespaces and Data File Sizes

The script **CheckTablespaceFileSize.sh** lets you check the tablespaces and their associated data files. The script provides the available free space within the tablespace and the room available for each individual data file to grow if the autoextensible parameter is set to ON.

To check the tablespaces and size of data files,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: **`CheckTablespaceFileSize.sh`**
3. Press Enter.

The script returns a table of information. Below is an example of the first line of the table that is returned.

TableSpace	F	ID File	All sizes in MB	Data File Max Size	Current File Size	Current Free Space	Room To Grow	AE
SYSTEM	1	/usr/vtls/clas24/system/sys_01.dbf		512	512	141	141	NO

The information provided by **CheckTablespaceFileSize.sh** helps you determine if you need to take action to add data files or resize data files.

4.2.9.2 Adding a Data File to a Specified Tablespace

The script **AddTablespaceDataFile.sh** lets you add a single 64MB data file that is autoextensible to 2G for a specified tablespace.

To add a data file to a specific tablespace,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:
   AddTablespaceDataFile.sh [tablespace name] [absolute path to datafile]

3. Press Enter.

Example:

```
AddTablespaceDataFile.sh small_tables usr/vtls/clas01/data/sm_tbl_03.dbf
```

4.2.9.3 Resizing a Data File within a Specified Tablespace

The script **ResizeTablespaceDataFile.sh** lets you resize a data file or temp file for a specified tablespace.

To resize a data file within a specific tablespace,

1. Log in to your server as the **dbadmin** user.

2. At the prompt, type:
   ResizeTablespaceDataFile.sh [tablespace name] [datafile name] [max size]

3. Press Enter.

Example:

```
ResizeTablespaceDataFile.sh UNDO /usr/vtls/clas87/undo/undo_01.dbf 4G
```

4.2.10 Gathering Statistics

Virtua provides two scripts for gathering statistics. **GatherSchemaStats.sh** gathers statistics on the database schema, and **GatherTableStats.sh** gathers statistics on the database tables.

4.2.10.1 GatherSchemaStats.sh

The script **GatherSchemaStats.sh** gathers statistics on an Oracle schema. Statistics are used by the Oracle optimizer to design the most efficient plan for retrieving data from the database.

Run this script after any major changes to a large number of database tables. This script is automatically run at the end of every database migration.

The default schema is DBADMIN.

Usage: `GatherSchemaStats.sh [schema] [parallel]`

4.2.10.2 GatherTableStats.sh

The script **GatherTableStats.sh** gathers statistics on a specific table. Statistics are used by the Oracle optimizer to design the most efficient plan for retrieving data from the database.

Run this script after any major changes to a table's data contents or its structure.

Usage: `GatherTableStats.sh schema table [degree_of_parallel]`

Example: `GatherTableStats.sh dbadmin patron 4`

4.2.11 Creating a Directory Object

Virtua provides a script **CreateDirectoryObject.sh** for creating an Oracle Directory Object.

A directory object is a logical object linked to a physical directory on the host server. It allows database users to read/write from a specific location on the host server. Before you can create an object for a directory, the directory must exist on the host server.

Usage: `CreateDirectoryObject.sh ObjectName FilePath`

Where **`ObjectName`** is the logical name or pointer stored within the database that points to the FilePath on the host server, and **`FilePath`** is the physical location being linked to on the host server.

5. Working with the psdriver.exe Program

The program **psdriver.exe** is the main server program in Virtua. This is the program to which the Virtua client connects. If this program is not running, you will not be able to use the Virtua client.

We recommend that you start at least two **psdriver.exe** processes:

- A process that handles persistent connections from the Virtua client.
- A process that handles repeated, stateless connections from Web-based Z39.50 clients.

Additionally, we recommend that you recycle your **psdriver.exe** processes periodically. If you run them uninterrupted for too long, they can use excessive amounts of memory and impede system performance. Ideally, you should restart your **psdriver.exe** processes daily. At a minimum, you should recycle your processes weekly.

This section discusses the following topics:

⇒ Running psdriver.exe
⇒ Logging Server Output
⇒ Identifying psdriver.exe Processes
⇒ Stopping a psdriver.exe Process
⇒ Working with User Limit Constraints
⇒ Working with the License Key

5.1 Running psdriver.exe

This section provides instructions for starting the program **psdriver.exe**. This is the primary program in the Virtua server installation.

Notes:

- If your installation is a two-tiered setup, it is likely that your databases will be on one server and your applications (Virtua, Chamo, etc.) on another. The **psdriver.exe** program needs to be started on the *application* server.

- The Open Skies URL setting MUST be configured correctly in the Profiler and the Open Skies-enabled Chamo instance must be running for you to start **psdriver.exe**. See the *Virtua Profiler/OPAC Parameters User's Guide* for more information.

There are two methods for running the program **psdriver.exe**:

- By directly executing the program from the command line.
- By running a script.

5.1.1 Running psdriver.exe Directly from the Command Line

To run the program psdriver.exe,

4. Log in to your server as the **dbadmin** user.
   The system prompts you for the database number.
5. Enter the database number.
6. At the database prompt, type: **`psdriver.exe [port #]`**

   Where **[port #]** is the port number on which you want the program to listen for connection requests. This number should be a four- or five-digit number between 1025 and 32767 that is not already being used by another process.

7. Press Enter.

The **psdriver.exe** process starts on the port you specified. You can use the Virtua client to connect to the Virtua server program via this port.

Tip: Since this program must be running at all times while Virtua is available, you may want to run it as a background process. For information about running a program as a background process, see the section "Running Scripts and Executables as Background Processes" in this document.

5.1.2 Running psdriver.exe from a Script

The script startps runs the program **psdriver.exe** as a background process. All log and error messages are redirected to **/dev/null**, meaning that they will not be logged.

To use startps to run psdriver.exe,

1. Log in to your server as the **dbadmin** user.
   The system prompts you for the database number.
2. Enter the database number.
3. At the database prompt, type: **`startps [port #]`**

   Where **[port #]** is the port number on which you want the program to listen for connection requests. This number should be a four- or five-digit number between 1025 and 32767 that is not already being used by another process.

4. Press Enter.

The **psdriver.exe** process starts on the port you specified. You can use the Virtua client to connect to the Virtua server program via this port.

5.2 Logging Server Output

When running the **psdriver.exe** program, you can have it log messages about events such as error and transaction information. You can log the messages to the screen or to a file. This information is useful for detecting and diagnosing problems that you encounter while using Virtua.

There are five levels of logging from which you can choose. The lower the log level, the greater the number of messages logged.

Log Levels

- **1** - Debug messages - Displays messages that are used by Innovative to debug Virtua when a problem arises. Additionally, this log level displays messages that qualify for logging at levels 2, 3, 4, and 5. We do not recommend that you use this log level as part of your day-to-day operations. An example of a debug message is the SQL statement that is used in a transaction:

  - EXECUTE SQL: "SELECT COUNT(rowid), SUM(SUBFIELD_DATA_LENGTH) FROM ISO_2709_LONG_PARAMS WHERE id = 20006 AND idtype = 211 AND subtype = 'eng '"

- **2** - Information messages - Displays information about transactions. Additionally, this log level displays messages that qualify for logging at levels 3, 4, and 5. This is the log level that is used if one has not already been set for your system. An

example of an information message is the following message that appears when you modify an item record.

- Item record 37502 modified by 9

- **3** - Warning messages - Displays messages that might indicate an error condition. Additionally, this log level displays messages that qualify for logging at levels 4 and 5. An example of a warning message is the message that appears when you attempt an OPAC search using a non-existent search type:

  - searchLocal: unknown ID type (355)

- **4** - Error messages - Displays messages that indicate an error condition. Additionally, this log level displays messages that qualify for logging at level 5.

- **5** - Critical messages - Few messages are logged on this level. Messages logged at this level generally describe an event such as a system crash. Use this level if you do not want to log messages. **Tip:** If you decide to redirect log messages to **/dev/null**, you should choose this option so that the server does not waste resources writing messages.

5.2.1 Setting the Log Level

You need to specify the log level *before* you start the **psdriver.exe** process. The log level you set applies to any processes you start until you end the UNIX session.

To set the log level,

1. Log in to your server as the **dbadmin** user.

2. Type: **`export V_LOG_LEVEL=[level]`**
   Where **[level]** is the number representing the log level you want to use. For example, if you want to log messages at level 4, type: **`export V_LOG_LEVEL=4`**

3. Press Enter.

Note:
- Depending on the number of transactions your library handles and the extent to which you review your logs, you should set the log level between 2 (Information) and 4 (Error).
- If you do NOT want to record log messages, set the log level to 5.

By default the messages that are generated by the **psdriver.exe** program appear on the screen unless you choose to redirect them to another destination. For information about redirecting these messages to another destination, see the sections "Redirecting Output" and "V_LOG_TYPE" in this document.

5.2.2 Using syslog to Log Messages

Every UNIX system has a syslog facility. The syslog tool is a flexible way of logging messages. It lets you set the level of messages that are logged and the location of the log files. It also lets you configure numerous aspects of the logging process. For example, you can choose to have errors broadcast, e-mailed, or sent to another server.

The syslog tool should be used only by system administrators. If you do not have experience configuring and using syslog, you should not use it to log Virtua messages.

To direct log messages to syslog,

1. Log in as the **dbadmin** user.
2. Type: **`export V_LOG_TYPE=2`**
3. Press Enter.

For more information about the V_LOG_TYPE environment variable, see the section "V_LOG_TYPE" in this document.

Note:
- Virtua information always logs to the LOCAL0 facility.
- The logging levels for syslog correspond to those available for Virtua.
- For more information about syslog, refer to your UNIX documentation.

5.3 Identifying psdriver.exe Processes

To identify psdriver.exe processes,

1. Log in as the **dbadmin** user.
2. Type: **`ps -ef | grep psdriver`**
3. Press Enter.

   The system responds with information about the **psdriver.exe** processes that are currently running. Below is an example of the output from running this command:

```
dbadmin  8228     1      0 16:53:20 ?           0:00 psdriver.exe 1901
dbadmin 13428  8228  0 11:10:04 ?           0:04 psdriver.exe 1901
dbadmin 13434  8228  0 11:10:04 ?           0:04 psdriver.exe 1901
dbadmin 13512  8228  0 11:10:04 ?           0:04 psdriver.exe 1901
dbadmin 11637  8228  0 11:10:04 ?           0:04 psdriver.exe 1901
```

In our example above,

- The parent **psdriver.exe** process is in the first row.
- The child processes of the parent process are in the subsequent rows.
- The second column indicates the Process ID.
- The last column indicates the port number.

Therefore, in this example, there is a parent **psdriver.exe** process running on port 1901 with a process ID of 8228. There are child processes running with the following process IDs:

- 13428
- 13434
- 13512
- 11639

5.4 Stopping a psdriver.exe Process

For certain database tasks you will want to stop, or kill, the **psdriver.exe** process. Before you kill the parent process, you should kill the child **psdriver.exe** processes.

Using the example in the previous section:

To stop the child psdriver.exe processes,

1. Log in as the **dbadmin** user.
2. Type: **`kill 13428 13434 13512 11637`**
3. Press Enter.

Next, to stop the parent psdriver.exe process,

- Type: **`kill 8228`**

Tip: If this command does not successfully kill the process, you can try it again with the **-9** option. This option specifies that the process should be ended without waiting to complete an action.

5.5 Working with User Limit Constraints

Your Virtua license allows a number of simultaneous connections to the Virtua server. When you reach this number, the server will not accept any more connections and will display an Error message (*Figure 5-1*).

Note: When a user logs off of the client, it takes 60 seconds for the session to be considered released.

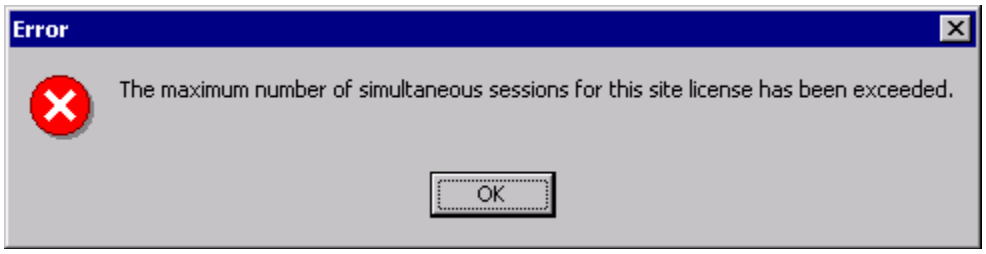


Figure 5-1. Virtua Client - Licensed User Limit Reached

Be aware that connections from *any* application to the **psdriver.exe** program are included in the count of simultaneous users. This includes any third-party application

For information about setting the Max[imum] Simultaneous Sessions limit in a user's profile, see the *Virtua Profiler/Introduction and Global Settings User's Guide*.

5.5.1 Viewing a Complete List of Connections

The Virtua server is a Z39.50-compliant product. This means that anyone who knows the IP address (or hostname) of your Virtua server and the port number on which you are running **psdriver.exe** can connect to Virtua via a Z39.50-compliant client.

ALL connections to the Virtua server count toward your user license limit. Connections from remote, third-party clients may interfere with your users' ability to

use Virtua. If you notice an unusually high number of connections to your Virtua server, you may want to view a detailed list of the connections to determine how many of them are of non-local origin.

To view a list of all connections to Virtua,

1. Log in to the server as the **dbadmin** user.
2. At the prompt, type:

   `netstat -a | grep xxxx`

   Where **xxxx** is the port number on which you are running **psdriver.exe**.
3. Press Enter.

   A list of connections appears.

Below is an example of the output from the **netstat** command:

tcp 0 0 server.vtls.edu:1234 visitor.anywhere.edu:5678 ESTABLISHED

In this example, **psdriver.exe** is running on a machine with the hostname **server.vtls.edu**, using the port **1234**. A client connection to that port has been made from the remote host **visitor.anywhere.edu**.

Hint: If the results of your **netstat** command display numeric IP addresses instead of hostnames, you can use the UNIX **nslookup** command to translate numeric addresses into hostnames.

After you review the results, you can investigate any connections being made from non-local sources.

5.6 Working with the License Key

The Virtua server program requires that an active license key exist in your system. If your system has an expired or invalid key, the **psdriver.exe** program will not accept connections from any source. You can obtain a new key from Innovative customer services.

Note: Features for FRBR, Linked Bibliographic Records, and the 3M Patron SelfCheck are enabled via the license key.

Each license key is specific to a particular database. A key for your clas01 database will not work on your clas99 database.

Note: When your current license key is within 45 days of expiring, you will be warned each time you log in to the Virtua client.

5.6.1 Determining the Virtua Release

If you are unsure which version of Virtua is running on one of your databases, you can run the script DisplayVirtuaDBRelease.sh to display the version of Virtua used by the database

To determine the Virtua release,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: `./DisplayVirtuaDBRelease.sh`

The database's Virtua version is printed to the screen:

```
The database is currently at Virtua version: 2014.2
```

5.6.2 Verifying the License Key and Expiration Date

To verify the license key for your database,

1. Log in to your server as the **dbadmin** user.
2. Make certain that the ORACLE_SID environment variable is set correctly. For information on the ORACLE_SID environment variable, see the section "ORACLE_SID" in this document.
3. At the prompt, type: **`GetExpirationDate.exe`**
4. Press Enter.

   The program displays the expiration date of the license key for the database.

5.6.3 Updating the License Key

To update the license key for your system,

1. Log in to your server as the **dbadmin** user.
2. Make certain that the ORACLE_SID environment variable is set correctly. For information on the ORACLE_SID environment variable, see the section "ORACLE_SID" in this document.
3. At the prompt, type: **`SetServerKeyCode.exe`**
4. Press Enter.

   The program prompts you for a license key code.

5. Type the license key code you received from Innovative.
6. Press Enter.

   If you entered a valid code, the software updates the key in your system.
7. Repeat steps 2-6 for each database on your server.

6. Database Backups

Your data is the most critical part of your library automation system. Without a good set of data, the rest of your system is useless. After an event such as a disk failure, you can replace the Virtua software that was installed on the disk, but you cannot replace the data if you do not have a good backup. To protect your data, it is important that you put in place an effective backup system.

This chapter provides basic information for taking backups of your database. The scope of the information in this chapter is limited to basic descriptions of important database backup concepts and generic procedures for performing necessary tasks. The descriptions and procedures in this guide may not necessarily be appropriate for your library.

Backup strategies vary from library to library (and the reasons for needing to recover differ from event to event). Based on the backup strategy you choose and the event that precipitates the need to recover, procedures for recovering from backup will vary. *We strongly recommend that you learn how to recover from your backup, and that you practice recovering from backup.* That said, we do offer instructions in this chapter for using two Innovative-provided scripts, one for taking a hot backup and one for recovering from a hot backup.

Many books are available from Oracle or other authoritative sources that provide more detail about backups than you will find in this guide. We recommend that you learn as much as possible about this topic.

Important: If you are not the database administrator for your system, you should not complete any of the procedures outlined in this chapter without first consulting the database administrator.

This chapter covers the following topics:

⇒ Importance of Database Backups
⇒ Options for Doing Database Backups
⇒ Validating Backup Files
⇒ Managing Media
⇒ Innovative Recommendations

6.1 Importance of Database Backups

When your database fails, nothing is more important than having a good backup from which you can restore your data. You can purchase new hardware and reinstall your software, but if you do not have a good backup you cannot restore your data.

Backup procedures can be time consuming and tedious. However, if your data is important to your library, you should spend as much time as possible developing, implementing, and practicing a backup strategy.

Some of the data that you risk losing *permanently* if you do not have a good backup strategy is:

- All of your bibliographic records.
- All of your patron records.
- All of your circulation transactions and accounts.
- All of your acquisitions information.

Basically, if you do not implement a good backup strategy, you risk losing ALL of the data in your database.

A good backup strategy is . . .

- **Regular** - Take full backups of the database as often as possible.
- **Redundant** - Maintain multiple copies of your backup files. Data tapes and disks that store backup files can fail as easily as your server disk. You should store at least one copy of your backup files offsite.
- **Reliable** - Test your backup files. Having corrupt backup files is no better than having no backup files at all.
- **Rehearsed** - Practice recovering from backup. The worst time to learn how to recover from backup is after your database has failed.

6.2 Options for Doing Database Backups

This section provides details about some of the database backup options available to you. Where appropriate, we provide basic procedures for completing these tasks.

Important: The procedures outlined in this section detail one way of performing a task. There are other ways of performing the tasks described in this section that may be more suitable for your library. Additionally, the information in this section is NOT comprehensive. Depending on your situation, there may be more information that you need before following any of these procedures.

6.2.1 Archive Logging

If archive logging is enabled for your system, Oracle periodically archives your redo log files to a specified location. If you ever need to recover from backup, you can restore from a full backup and then apply the archive logs. If you do not use archive logging, your database can only be restored to the point in time at which you took the full backup.

With archive logging, you can recover up to the point at which the last redo log was archived. If your system is set up to write redo logs to a separate disk, you might also be able to recover to the last transaction that was committed before the database failed.

This section provides basic instructions for configuring, enabling, and disabling archive logging.

6.2.1.1 Configuring Archive Logging

In the **initvtls.ora** file, you can configure the following aspects of archive logging:

- The location to which Oracle writes archive logs.
- The format of the filenames of the archive log files.

Both of these options are configured in the **initvtls.ora** file. Changes you make to this file will not take effect until you restart the database.

6.2.1.1.1 Determining the Location to Which Oracle Writes Archive Log Files

The parameter log_archive_dest in the **initvtls.ora** file determines the location to which Oracle writes archive log files.

Important: When selecting a destination for your archive log files, it is important that you choose a partition that will have enough space to accommodate the archive logs that will accumulate until the next full backup. If the partition to which you write archive logs becomes full, your database *will cease to operate* until more space is made available.

To determine the location to which Oracle writes archive log files,

1. Open the file **initvtls.ora** in a text editor.
2. Find the parameter log_archive_dest.
3. For this parameter, type the full path to the directory in which you want Oracle to write archive logs. This directory must exist before Oracle will write archive logs.

   Note:
   - Do NOT choose a directory on the same disk as your database files. If your disk fails, archive log files stored on that disk will be just as inaccessible as your database files.
   - If desired, you can include, at the end of the path, text that will be used as a prefix for each filename.
4. Save your changes to the file.

6.2.1.1.2 Duplexing Archive Logs

Like any other data, your archive logs are not immune to corruption or media failure. You can configure Oracle to *duplex* archive logs by writing a second set of archive logs to a separate disk. If you have the resources to duplex your archive logs, we recommend that you do so as it will further insulate you from data loss.

To configure Oracle to duplex archive logs,

1. Open the file **initvtls.ora** in a text editor.

2. Find the parameter log_archive_duplex_dest. If this parameter does not exist, create it by typing on a new line: **`log_archive_duplex_dest =`**

3. For this parameter, type the full path to the directory in which you want Oracle to write duplicate archive logs. This directory must exist before Oracle will write archive logs.

   Note:
   - Do NOT choose a directory on the same disk as your database files or your first set of archive logs. If your disk fails, archive log files stored on that disk will be just as inaccessible as any other files.
   - If desired, you can include, at the end of the path, text that will be used as a prefix for each filename.

4. Find the parameter log_archive_min_succeed_dest. If this parameter does not exist, create it by typing on a new line: **`log_archive_min_succeed_dest =`**

5. For this parameter, type **`2`**.

6. Save your changes to the file.

6.2.1.1.3 Determining the Format of Archive Log Filenames

To determine the format of archive log filenames,

1. Open the file **initvtls.ora** in a text editor.
2. Find the parameter log_archive_format.
3. For this parameter, type the text that you want to appear in the filename of each file. Additionally, wherever you choose, type **`%s`** to specify that an automatically incrementing number be inserted.

   For example, if you specify **`log%s.arc`**, Oracle will create archive log files with the names **log1.arc**, **log2.arc**, **log3.arc**, etc.

4. Save your changes to the file.

6.2.1.2 Enabling Archive Logging

The script **EnableArchiveLogging.sh** enables archive logging. The script shuts down the database, disconnects all users, and restarts the database in archive log mode.

To enable archive logging,

1. Log in to your server as the **dbadmin** user.
2. Create a directory to store archive logs, such as **/usr/vtls/arch**

3. Add the following parameter to the bottom of the initvtls.ora:
   LOG_ARCHIVE_DEST_1='Location=/usr/vtls/arch'

4. Restart the database to enable the changes.

5. Run: **$EXE_DIR/EnableArchiveLogging.sh**

6. Press Enter.

When Oracle returns the text *ARCHIVELOG*, archive logging will be enabled. Your redo logs will be archived as they fill up to the directory specified by the log_archive_dest parameter.

Important: After enabling archive logging, shut down the database and take a full cold backup of the database. If you do not take a cold backup at this point, your archive logs will not be usable. For instructions for taking a cold backup, see the section "Procedures for Taking a Cold Backup" in this document.

6.2.1.3 Determining Whether Archive Logging Is Enabled

The script **IsArchiveLoggingEnabled.sh** tells you whether archive logging is enabled in the database.

To determine whether archive logging is enabled,

1. Log in to your server as the **dbadmin** user.
2. At the prompt, type: **IsArchiveLoggingEnabled.sh**
3. Press Enter.

The script will return one of the following values:
0 – No, it is not enabled.
1 – Yes, it is enabled.

6.2.1.4 Disabling Archive Logging

The script **DisableArchiveLogging.sh** disables archive logging. The script shuts down the database, disconnects all users, and restarts the database in non-archive log mode.

To disable archive logging,

1. Log in to your server as the **dbadmin** user.
2. Run: **$EXE_DIR/DisableArchiveLogging.sh**
3. Press Enter.

   Archive logging is now disabled. Your redo logs will NOT be archived.

Important: After disabling archive logging, shut down the database and take a full cold backup of the database. For instructions for taking a cold backup, see the section "Procedures for Taking a Cold Backup" in this document.

6.2.2 Cold Backups

A cold backup consists of a copy of all database files. Before you take a cold backup, you must shut down the database until the backup is complete. While the database is down, you will not be able to use Virtua or perform any actions on the database.

In most cases, cold backups are easier to take than hot backups. Unless you cannot bring the database down for long enough to take a cold backup, we recommend that you use this method.

6.2.2.1.1 Procedures for Taking a Cold Backup

We recommend taking a cold backup by simply tarring the entire database directory. Using this method, you need to keep in mind that if you recover, you will return your database directory to the *exact* state that it was in at the time of the tar.

Important: If you need to use this tar file to recover, you must first copy the current archive and redo log files to another location before untarring the backup files. If you do not copy the archive and redo log files to another location, you will overwrite them with the old files in the backup.

To create a tar file of the entire database directory,

1. Shut down the database. For instructions for shutting down the database, see the section "Shutting Down the Virtua/Oracle Database" in this document.

2. Navigate to the directory above the database directory that you want to backup.

3. At the prompt, type:

```
tar -chv [filename] [directory name]
```

   Where . . .
   - **[filename]** is the name of the file to which you want to write the backup.
   - **[directory name]** is the name of the database directory.

   Tip: The **-h** option specifies that the contents of linked directories will be included in the tar file.

4. Press Enter.

   A tar file is created storing the entire contents of the database directory.

5. Restart the database.

6.2.3 Hot Backups

A hot backup is a backup you can take while Virtua is running. RMAN, the Oracle recovery manager tool, is used for performing hot backups, and Innovative provides scripts that utilize this tool. If your library cannot bring down the database long enough to take a cold backup, you will need to take a hot backup instead.

Note: If you do not use archive logging, you cannot take a hot backup.

Taking a hot backup can be a complicated procedure. To make it less complicated, Innovative provides the self-documenting script **CreateRMANOnlineBackup.sh**. This script creates a custom restore script **Restore_clas$DB_DIGITS.sh** for each backup. You can use **Restore_clas$DB_DIGITS.sh** to restore the entire database in the event of a loss.

Below are instructions for using the scripts.

6.2.3.1 Using CreateRMANOnlineBackup.sh

The **CreateRMANOnlineBackup.sh** script creates an RMAN online backup while the database is open and actively being used. The database must be in archive log mode.

You can run **CreateRMANOnlineBackup.sh** interactively or as a background process (i.e., via nohup or crontab). We recommend that you run the script during off-peak hours.

To create an RMAN online backup of your open database,

1. Log in to the database host server with Oracle 11*g* or higher as **dbadmin**.

2. Set up in the **initvtls.ora** a flash recovery area where all backup files will be stored. For details on how to do this, see the instructions below.

3. Run the backup script by typing: **`./CreateRMANOnlineBackup.sh`**

6.2.3.1.1 Setting Up a Flash Recovery Area

To set up a flash recovery area to store the database backup files,

1. Create a directory for the flash recovery area *outside of* the database home /usr/vtls/clasXX. To do this, type:

   `mkdir /usr/vtls/rman/backup/$ORACLE_SID`

2. Add the following two parameters to the bottom of the **/usr/vtls/clasXX/initvtls.ora**:

   db_recovery_file_dest=/usr/vtls/rman/backup/$ORACLE_SID
   [Replace $ORACLE_SID with the actual value (e.g., vtls01).]

   db_recovery_file_dest_size=20G
   (The db_recovery_file_dest_size directory must be at least one-half the total size of the database.)

   Note: It is *highly* recommended that the above directory structure reside on an NFS, an NAS, an SAN, or some other high-availability hardware architecture so that in the event of a full server loss, the backup files can be recovered and used to restore the database.

3. Restart the database to enable the new parameters.

Caution: Ensure that the disk space specified via the db_recovery_file_dest_size parameter is available to Oracle on the file system. Pro-actively monitor it. If the **db_recovery_file_dest** directory runs out of disk space, the database will halt.

All backup files will now be stored in the **db_recovery_file_dest** directory: **/usr/vtls/rman/backup/$ORACLE_SID**.

6.2.3.2 Creating a Backup with an Spfile

You may wish to create a backup that includes the initialization instructions contained in **initvtls.ora**. In most cases, this is not necessary.

However, if you want to create an RMAN backup using an spfile instead of a pfile, or if Innovative personnel instruct you to do so, you can use the scripts described below.

Hint: If you do not know the difference between a pfile and an spfile, then it is best for you to use Innovative's standard RMAN backup procedure. Contact Innovative Customer Support if you have further questions.

6.2.3.2.1 Creating an Spfile

The script **Cr_Spfile.sh** lets you create an spfile that contains the parameters in initvtls.ora. Unlike a pfile, this spfile is stored in your database, and so will be included when you take a backup of or restore a database.

Contact Innovative Customer support for help running this script the first time you run it.

6.2.3.2.2 Checking to see if an Spfile is Being Used

The script **IsSpfileInUse.sh** lets you check to see if your backup file is an spfile.

To run IsSpfileInUse.sh,

1. Log in to your server as dbadmin
2. Type **`./IsSpfileInUse.sh`**
3. Press the Enter key.

If an spfile is documented in the startup file, the script prints a **`1`** to the screen.
If the startup file specifies initvtls.ora, the script prints a **`0`** to the screen.

Note: Oracle determines whether to use **initvtls.ora** or **spfile** based on how the database is started:

- If the **pfile=** parameter is specified in the startup command, which Virtua's default behavior, then the **initvtls.ora** is utilized, and any existing **spfile**s on the server are ignored.
- If the database is started with*out* specifying a **pfile=** parameter, then an **spfile** is utilized, and the **initvtls.ora** file is ignored.

6.2.3.3 Restoring the Database

The **CreateRMANOnlineBackup.sh** script calls **CreateRMANRestoreScript.sh** to automatically create a custom restore script **Restore_clas$DB_DIGITS.sh**, which you can use to restore the entire database in the event of a loss. Once created, the restore script MUST be tested and modified to ensure accuracy and completeness.

Important: You must restore backups on the same platform from which the backup was taken.

The automatically generated restore script will be associated with the latest set of backup files located in the **/usr/vtls/rman/backup/$ORACLE_SID** directory. If you want to recover older versions of the database (not just the most recent backup), then after doing an RMAN backup, tar/move the entire directory contents of **/usr/vtls/rman/backup/$ORACLE_SID** to an alternate location.

Before running **Restore_clas$DB_DIGITS.sh**, do the following:

1. Shut down any database processes that are running on the server that you plan to restore the database to.
2. Remove the old **/usr/vtls/clasXX** directory, if it exists.
3. Restore the OS, \$ORACLE_HOME, and other Virtua binary files such as \$EXE_DIR.
4. Create a new database home directory by typing: **`mkdir /usr/vtls/clasXX`**
5. Run **\$EXE_DIR/Cr_DB_dirs.sh** from within the new database home directory.
6. Create a new **initvtls.ora**.
7. Ensure all RMAN backup files are located in a directory named **/usr/vtls/rman/backup/\$ORACLE_SID**.

To restore the database from the backup files,

1. Run **$BACKUP_DIR/Restore_clas$DB_DIGITS.sh**

   Where **"$DB_DIGITS"** is a variable representing the number of the actual database that was backed up.

   The database will be restored and closed.

2. Restart the database.

6.3 Validating Backup Files

As important as it is to take regular backups, it is equally important to ensure that these backups are useable. If your backup files are corrupt, you will NOT be able to recover from a disk failure.

Oracle provides a program, **dbv**, which checks data files for corruption. You can use this program to check data files for corruption.

Note:
- The program **dbv** cannot be used to validate exports or redo log files.
- While a data file may be free of corruption, it may not necessarily be useable for recovery.

6.3.1 Validating Data Files

To check for corruption in a data file,

1. Navigate to the directory in which the data file is stored.

2. At the prompt, type:

   ```
   dbv file=[filename]
   ```

   Where **[filename]** is the name of the data file that you want to validate.

3. Press Enter.

The program examines the file for corruption and prints out the results to the screen.

If the program finds corruption in the file, it lists information about the corruption in the following format:

```
Page 298 is marked corrupt
***
Corrupt block relative dba: 0x9880012a (file 0, block 298)
Bad header found during dbv:
Data in bad block -
 type: 133 format: 221 rdba: 0xe9f839b7
 last change scn: 0xeefc.32fbf844 seq: 0x94 flg: 0x04
 consistency value in tail: 0xc8efac71
 check value in block header: 0x31ee, computed block checksum: 0x644
 spare1: 0xf1, spare2: 0xe0, spare3: 0x36b
```

If the program does not find corruption in the file, it prints summary information about the scan in the following format:

```
DBVERIFY - Verification complete

Total Pages Examined         : 84992
Total Pages Processed (Data) : 50328
Total Pages Failing   (Data) : 0
Total Pages Processed (Index): 698
Total Pages Failing   (Index): 0
Total Pages Processed (Other): 733
Total Pages Empty            : 33233
Total Pages Marked Corrupt   : 0
Total Pages Influx           : 0
```

Note: For information about other options available for the **dbv** program, refer to your Oracle documentation.

6.4 Managing the Physical Media of Backup Files

Another critical aspect of your backup strategy is the physical media on which your backups are stored. If your tape or backup disk is unusable, your backup files on that media are also unusable. It is important that you ensure that at least one copy of your backup files will be available in any situation.

Tip: We strongly recommend that your server include a tape drive onto which you can transfer large amounts of data, such as a full backup.

Here are some tips for managing backup files:

- Disk backups should reside on a separate physical disk from your database. Ideally, copies of the backup should be ported to another server or a separate disk storage device.
- Data storage media can fail, so it is a good idea to maintain multiple copies of backup files (i.e., archived to tape and disk).
- To guard against data loss in the event of a catastrophe, store at least one copy of your backup files offsite (e.g., a safety deposit box).

If you archive your backups on data tapes . . .

To ensure the integrity of your backups, you must make sure that the data tapes on which you store your data are in working order. Some basic tips for working with data tapes:

- Do NOT use a single data tape more than 50 times. **Note:** You should check with the usage recommendations from the tape's manufacturer.
- Do NOT rely on a single tape for all of your backups.
- Establish a rotation for your tapes. One example of a rotation scheme is the Grandfather-Father-Son rotation. A good tape rotation ensures that tapes are not overused and that more than just the most recent backup is available.

6.5 Innovative Recommendations

Here are some basic recommendations for an effective backup strategy. These recommendations are generic and may not be appropriate for your system. You should evaluate these tips based on your library's needs and available resources.

6.5.1 Use Archive Logging

If possible, run your database in archive log mode. If you have a good set of archive logs, you can recover up to the last committed transaction. If you do not have a good set of archive logs, you can recover the database only to the state it was in at the time of the last full backup. This means that you will lose . . .

- All new records.
- Any modifications to existing records.
- All information about circulation transactions processed since the last backup.
- All information about acquisitions transactions processed since the last backup.

6.5.2 Take Regular Backups

Take a full backup of your database once a day, or if this is not feasible, at least once a week. If you can take your database down for long enough to do so, we recommend that you take cold backups. After each cold backup, take a full export of the database using the Data Pump tool, which is a feature of Oracle (see the section "Exporting and Importing Using Data Pump."

Note: The Data Pump export serves as another level of redundancy in the backup strategy. Thus if you are unable to recover from the full backup, you can use the export to create a new database.

6.5.3 Test Your Backup Files

Use the **dbv** utility to validate your backup data files immediately after taking the backup. If a file is not valid, you will not be able to use it to recover.

6.5.4 Retain Past Backup Files

Retain at least one backup prior to the current backup, including archive log files. If your current backup fails, you can recover from the previous backup by applying archive logs from the previous and current backup.

6.5.5 Store Multiple Copies of Backup Files

At a minimum, you should have one copy of your backup files onsite, and another copy offsite. An offsite copy is necessary to guard against data loss from a catastrophic event such as a fire.

Do NOT store the only copy of a backup on the same disk or server as your database.

6.5.6 Practice Recovering from Backup

The only way to know if you have an effective backup strategy is to recover from a backup. By practicing various recovery scenarios, you will get a good idea of the strengths and weaknesses of your backup strategy.

An additional benefit of practicing database recovery is that you will be better prepared to recover if the need arises. The more experience you have recovering from backup, the easier it will be to recover when you actually need to. We recommend that you regularly practice recovering from backup on a *non-production* database.

7. Exporting and Importing a Virtua Database Using Data Pump

As of version 10*g*, Oracle offers a Data Pump tool that lets you export and import Oracle databases. This section gives instructions for using Innovative-provided scripts to make a Data Pump export and import of a Virtua database. You can use these instructions for any version of Oracle starting with 10*g*.

Note: If you have Oracle 11*g* or higher, do NOT use the old "exp" and "imp" utilities; use only Data Pump, whose "expdp" and "impdp" utilities offer greatly increased functionality. However, if you are importing a database exported with the old "exp" command, then you may want to import using the old "imp" command.

This chapter covers the following topics:

⇒ Exporting the Database
⇒ Importing the Database

7.1 Exporting the Database

To take a Data Pump export of the database,

1. Navigate to the clasXX directory of the database you want to take the export from:

   ```
   cd /usr/vtls/clasXX
   ```

2. Create a directory to work in and name the directory "impexp":

   ```
   mkdir /usr/vtls/clasXX/impexp
   ```

3. Move into the newly created directory:

   ```
   cd /usr/vtls/clasXX/impexp
   ```

4. From **/impexp**, run **`DpExport_Database.sh`**

Note: The total time needed to export the database depends on the size of the database, but you can expect anywhere between five minutes and three hours.

The **.dat** export file will reside in the **/usr/vtls/clasXX/impexp** directory. After the script completes, the script will provide the command line parameters to use when importing the database at a later time, e.g.,

```
Finished DpExport_Database.sh Wed Aug 14
18:08:22 EDT 2013...when importing this
database use the following command line
parameters:
$EXE_DIR/DpImport_Database.sh
Database_export_20130814180456.dat chamo
```

5. Check the export log for errors.

7.2 Importing the Database

Note: Before you can import a database, you must request a BLANK database from Innovative Customer Support. This database will provide only the framework for a Virtua database. It does not contain any actual data such as patron records, transactions, or parameters.

To perform a Data Pump import into a new blank database,

1. Navigate to the clasXX directory of the blank database into which you want to import the data:

   ```
   cd /usr/vtls/clasXX
   ```

2. Create a directory to work in and name the directory "impexp":

   ```
   mkdir /usr/vtls/clasXX/impexp
   ```

3. Move into the newly created directory:

   ```
   cd /usr/vtls/clasXX/impexp
   ```

4. Move the **.dat** export file into **/impexp**.

5. Verify that the $ORACLE_SID is set to the current blank database:

   ```
   echo $ORACLE_SID
   ```

6. Verify that the blank database is running.

7. Verify that Temp space is a minimum of 2G (the TEMP tablespace has a maximum size of 2G). To do this, run the script **CheckTablespaceFileSize.sh**.

8. After the verifications, run **DpImport_Database.sh** using the parameters you received after running **DpExport_Database.sh**.

   Note: If the parameters are not available, you will need to supply the following instead:
   `DpImport_Database.sh <dumpfile to import> <chamo|iportal> [include_profiler_schema]`

   Where…
   <dumpfile to import> is the datapump.dat export file.
   <chamo|iportal> gives you the choice for your Web OPAC.
   [include_profiler_schema] must be added only if the database you exported had the Profiler schema included (and all databases as of 2012.3 do have this schema).

9. Check the import log for errors. If your Data Pump export was from an Oracle 10*g* database with a 2k block size, the DBADMIN.TAI_LOCATION trigger may not have imported correctly. In such a case, you will see an error message. To re-create the trigger, after the import is complete, run **$EXE_DIR/Cr_TAI_LOCATION.sh**.

10. To verify that all objects are present, run the **testState.sh** script:

    `$EXE_DIR/sql/migration/testState.sh`

11. If your institution uses Ad Hoc Reporting, run **create_adhoc_views.sh**.

12. The import is now complete. Note that you may need to request a new expiration key from Innovative.

8. Copying Production Data to the Test Database

This chapter provides instructions for copying a production database to a test database using a utility supported by Oracle 11g. After you complete these procedures, your **clas99** test database will be an exact copy of your production database.

This chapter covers the following topics:

⇒ Before you Begin . . .
⇒ Creating a Test Database from the Production Database

8.1 Before you Begin . . .

Before you begin these procedures, you need to obtain the following from Innovative:

- The script **99dmpXX_X**, where XX_X is the version of the database (this script comes as part of a package that includes a blank test database, which includes an **impexp** directory).
- A valid server key code for the clas99 test database (you might already have this).

Additionally, you will need extra free disk space available for the test database. To get an estimate as to the amount of disk space the exported data file will require, run the script **prod_db_stats** to generate a list of statistics about the export.

Note:

- As part of the copy process, a script will export the contents of your production database and then import this data to the **clas99** database. As this is a system-intensive process, we recommend that you wait until a period of low system activity to begin this procedure.
- It is always important to have a good backup of your database, particularly when you are working with large data transfers as is the case here. We recommend that you take a backup of your database before beginning this procedure. For information about database backups, see the chapter "Database Backup" in this guide.

8.2 Creating a Test Database from the Production Database

To create a test database (clas99) from your production database, you need to complete four steps:

1. Create a blank **clas99** database. Upon request, Innovative will copy the **99dmpXX_X** script to the **/usr/vtls** directory of your server. This script will generate the blank **clas99** database.

   Important: If you already have an existing **clas99** database, this procedure will overwrite this database with a blank one.

2. (*Optional*) Determine how much space is required to hold the export file.

3. Copy the production data to the blank clas99 database.

4. Make the database copy ready for use.

Each of these steps are described below.

8.2.1 Creating a Blank Test Database

To create a blank clas99 database,

1. Log in to your server as the **dbadmin** user.

2. Untar the file provided by Innovative in the **/usr/vtls** directory. To do this, at the prompt, type:

   ```
   ./99dmpXX_X
   ```

   Where **XX_X** is the version number of the clas99 database, for example, **12_3**.

   Note: The version number of the server scripts provided by Innovative may differ somewhat from the version number of your database. For instance, the server scripts may have 10.4 as the version number, but they would still be valid for an 11.1 database.

2. Press Enter.

   The script creates a blank clas99 database in the directory **/usr/vtls/clas99**

8.2.2 Determining the Amount of Disk Space Needed

Note: This step is optional

To determine how much disk space is required to hold the export file,

1. Log in as **dbadmin** and pick the source database.
2. Navigate to the directory **/usr/vtls/clas99/impexp**.
3. At the prompt, type: **`./prod_db_stats`**
4. Press Enter.

In the output, the line that says "`Total estimation using BLOCKS method:`" provides the estimated disk space for the exported data file.

8.2.3 Copying Data from the Production Database to the Test Database

To copy data from the production database to the blank clas99 database,

1. Make sure the source database is running, and start up the new **clas99** database using the **initvtls.ora** file in the **clas99** directory. For details, see the section "Starting and Shutting Down the Oracle Database" in this guide.
2. Log in as **dbadmin** and pick the source database. (The system prompts you for the database number.)
3. Navigate to the **/usr/vtls/clas99/impexp** directory.
4. At the prompt, type: **`nohup ./prod_db_copy &`**
5. Press Enter.

The script **prod_db_copy** exports all of the data from the production database and then imports it to the clas99 database. All diagnostic messages are printed to **nohup.out**.

Note: Depending on available system resources and the size of the database being copied, this process may take some time to complete. For details, please refer to the benchmarks that the system provides when you run the script.

6. Open the file **nohup.out** in a text editor and report any errors to Innovative Customer Services. Note that you can *ignore* the following errors:

   ERROR at line 1:
   ORA-04043: object DATADIR1 does not exist

   ORA-31684: Object type USER:"DBADMIN" already exists
   ORA-31684: Object type USER:"IPORTAL" already exists

   ORA-39083: Object type ROLE_GRANT failed to create with error:

   In addition, ignore any errors with 'VTLS_REPORT_PRIVS.' This error refers to Ad Hoc reporting. For information, see the *Ad Hoc Reporting Reference Guide*.

   At this point the database copy is complete. Complete the following tasks to make it ready to use.

8.2.4 Making the Database Copy Ready for Use

To make the database copy ready for use,

1. Log in as **dbadmin**; pick the 99 database number.

2. Copy the source database **clasXX.maps** to **/usr/vtls/clas99/clas99.maps.** To do this, type:

   ```
   $EXE_DIR/../lang/jloadmaps –e $EXE_DIR -1 $EXE_DIR/../lang
   /usr/vtls/clas99/clas99.maps
   ```

3. Type: **impexp/crprefs**

4. Type: **`impexp/resyns`**

5. Use the program **SetServerKeyCode.exe** to enter (i.e., install) the server key code for the **vtls99** database. For details, see the section "Updating the License Key" in this guide.

9. Character Mapping

Character maps are used by Virtua to input, output, and sort data. This chapter provides basic instructions for working with character mapping. For information about sort mapping, see the appendix "Using Sort Maps" in this document.

This chapter discusses the following topics:

⇒ Determining which Input and Output Maps Are Loaded
⇒ Loading Character Maps
⇒ Converting the Character Set of a Text File

9.1 Determining which Input and Output Maps Are Loaded

You can use the program **map.exe** to determine which input and output maps are loaded and stored in the database.

To determine which input and output maps are loaded,

1. At the server prompt, type: **`map.exe -s map.txt`**
2. Press Enter.

The program **map.exe** writes the list of input and output maps that are stored in the database specified by the ORACLE_SID environment variable. Additionally, if there is no sort map loaded in the database, the program writes to the specified output file the following message:

SORT MAP: No sort map is loaded. Load one ASAP! If this database is live, contact VTLS for further instructions.

If you receive this message, your database does not have a sort map loaded. Many functions of Virtua will NOT work without a sort map.

9.2 Loading Character Maps

This section provides information for loading character maps. In most cases, your character maps will be loaded at installation based on your needs, and you will not need to follow the instructions in this chapter.

There are three types of character maps that you can load to your database:

- **Input map** - For a character set, determines how each character is mapped to UTF-8.
- **Output map** - Maps UTF-8 characters to another character set.
- **Sort map** - Determines how characters are sorted in a results set.

  Note: It is best to use the script **LoadSortMaps.sh** to load a new or altered sort map to your database; see the section, "Loading Sort Maps" in this guide.

For each database you can load . . .

- One or more input maps.
- One or more output maps.
- *One* sort map.

Input and output maps are relatively small in file size, therefore it is a good idea to load any input or output maps that you think you might use.

There are two steps to loading character maps to your database:

1. Create a file that specifies which character maps you want to load.
2. Run **jloadmaps** to load the character maps you specified.

Step I: To create a file specifying which character maps you want to load to the database,

1. Log in to your server as the **dbadmin** user.

2. Using a text editor, create and open a blank text file.

3. On the first line of your new text file, type **`input`** followed by the name of each input map you want to load.

   For example, if you want to load Greek, Swiss ANSI8, and EUROPA3 input maps, type:

```
input greek swiss-ansi8 euro3
```

Tip: For a list of input map names, see the appendix "Mapping Codes" in this document.

4. Press the Enter key to go to the next line.
5. Type **output** followed by the name of each output map you want to load.

   For example, if you want to load ANSI8, Windows Latin 2, and Windows Cyrillic output maps, type:

```
output ansi8 wlatin2 wcyrillic
```

   Tip: For a list of output map names, see the appendix "Mapping Codes" in this document.

6. Press the Enter key to go to the next line.
7. Type **sort** followed by the name of the sort map you want to load.

   Note: You can load only ONE sort map to a database. When you load a sort map, it overwrites the previously loaded sort map.

   For example, if you want to load the Scandinavian sort, type:

```
sort scand
```

   Tip: For a list of sort maps, see the appendix "Mapping Codes" in this document.

8. Press the Enter key to go to the next line.
9. Type **uppercase**
10. Save and exit the file.
11. Copy the file to **/usr/vtls/virtua/r_XX_X/lang**.

Step II: To load specified character maps to the database,

Important: This step is to be performed *before* loading records to your database. If you load a new sort map to the database after you load records, your indexes will be invalidated and you may not be able to find the previously loaded records via a browse or keyword search.

1. Navigate to **/usr/vtls/virtua/r_XX_X/lang**, where XX_X is the version of Virtua you are running.

2. At the prompt, type:

   ```
   ./jloadmaps -e ../src -l . [filename]
   ```

   where **[filename]** is the name of the maps file that you created in the first section of these instructions. For example, if the name of your file is **maps.txt**, type:

   ```
   ./jloadmaps -e ../src -l . maps.txt
   ```

   The character maps you specified in the file are loaded to the database.

Note: Any sort map you choose replaces the sort map that was previously in your database. If you loaded records to your database prior to running **jloadmaps**, it is important that you re-index your keyword and browse tables (i.e., run **Re_CreateHeadingSort.sh**, **KeywordIndex.exe**, and **CallIndexForm.sh**). Until you do this, your OPAC searches will not work correctly.

9.3 Converting the Character Set of a Text File

The program **MapChars.exe** converts the text in an input file from one character set to another. To use this program, you must . . .

- Know the character set of the text file that you want to convert.
- Have loaded in your database the input map for the character set of the text file that you want to convert.
- Have loaded in your database the output map for the character set to which you want to convert the text file.

Note: The ORACLE_SID environment variable must specify the database in which the appropriate input and output maps are loaded.

To convert the character set of a text file,

1. At the server prompt, type:

   ```
   MapChars.exe [input code] [output code] 1 [input file] [output file]
   ```

   Where . . .

   - **[input code]** is the numeric code that represents the character set of the input file. For a list of input character sets and codes, see the section "Input Maps" in this document.
   - **[output code]** is the numeric code that represents the character set to which you want to convert the file. For a list of output character sets and codes, see the section "Output Maps" in this document.
   - **[input file]** is the name, and if necessary, the path of the file that you want to convert.
   - **[output file]** is the file to which you want to write the converted file. If you do not include this parameter, the program will write the converted text to **stdout**.

2. Press Enter.

10. Appendix A - Scripts and Executables

This appendix lists most of the scripts and executables available in Virtua. The majority of programs listed here are documented in one of the Virtua System Reference guides, as indicated in the *Documented In . . .* column in the table. Those programs listed here that are not explicitly documented in the Virtua guides display guidelines for their use at the command line.

Filename	Purpose	Documented in . . .
2709ToXML.exe	Converts a file of MARC 2709 records to MARCXML records.	*System Management: Cataloging User's Guide*
AddDiagnosticText.exe	Translates the diagnostic and error data generated by the **vload.exe** program from code into a human-readable language.	*Virtua Record Loading User's Guide*
AddItemCallNumberPrefix.sh	Adds a prefix to the call numbers of the specified item records.	*System Management: Cataloging User's Guide*
addItemStatus.sh	Adds an item status to the specified records.	*System Management: Cataloging User's Guide*
AddTablespaceDataFile.sh	Adds a single 64MB data file that is autoextensible to 2G for a specified tablespace.	"Adding a Data File to a Specified Tablespace" in this guide.
AnalyzeCTXBibliographicIndex.sh	Determines the percent of fragmentation that exists within the CTX_BIBLIOGRAPHIC index, which is used only for keyword searches in the Virtua client.	*System Management: OPAC User's Guide*

Filename	Purpose	Documented in . . .
authClr.sh	Removes orphaned authority headings and bibliographic IDs from the browse list.	*System Management: Cataloging User's Guide*
autoClaim.exe	Processes in batch Acquisitions and Serials claims.	*System Management: Acquisitions and Serials User's Guide*
BatchDeleteItems.exe	Deletes the item records associated with the specified list of Item-IDs.	*System Management: Cataloging User's Guide*
BatchDeleteVendors.exe	Deletes vendor records based on a file of vendor IDs.	*System Management: Acquisitions and Serials User's Guide*
blockpat.exe	Generates a list of blocked patrons and a list of block codes.	*Circulation Control/Circulation Backup System User's Guide*
CallIndexForm.sh	Indexes call numbers for the call number browse list.	*System Management: OPAC User's Guide*
CallIndexFormItemsOnly.sh	Indexes item-level call numbers for the call number browse. This script allows you to re-index item-level call numbers after running the **SetPreventBrowseOf2ndItemLevelCallNumber.sh** script without recreating the entire call number index.	*System Management: OPAC User's Guide*
callNumberKey.exe	Converts call numbers into a normalized sort form.	*System Management: Reporting User's Guide*
ChangeAllowRequest.sh	Sets the request setting in the specified records.	*System Management: Cataloging User's Guide*
ChangeItemClass.sh	Changes an item class to a different item class in the specified records.	*System Management: Cataloging User's Guide*
ChangeItemPrice.sh	Changes an item price to a different item price in the specified records.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
ChangeItemStatus.sh	Changes an item status to a different item status in the specified records.	*System Management: Cataloging User's Guide*
ChangeLoanPeriod.sh	Changes the loan period to a different loan period for the specified items.	*System Management: Cataloging User's Guide*
ChangeLocation.sh	Changes the owning location of items that you specify in a file of Item-IDs or barcodes	*System Management: Cataloging User's Guide*
ChangeLocationByBarCode.sh	Modifies the location of items that you specify in a file of barcodes.	*System Management: Cataloging User's Guide*
ChangeLocationByCallNumberRange.sh	Modifies the shelf and owning location of items within a range of item-level call numbers.	*System Management: Cataloging User's Guide*
ChangeLocationByItemId.sh	Modifies the location of items that you specify in a file of item IDs.	*System Management: Cataloging User's Guide*
ChangeLocationId.sh	Changes one location code to a new location code that does not yet exist.	*System Management: Cataloging User's Guide*
ChangeOracleUserPassword.sh	Changes the password for any Innovative-created Oracle user.	"Changing Passwords for Oracle Users" in this guide.
ChangePermAuthorityToProv.kh	Converts permanent authority records to provisional.	*System Management: Cataloging User's Guide*
ChangeReserveItemClass.sh	Changes the reserve item class to a different reserve items class for the specified items.	*System Management: Cataloging User's Guide*
ChangeShelfLocation. sh	Changes the shelving location to a different shelving location for the specified records.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
CheckAllFRBRLinks.sh	Checks all records found in the frbr_parent_child database table and verifies that they are linked to the correct type of FRBR entity (Work, Expression, or Manifestation).	*System Management: Cataloging User's Guide*
CheckCircParameters.sh	Examines your circulation parameters and reports any discrepancies.	*System Management: Circulation User's Guide*
CheckForReciprocal1xx5xxAuthorities.sh	Checks in authority records for "un-linked" 5XX cross-references (i.e., a 5XX tag *See also* heading that is not linked to a 1XX tag in an LC authority record).	*System Management: Cataloging User's Guide*
CheckOracleForBlockCorruption.sh	Scans the Oracle data files for both physical and logical block corruption. If any block corruption is detected the Oracle object and block number are reported.	"Checking Oracle for Block Corruption" in this guide.
CheckTablespaceFileSize.sh	Checks the tablespaces and their associated data files. The script provides the available free space within the tablespace and the room available for each individual data file to grow if the autoextensible parameter is set to ON.	"Checking Tablespaces and Data File Sizes" in this guide.
CheckedOutItems.sh	Outputs the barcodes of all items in the database that have been checked out.	*System Management: Circulation User's Guide*
circbackup.exe	Processes transactions recorded by the Circulation Backup System.	*Circulation Control/Circulation Backup System User's Guide*

Filename	Purpose	Documented in . . .
CircReport.exe	Generates information for Circulation reports and notices.	*System Management: Reporting User's Guide*
CleanUpRequestDirectories.ksh	Purges the contents of the **log**, **temp**, and **archive** directories in the request slip printing system.	*Chamo Administration Guide*
CombineFRBR.exe	Recombines a set of FRBR entities into a traditional MARC 21 bibliographic record.	*Virtua FRBR Implementation User's Guide*
CombineFRBRRecords.sh	Recombines a set of FRBR entities into a traditional MARC 21 bibliographic record and reloads it into the database.	*Virtua FRBR Implementation User's Guide*
ConvertMARCFileCharacterSet.exe	Converts a file of MARC records from one character set to UTF-8 or vice versa.	*System Management: Cataloging User's Guide*
ConvertMARCRecordsToRDA.exe	Provides numerous options for converting a file of MARC authority or bibliographic records to RDA.	*System Management: Cataloging User's Guide*
ConvertMARCRecordsToRDA.sh	Uses a helper script to load RDA translations into the database and then converts a file of MARC records to RDA.	*System Management: Cataloging User's Guide*
cr_CTX_Indexes.sh	Recreates the keyword indexes.	*System Management: OPAC User's Guide*

Filename	Purpose	Documented in . . .
Create024PermalinkTagInAuthorityRecords.exe	Adds permalink 024 tags to existing authority records. **Note:** Innovative's permalinks functionality can be purchased separately as an add-on to Virtua. Contact Innovative Customer Services for more information.	*System Management: Cataloging User's Guide*
Create024PermalinkTagInBibRecords.exe	Adds permalink 024 tags to existing bibliographic records. **Note:** Innovative's permalinks functionality can be purchased separately as an add-on to Virtua. Contact Innovative Customer Services for more information.	*System Management: Cataloging User's Guide*
create_adhoc_views.sh	Creates Ad Hoc views and is provided ONLY to customers who purchase the Ad Hoc Reporting product. It is run automatically at the end of every Virtua migration.	*N/A – Contact Innovative Customer Support* **Note:** Customers who purchase Ad Hoc should untar the Ad Hoc software before running the **migrate.sh** script.
create_intermedia.pl	Extracts data from the thesaurus tables and writes it to a file that can be loaded to interMedia.	*System Management: OPAC User's Guide*
CreateBibIdsFromItemIds.sh	Generates a file of Bib IDs from an input file of Item IDs.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
CreateCustomAdHocView.sh	Creates custom views that provide access to one or more tags stored in the Bibliographic_Record table.	*System Management: Reporting User's Guide*
CreateDirectoryObject.sh	Creates an Oracle Directory Object, which is a logical object linked to a physical directory on the host server.	"Creating a Directory Object" in this guide.
CreateEnvVariableScript.sh	Identifies current environment variables and generates a script to set the environment variables to the current values.	"Creating a Script to Set Environment Variables" in this guide.
CreateItemBarcodesFromItemIds.sh	Generates a file of item barcodes from an input file of Item IDs.	*System Management: Cataloging User's Guide*
CreateItemIdsFromBarcodes.sh	Generates a file of Item IDs from an input file of item barcodes.	*System Management: Cataloging User's Guide*
CreateItemIdsFromBibIds.sh	Generates a file of Item IDs from an input file of Bib IDs.	*System Management: Cataloging User's Guide*
CreateRMANOnlineBackup.sh	Performs a hot backup and creates a restore script for restoring the entire database.	"Hot Backups" in this guide.
Cr_Spfile.sh	Creates an Spfile from the current initialization parameters stored in memory and set by the text file **initvtls.ora**.	"Creating a Backup with an Spfile" in this guide.
CreateSubjectThesauri_1.sh	Identifies any problems with authority records as part of the enable multiple subject headings process.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
CreateSubjectThesauri_2.sh	As part of the enable multiple subject headings process… • Uses the file of bib-IDs generated in **CreateSubjectThesauri_1.sh** to create duplicate subject headings for subject headings in more than one thesaurus. • Creates indexes for each thesaurus.	*System Management: Cataloging User's Guide*
DecodeBibLeaderData.sh	Displays decoded Leader data for a specified Bib ID and Leader value.	*System Management: Reporting User's Guide*
Delete_Circ_Trans_Log_OFFLINE.sh	Deletes a large amount of circulation transaction log data while the Virtua system is completely *offline*.	*System Management: Circulation User's Guide*
Delete_Circ_Trans_Log_ONLINE.sh	Deletes a small number of circulation transaction log entries, generally no more than one month of data, while the Virtua system is *online* and in use.	*System Management: Circulation User's Guide*
Delete_Operation_Log.sh	Deletes entries within a specified date range from the database operation log.	*System Management: OPAC User's Guide*
DeleteExpiredPatronBlocks.sh	Deletes expired blocks (043 tags) from patron records in the database.	*System Management: Circulation User's Guide*
DeleteExpiredPatrons.sh	Generates a list of patron IDs and deletes the patron records (unless otherwise specified) based on the expiration date of the patron records	*System Management: Circulation User's Guide*
DeleteExtraCombinedFRBR.sh	Removes FRBR keyword combined records that were created erroneously as a result of a bug before Virtua 2014.2.1.2 or 2014.3.	*FRBR Implementation User's Guide*

Filename	Purpose	Documented in . . .
DeleteInactivePatrons.sh	Generates a list of patron IDs and deletes the patron records (unless otherwise specified) based on one of the following two criteria: • Last activity date of the patron record. • Length of time (in months or years) that the patron record has been inactive.	*System Management: Circulation User's Guide*
DeleteItemsByBarcode.sh	Deletes the item records associated with the specified list of item barcodes	*System Management: Cataloging User's Guide*
DeleteItemsByDueDate.sh	Deletes from the database all items that were due (and still checked-out) *before* the specified date.	*System Management: Cataloging User's Guide*
DeleteItemsByLocation.ksh	Deletes all the item records associated with the specified location code.	*System Management: Cataloging User's Guide*
DeleteItemsByStatus.sh	Deletes from the database all items that carry the specified status.	*System Management: Cataloging User's Guide*
DeleteOldFines.sh	Deletes fines and fees assessed before a given date, and outputs a file of information about the deleted fines and fees.	*System Management: Circulation User's Guide*
DeleteOldItems.sh	Deletes items from the database based on chosen criteria of due date, status, and/or owning location.	*System Management: Cataloging User's Guide*
DeleteOldPurchaseRequests.sh	Deletes purchase requests based on chosen criteria of retaining date.	*System Management: Acquisitions User's Guide*

Filename	Purpose	Documented in . . .
DeleteOrphanFRBRLinksAndBibs.sh	First deletes any orphaned FRBR links and then deletes any orphaned FRBR bibliographic records from the database.	*FRBR Implementation User's Guide*
DeletePatronPasswords.sh	Deletes unencrypted patron passwords in subfield $b of the 015 and 016 tags from all patron records that have the new encrypted version in subfield $c	*System Management: Circulation User's Guide*
DeletePatrons.sh	Deletes patron records, regardless of their outstanding associations, based on an input file of patron IDs.	*System Management: Circulation User's Guide*
DisableArchiveLogging.sh	Disables archive logging by shutting down the database, disconnecting all users, and restarting the database in non-archive log mode.	"Disabling Archive Logging" in this guide.
DisplayActiveDBConnections.sh	Displays all Virtua components and other external programs currently connected to the database.	"Identifying Active Database Connections" in this guide.
DisplayNoLoggingTables.sh	Displays any tables that have been created or modified with the NOLOGGING option, which is a special setting that can be used to speed up the process of a data load or a migration.	"Displaying Tables that Disallow Logging" in this guide.
DisplayOracleRelease.sh	Displays to the screen the current database Oracle release and edition.	"Identifying the Oracle Version" in this guide.

Filename	Purpose	Documented in . . .
DisplayParallelTables.sh	Displays any tables that have been created or modified with the PARALLEL option, which is a special setting that can be used to speed up the process of a data load or a migration.	"Displaying Tables Running in Parallel Mode" in this guide.
DisplayRedoLogSwitches.sh	Used for Oracle troubleshooting purposes only, the script displays the number of times the Oracle redo log groups switched each hour for the past 24 hours.	"Working with Redo Log Groups" in this guide.
DisplayVirtuaDBRelease.sh	Displays to the screen the release version of the Virtua database being used.	"Determining the Virtua Release" in this guide.
DocumentBatchDeleter.sh	Removes documents from the full text index.	*System Management: OPAC User's Guide*
DocumentBatchLoader.sh	Loads and indexes a batch of documents for full text searching.	*System Management: OPAC User's Guide*
DocumentLoader.sh	Loads and indexes a single document for full text searching.	*System Management: OPAC User's Guide*
DpExport_Database.sh	Takes a Data Pump export of the database.	"Exporting the Database" in this guide.
DpImport_Database.sh	Imports the database that was exported using **DpExport_Database.sh**.	"Importing the Database" in this guide.
drop_CTX_Indexes.sh	Drops all CTX_* indexes used for keyword and heading keyword indexing.	*System Management: OPAC User's Guide*

Filename	Purpose	Documented in . . .
DuplicateOptions.exe	Generates command options for duplicate test and merge functions of **vload.exe**.	*Virtua Record Loading User's Guide*
EnableArchiveLogging.sh	Enables archive logging by shutting down the database, disconnecting all users, and restarting the database in archive log mode.	"Enabling Archive Logging" in this guide.
ExportPatronBarcodeTable.sh	Exports the patron_barcode table. **Update_Patron_015b.sh** runs this script automatically.	*System Management: Circulation User's Guide*
extract_bibs_by_creation_date.sh	Extracts bibliographic records based on creation date.	*System Management: Cataloging User's Guide*
extract_deleted_bibs.pl	Extracts MARC21 (UTF-8) bibliographic records from a Virtua database that have been deleted.	*System Management: Cataloging User's Guide*
Extract_FRBR_Records.sh	Extracts all FRBR records in a database to a file that can be used to load FRBR records to another FRBR database.	*Virtua FRBR Implementation User's Guide*
extract_patrons_by_modify_date.sh	Writes a file of patron records that have been modified within a specified date range.	*System Management: Cataloging User's Guide*
extract_patrons_in_error_state.pl	Extracts patron records in a Virtua database that are in Error state.	*System Management: Cataloging User's Guide*
ExtractAuthorityRecordsUsingControlNumbers.sh	Creates a file of authority records from a file of authority control numbers.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
ExtractForDiscovery.sh	Extracts bibliographic records, either fully or incrementally, for discovery purposes and converts them to XML format.	*System Management: Cataloging User's Guide*
FindMissingFRBRLinks.sh	Identifies FRBR bibliographic records that have missing parent links.	*System Management: Cataloging User's Guide*
FixAuthConflictErrors.sh	Modifies authority records with a defined prefix to add to the 4XX tag a subfield $z that will contain the contents of the 035 tag.	*System Management: Cataloging User's Guide*
FRBRQualifyFromVTLS.sh	Identifies records in a database that qualify for becoming FRBR records.	*Virtua FRBR Implementation User's Guide*
fyRollover.exe	Rolls over information from one fiscal year to a new fiscal year.	*System Management: Acquisitions and Serials User's Guide*
GatherSchemaStats.sh	Gathers statistics on an Oracle schema.	"Gathering Statistics" in this guide.
GatherTableStats.sh	Gathers statistics on a specific table in the database.	"Gathering Statistics" in this guide.
GenerateStatsPackReport.sh	Generates a detailed report of internal database metrics regarding structures and usage.	"Reporting on Performance Problems" in this guide.
get_bibs.pl	Extracts bibliographic data from records in your database.	*System Management: Reporting User's Guide*
GetExpirationDate.exe	Returns the expiration date of the license key of the database it is run against.	"Verifying the License Key and Expiration Date" in this guide.
Get_FixedField	Returns the specified fixed field tag information from one or more records.	*System Management: Reporting User's Guide*

Filename	Purpose	Documented in . . .
GetJournalTitleIdsUsingTHL.sh	Gets a file of Auth IDs of journal titles using TITLE_HEADING_LINK	*System Management: Cataloging User's Guide*
Get_Leader	Returns the specified leader values from one or more records.	*System Management: Reporting User's Guide*
Get_Tag	Returns the specified variable field tag information from one or more records.	*System Management: Reporting User's Guide*
getURLs.sh	Reports on the records in your database that include URLs.	*System Management: Reporting User's Guide*
globalChange1.ksh	Outputs a list of authority records containing the text you specify.	*System Management: Cataloging User's Guide*
globalChange2.ksh	Finds and replaces text in a tag range of a given list of authority records.	*System Management: Cataloging User's Guide*
HandleRequestDeactivationStatus.sh	Identifies requests that have been deactivated and removes the Requested for Loan status from untrapped items that have been assigned to satisfy the requests, AND identifies requests that have been reactivated, and adds the Requested for Loan status to items that had been previously assigned to satisfy the request.	*System Management: Circulation User's Guide*
IdentifyBibTitleWithReturnChars.sh	Identifies and writes to a file a list of Bib-IDs and title tags in bibliographic records that contain a Windows return character.	*System Management: Cataloging User's Guide*
IdentifyDuplicatePatronBarcodes.sh	Identifies and writes to a file a list of duplicate patron barcodes.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
ImportRDATranslations.sh	A helper script that can be used in standalone mode to load either bibliographic or authority RDA translations into the database (see **ConvertMARCRecordsToRDA.sh**).	*System Management: Cataloging User's Guide*
Index24PermalinkTagInAuthorityRecords.exe	Indexes permalink 024 tags in authority records. **Note:** Innovative's permalinks functionality can be purchased separately as an add-on to Virtua. Contact Innovative Customer Services for more information.	*System Management: Cataloging User's Guide*
IsArchiveLoggingEnabled.sh	Tells you if archive logging is enabled in the database.	"Determining Whether Archive Logging Is Enabled" in this guide.
IsSpfileInUse.sh	Determines whether an Spfile is in use.	"Checking to see if an Spfile is Being Used" in this guide.
ItemStatusMonitor.exe	Finds any items with a status that has expired and moves it to the next status.	*System Management: Cataloging User's Guide*
itivaReport.sh	Runs **CircReport.exe** once for each notice type and generates an input file that the i-tiva Message system uses to contact patrons.	*System Management: Reporting User's Guide*
jloadmaps	Loads input, output, and sort maps to the database.	"Loading Character Maps" in this guide.
KeywordIndex.exe	Indexes bibliographic data for keyword searches.	*System Management: OPAC User's Guide*

Filename	Purpose	Documented in . . .
KeywordIndexParallelJob.exe	Keyword indexes the entire Virtua database utilizing parallel processing for **KeywordIndex.exe**.	*System Management: OPAC User's Guide*
List853Holdings.sh	Extracts the 853 tags from the holdings records in your database.	*System Management: Cataloging User's Guide*
LoadSortMap.sh	Loads a specified sort map to the database	"Loading Sort Maps" in this guide.
LoadPatronLanguageCode.sh	Loads the languages in a text file into the database to be used as possible patron languages.	*System Management: Cataloging User's Guide*
LoadRecordConverterFiles.sh	Places and stores a style sheet for MODSXML format conversions on the server.	*System Management: Cataloging User's Guide*
LoadSynonymFile.sh	Loads a text file containing sets of synonyms to the database, allowing for synonym searching.	*System Management: OPAC User's Guide*
LoadThesaurus.ksh	Loads a thesaurus file to the database.	*System Management: OPAC User's Guide*
map.exe	Outputs the list of the input and output maps that are loaded to your database.	"Determining which Input and Output Maps Are Loaded" in this guide.
MapAynAlifCharsInAuthorityRecords.sh	Updates existing Unicode U+02BF characters in authority records to the updated U+02BB characters, and updates U+02BE characters to U+02BC characters.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
MapAynAlifCharsInBibRecords.sh	Updates existing Unicode U+02BF characters in bibliographic records to the updated U+02BB characters, and updates U+02BE characters to U+02BC characters.	*System Management: Cataloging User's Guide*
MapSharpSCharacters.exe	Maps ß to double "ss" in MARC 2709 records. This executable uses an input file and generates an output file.	*System Management: Cataloging User's Guide*
MapSharpSCharacterInAuthorityRecords.sh	Maps ß to double "ss" in authority records.	*System Management: Cataloging User's Guide*
MARCBibUpdate.exe	Alters, in batch, bibliographic records in a database or in a file based on the input options you specify.	*System Management: Cataloging User's Guide*
MarcView.exe	Outputs records in an easy-to-read format.	*System Management: Cataloging User's Guide*
MoveStateRecords.exe	Processes records and attempts to move them to a new state.	*System Management: Cataloging User's Guide*
OptimizeAllIndexes.sh	Optimizes the five keyword indexes to reduce fragmentation.	*System Management: OPAC User's Guide*
OracleListener.sh	Starts, stops, or displays status, depending on the argument used.	"Working with the Oracle Listener" in this guide.
PermAuthIdsToProv.ksh	Creates a file of IDs for permanent authority records that are candidates for conversion back to provisional. Records that qualify have a 1XX tag but no cross-references (4XX, 5XX, or 7XX) and no control numbers (010 or 035 tags).	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
PopulateBibFields.exe	Populates the bibliographic_fields table in the database with bibliographic record data.	*System Management: Cataloging User's Guide*
PopulateChamoIndexQueue.sh	Populates Chamo's index queue with a file of Bib IDs, which will trigger a SOLR re-index for each Bib ID provided in the file.	*Chamo Administration Guide*
PopulateFRBRLocationFilters.sh	Enables support for location filtering for items attached to Manifestations.	*Virtua FRBR Implementation User's Guide*
PopulateCollectionGroupFilters.sh	Populates browse and heading search filters with collection code groups.	*System Management: OPAC User's Guide*
PopulateItemClassFilters.sh	Populates browse and heading search filters with item classes.	*System Management: OPAC User's Guide*
PopulatePostalCode.sh	Loads postal codes into Virtua or deletes existing postal codes.	*System Management: Cataloging User's Guide*
PopulateMarc21FormatFilter.sh	Reloads the default values for the OPAC parameter: Client Keyword Search Filters > Formats.	*System Management: OPAC User's Guide*
PopulateMissingLocationFilters.sh	Updates the Bibliographic_Location table to include any missing locations for items and holdings and removes extra data that does not exist for items or holdings.	*System Management: OPAC User's Guide*
PopulateRequestGroup.sh	Assigns a request group to each request in the database	*System Management: Circulation User's Guide*
PopulateUDCBrowse.sh	Populates the UDC Browse table so a user can Browse by UDC.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
PrintAuthoritybyState.ksh	Writes to a file the Auth-ID of each authority record in the database with a given state.	*System Management: Cataloging User's Guide*
psdriver.exe	Handles most server functions. The Virtua client connects to this program.	The chapter "Working with the psdriver.exe Program" in this guide.
Re_CreateHeadingSort.sh	Re-creates the heading_sort, heading_display, and heading_search columns for the author, title, and subject browse tables.	*System Management: OPAC User's Guide*
Re_indexForUnicodeNormalization.sh	Normalizes Unicode characters and re-indexes bibliographic and authority records.	*System Management: Cataloging User's Guide*
RecordConverter.exe	Converts a file of records in one format to a file of records in another format.	*System Management: Cataloging User's Guide*
RefreshSequences.sh	When necessary, refreshes the length of the column of the database sequences to be larger than the maximum value of the column they populate.	N/A
RefreshSyns.sh	Refreshes Virtua's Oracle users. It is run automatically as part of all migrations and can be run at any time by users to refresh the privileges of their Oracle database users and roles.	"Refreshing Permissions for Virtua Oracle Users" in this guide.
ReIndexPatrons.sh	Indexes patron names for the patron browse list.	*System Management: OPAC User's Guide*
ReIndexUserHeadings.sh	Indexes user-defined authority headings for browse lists.	*System Management: OPAC User's Guide*

Filename	Purpose	Documented in . . .
removeAddItemStatus.ksh	Removes an item status from and adds an item status to the specified records.	*System Management: Cataloging User's Guide*
RemoveBlankItemCallNum.sh	Removes blank item-level call numbers.	*System Management: Cataloging User's Guide*
RemoveCRLF.exe	Removes the <CR><LF> control character combination from a file of MARC records.	*System Management: Cataloging User's Guide*
RemoveDeactivatedRequests.sh	Removes requests that have been deactivated at least a defined number of days.	*System Management: Circulation User's Guide*
removeItemStatus.ksh	Removes an item status from the specified records.	*System Management: Cataloging User's Guide*
RemovePatronsByDeletionDate.sh	Generates a list of patron IDs and deletes the patron records (unless otherwise specified) whose deletion date has passed.	*System Management: Circulation User's Guide*
RemoveStreetDateStatus.sh	Removes street date statuses from items attached to bibliographic records with a street date (369 subfield $a) in the past.	*System Management: Circulation User's Guide*
RemoveTabCharacterFromBibRecords.exe	Removes tab spaces from a file of MARC bibliographic records and generates a report.	*System Management: Cataloging User's Guide*
RemoveXMLRecordFromBibs.sh	Removes XML data from existing bibliographic records.	*System Management: Cataloging User's Guide*
RepairUTF8InAuthorityRecords.sh	Removes invalid UTF-8 characters, including the Unicode Replacement Character U+FFFD, from authority records in the database.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
RepairUTF8InBibRecords.exe	Removes invalid UTF-8 characters, including the Unicode Replacement Character U+FFFD, from a file of MARC records.	*System Management: Cataloging User's Guide*
RepairUTF8InBibRecords.sh	Removes invalid UTF-8 characters, including the Unicode Replacement Character U+FFFD, from bibliographic records in the database.	*System Management: Cataloging User's Guide*
RepairXMLFile.exe	Removes invalid UTF-8 characters and control characters from a file of XML records.	*System Management: Cataloging User's Guide*
ReplaceSubstringInItemCallNumber.sh	Replaces a defined substring in item-level call numbers with another substring.	*System Management: Cataloging User's Guide*
ReProcess240Titles.sh	Reprocesses the 240 titles in the database after **SetTag240IsSeparateTitleFlag.sh** is run.	*Virtua Cataloging Reference Guide*
ReProcessAuths.ksh	Reprocesses all authority records and attempts to move them to the Process Immediately state.	*System Management: Cataloging User's Guide*
ReProcessBibs.ksh	Reprocesses all bibliographic records and attempts to move them to the Process Immediately state.	*System Management: Cataloging User's Guide*
ReProcessBibsToIndexVitalPid.sh	Creates a file of Bib IDs that contain tag 856 subfield $i VITAL and reprocesses the Bib IDs to index the VITAL PID in the Vital_Pid column of the Bibliographic_Record table.	*System Management: Cataloging User's Guide*

Filename	Purpose	Documented in . . .
ReProcessForPublisherHeadings.sh	Creates a file of Bib IDs that contain the 260 $b or 264 $b and reprocesses them to index the publisher headings for browse searches without updating the bibliographic records.	*System Management: Cataloging User's Guide*
ReProcessForPublisherUserHeadings.sh	Creates a file of Bib IDs that contain the 260 $b or 264 $b and reprocesses them to index the publisher headings and user-defined headings for browse searches without updating the bibliographic records.	*System Management: Cataloging User's Guide*
ReProcessFRBRWorksForSubjects.sh	Indexes the subject headings of FRBR Work records.	*System Management: Cataloging User's Guide.*
ReProcessPatrons.sh	Reprocesses all patron records and attempts to move them to the Process Immediately state.	*System Management: Cataloging User's Guide.*
RequestDaemon.sh	Prints request slips for page requests initiated in Chamo.	*Chamo Administration Guide*
RequestRefresh.exe	Reprocesses or refreshes requests whose types have been changed due to a migration or other event.	*System Management: Circulation User's Guide*
ResetChamoAdminPassword.sh	Resets the password for the Chamo Admin user to the default value of *admin*.	*Chamo Administration Guide*
ReSetTempCircCount.ksh	Resets to zero the temporary circulation count of the specified items.	*System Management: Circulation User's Guide*

Filename	Purpose	Documented in . . .
ResizeTablespaceDataFile.sh	Resizes a data file or temp file for a specified tablespace.	"Resizing a Data File within a Specified Tablespace" in this guide.
RestoreAuthorityNoteModifyDate.exe	Restores the last modified date to authority records after **ReProcess240Titles.sh** is run.	*Virtua Cataloging Reference Guide*
Restore_clas$DB_DIGITS.sh	Is automatically generated by **CreateRMANRestoreScript.sh** and restores the entire database in the event of a loss.	"Hot Backups" in this guide.
RestorePatronsToGoodStanding.sh	Changes patrons of one patron type to another patron type.	*System Management: Circulation User's Guide*
RetrieveAuthFixedTagData.sh	Produces a comma-delimited file of Auth IDs and tag data stored in fixed field tags within authority records.	*System Management: Reporting User's Guide*
RetrieveAuthLeaderData.sh	Produces a comma-delimited file of Auth IDs and tag data stored in the Leader of authority records.	*System Management: Reporting User's Guide*
RetrieveAuthVariableTagData.sh	Produces a comma-delimited file of Auth IDs and tag data stored in variable field tags within authority records.	*System Management: Reporting User's Guide*
RetrieveBibFixedTagData.sh	Produces a comma-delimited file of Bib IDs and tag data stored in fixed field tags within bibliographic records.	*System Management: Reporting User's Guide*
RetrieveBibLeaderData.sh	Produces a comma-delimited file of Bib IDs and tag data stored in the Leader of bibliographic records.	*System Management: Reporting User's Guide*

Filename	Purpose	Documented in . . .
RetrieveBibVariableTagData.sh	Produces a comma-delimited file of Bib IDs and tag data stored in variable field tags within bibliographic records.	*System Management: Reporting User's Guide*
rlint.exe	Prepares RLIN records for loading to the Virtua database.	*System Management: Cataloging User's Guide*
Schedule_FloatingMV_Refresh.sh	Enables floating collections and defines an update interval.	*System Management: Circulation User's Guide*
SetAutoAddPermalinksFlag.sh	Configures Virtua to automatically create a permalink 024 tag when a bibliographic or authority record is cataloged. **Note:** Innovative's permalinks functionality can be purchased separately as an add-on to Virtua. Contact Innovative Customer Services for more information.	*System Management: Cataloging User's Guide*
SetAutoShelfLocationReset.sh	Disables or enables the reset of shelving location when checking in items whose At Shelving Location Until date has passed.	*System Management: Circulation User's Guide*
SetConsiderAllRequestedItemsFlag.sh	Disables or enables Virtua's consideration of all requested overdue items as recalled overdue items for the sake of the Recalled Overdues report, Recalled Bills report, and the Alerts and Blocks Matrix.	*System Management: Circulation User's Guide*
SetIndexInvalidISBNFlag.sh	Indexes invalid ISBNs.	*System Management: OPAC User's Guide*

Filename	Purpose	Documented in . . .
SetMaskAndKeywordIndexBibs.sh	Masks or unmasks a file of bibliographic records and reindexes the records for keyword searching.	*System Management: Cataloging User's Guide*
SetNoNISOSerialDisplay.sh	Determines the format (NISO or not) of the display of compressed serials holdings.	*System Management: Acquisitions and Serials User's Guide*
SetPatronBarcodesAreUniqueFlag.sh	Sets a database parameter indicating that patron barcodes are unique across patron barcode types, i.e., across primary and alternate barcodes.	*System Management: Cataloging User's Guide*
SetPreventBrowseOf2ndItemLevelCallNumber.sh	Allows you to specify that Virtua index the first item-level call number, but *not* index the second item-level call number.	*System Management: OPAC User's Guide*
SetServerKeyCode.exe	Sets the license key code for your system.	"Updating the License Key" in this guide.
SetTag240IsSeparateTitleFlag.sh	Sets a flag to treat the 240 tag as a separate title—where Virtua will index the 240 tag as a uniform title authority. **ReProcess240Titles.sh** is to be run afterwards.	*Virtua Cataloging Reference Guide*
SetUpRequestsDirectories.sh	Creates the directory hierarchy for the request slip printing system.	*Chamo Administration Guide*
SetUseKeywordSortMapTableFlag.sh	Determines whether the new keyword sort map is used, which is better at sorting CJK languages.	*System Management: OPAC User's Guide*

Filename	Purpose	Documented in . . .
StopIndexingSubfieldEInCorporateHeadings.sh	Creates a file of Auth IDs and a file of Bib IDs that need to be reprocessed to eliminate the subfield $e from the indexed headings in tags 110, 410, 610, 710, and 810 for corporate authors and subjects.	*N/A (See also the Cataloging Basic Options parameter in the Virtua Profiler.)*
SwitchCurrentRedoLogGroup.sh	Used for Oracle troubleshooting purposes only, switches the current redo log group.	"Working with Redo Log Groups" in this guide.
testState.sh	After a database import, verifies that all objects are present.	"Importing the Database" in this guide.
ToMARC21.exe	Converts UNIMARC authority records to MARC 21 format.	*System Management: Cataloging User's Guide*
UnlockMARCRecord.sh	Unlocks a MARC record that is no longer being edited.	*System Management: Cataloging User's Guide*
unlockProfilerAccount.sh	Unlocks Oracle user accounts.	*Virtua Profiler/Getting Started Primer*
UpdateDueDate.sh	Updates the due dates of items based on a list of item IDs, original due date, and/or location.	*System Management: Circulation User's Guide.*
UpdateOverdueFines.exe	Calculates overdue fines for overdue items that have NOT yet been checked in.	*System Management: Circulation User's Guide*
UpdatePatronTypeByAge.sh	Changes a patron's type if the patron meets the age threshold specified in the Patron Types parameter in the Virtua Profiler.	*System Management: Circulation User's Guide*

Filename	Purpose	Documented in . . .
update_circ_transaction_log.pl	Deletes identifying patron information (patron IDs and patron barcodes) from the circulation transaction log entries.	*System Management: Circulation User's Guide*
Update_Patron_015b.sh	Adds or updates tag 015, subfield $b for the patrons specified in a file of patron IDs.	*System Management: Circulation User's Guide*
vload.exe	Loads records to the database, supporting a variety of load options.	*Virtua Record Loading User's Guide*
WhichSortMapsLoaded.sh	Prints the sort order used in your database.	*System Management: OPAC User's Guide*
write2709.exe	Extracts records from the database.	*System Management: Cataloging User's Guide*
WriteAuthIdsFile.ksh	Extracts to a file a list of authority IDs for the authority records in your database.	*System Management: Cataloging User's Guide*
WriteBibIdsFile.ksh	Extracts to a file a list of bibliographic IDs for the bibliographic records in your database.	*System Management: Cataloging User's Guide*
WriteHoldingIdsFile.ksh	Extracts to a file a list of holdings IDs for the holdings records in your database.	*System Management: Cataloging User's Guide*
WriteItemIdsFile.ksh	Extracts to a file a list of item IDs for the item records in your database.	*System Management: Cataloging User's Guide*
WriteMaskedBibIdsFile.sh	Extracts to a file a list of bibliographic IDs for the masked bibliographic records in your database.	*System Management: Cataloging User's Guide*
WritePatronIdsFile.ksh	Writes to a file the patron ID (001 tag) of every patron record in the database.	*System Management: Circulation User's Guide*

Filename	Purpose	Documented in . . .
WritePermAuthIds.ksh	Extracts to a file a list of authority IDs for the *permanent* authority records in your database.	*System Management: Cataloging User's Guide*
WritePermAuthFile.ksh	Extracts to a file a list of *permanent* authority records in your database.	*System Management: Cataloging User's Guide*
WritePermSubjectIds.ksh	Extracts to a file a list of authority IDs for the *permanent* subject authority records in your database.	*System Management: Cataloging User's Guide*
XMLToMARC.exe	Converts a file of MARC XML records to a file of MARC 2709 records.	*System Management: Cataloging User's Guide*

11. Appendix B - Environment Variables Used by Virtua

This appendix describes the environment variables that are used by Virtua. In many cases, Innovative will have set these variables for you when they set up your database, but you may need to set them yourself or modify them at some point.

Here is the list of environment variables used by Virtua. The seven variables listed first are crucial to the proper operation of your system. These are marked with an asterisk.

- EXE_DIR *
- ORACLE_HOME*
- ORACLE_SID*
- NLS_LANG*
- PATH*
- VIRTUA_USER*
- VIRTUA_PASSWORD*
- V_LOG_LEVEL
- V_LOG_TYPE
- V_LOG_TYPE_PARAM
- VTLS_TEMP
- IIRS_SYS
- EDIFACT_PATH
- EDIFACT_NEW_LINE
- CLAIM_BATCH_LEVEL
- CLAIM_BATCH_TYPE
- ADD_AAP_LINKS_ONLY
- LC_TIME
- LC_ALL (*for use with Solaris servers*)
- KEYWORD_INDEXING_OFF
- HEADING_INDEXING_OFF

11.1 EXE_DIR

Environment Variable: EXE_DIR

Purpose: Determines the path to the directory in which Virtua scripts and executables are stored. You can run scripts and executables in this directory without specifying the directory in your command as long as it is in the $PATH.

Additionally, you can use this environment variable as part of a path as long as the directory you are indicating is in the same directory as that specified by the EXE_DIR environment variable; for example: **`$EXE_DIR/../sql`**

Sample Value: /usr/vtls/virtua/r_44_0/src

11.2 ORACLE_HOME

Environment Variable: ORACLE_HOME

Purpose: Specifies the path to the Oracle directory.

The **.profile** script, which runs automatically on the server and configures the environment that Virtua needs, contains an entry that points Virtua to your Oracle installation. You need to edit that path.

To set the ORACLE_HOME value,

- Log in to your server as **dbadmin**.
- In the **/usr/vtls** directory, open the **.profile** script for editing.
- Modify the ORACLE_HOME value so that it matches the entry below:

```
export ORACLE_HOME=/usr/vtls/virtua/ora11
```

- Save your changes.

11.3 ORACLE_SID

Environment Variable: ORACLE_SID

Purpose: Specifies the Oracle System Identifier (SID) of the database that the system is using. The database specified by this environment variable is used when you access SQL*Plus or when a script or executable attempts to modify or retrieve data.

It is important that this environment variable is set correctly *for each database* at all times.

Sample Value: vtls01

11.4 NLS_LANG

Important: The character set portion of the NLS_LANG environment variable must be set to AL32UTF8. If this value is not set correctly, any data processed by Virtua will be corrupted.

Environment Variable: NLS_LANG

Purpose: Determines the character set used by Oracle and consists of three parts: language, territory, and character set.

Enter the NLS_LANG variable in the following format:

[language]_[territory].[Character set]

Value: AMERICAN_AMERICA.AL32UTF8

11.5 PATH

Environment Variable: PATH

Purpose: Specifies a colon-separated list of directories. When you execute a command, the shell searches through each of these directories in order, one by one, until it finds a directory where the executable exists. The first directory in the $PATH should be the $EXE_DIR followed by $ORACLE_HOME/bin and then by other Virtua and standard OS paths such as /usr/bin.

Example:
/usr/vtls/virtua/r_2014_1/src:/usr/vtls/virtua/ora11/bin:/usr/vtls/virtua/perl11/bin:/usr/vtls/virtua/mutt:/usr/sbin:/usr/vtls/virtua/apache/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:

11.6 VIRTUA_USER

Environment Variable: VIRTUA_USER

Usage: Specifies the password-authenticated Oracle user.

Value: dbadmin

11.7 VIRTUA_PASSWORD

Environment Variable: VIRTUA_PASSWORD

Purpose: Specifies the password of the authenticated **dbadmin** Oracle user.

Value: Must match **dbadmin**'s Oracle password.

11.8 V_LOG_LEVEL

Environment Variable: V_LOG_LEVEL

Purpose: Determines the type of information that is logged by the Virtua software (e.g., **psdriver.exe**). For information about logging server output, see the section "Logging Server Output" in this document.

Values: You can assign any of the following values for this environment variable:

- **1** - Log debug messages.
- **2** - Log information messages.
- **3** - Log warning messages.

- **4** - Log error messages.
- **5** - Log critical messages

If this variable is not set, the default log level is **2**.

11.9 V_LOG_TYPE

Environment Variable: V_LOG_TYPE

Purpose: Specifies the destination of log messages. You can choose to output log messages to standard error (**stderr**), syslog, or a file. If you do not set this environment variable, Virtua will write log messages to standard error. If you choose to write messages to a file, you must choose a filename by setting the environment variable V_LOG_TYPE_PARAM. For information about logging server output, see the section "Logging Server Output" in this document.

Values: You can assign any of the following values for this environment variable:

- **1** - Log messages to standard error (**stderr**).
- **2** - Log messages to syslog.
- **3** - Log messages to a file specified by the V_LOG_TYPE_PARAM environment variable.

11.10 V_LOG_TYPE_PARAM

Environment Variable: V_LOG_TYPE_PARAM

Purpose: Specifies the path and filename of the file to which Virtua writes log messages. This variable is used if you set the environment variable V_LOG_TYPE to **3**. **Tip:** The directory you specify must already exist on your server.

Sample Value: /usr/vtls/virtua/v_log.log

11.11 VTLS_TEMP

Environment Variable: VTLS_TEMP

Purpose: Determines the directory to which some Virtua programs write files. This environment variable is not used by all Virtua programs. If this environment variable is used by a program, it is mentioned in the associated documentation. An example of a program that uses the VTLS_TEMP environment variable is **vload.exe**.

Sample Value: /usr/vtls/virtua/temp_files

11.12 IIRS_SYS

Environment Variable: IIRS_SYS

Purpose: Determines the location of files that are used for Thai parsing. If your database includes Thai characters, this environment variable should always be set to the same value as the EXE_DIR environment variable.

Sample Value: /usr/vtls/virtua/r_44_0/src

11.13 EDIFACT_NEW_LINE

Environment Variable: EDIFACT_NEW_LINE

Purpose: Determines whether new line characters are included in the EDIFACT messages. The new line character creates line breaks in the text.

Values: You can assign any of the following values for this environment variable:

- **0** - New line characters are NOT included (all text is written to the same line).
- **1** - New line characters are included (the text includes line breaks).

Note: If you do not set this environment variable, new line characters are included with EDIFACT messages, unless they are for the vendor Yankee Book Peddler. Messages for this vendor will not include new line characters.

11.14 CLAIM_BATCH_LEVEL

Environment Variable: CLAIM_BATCH_LEVEL

Purpose: Determines the type of information that is logged by the **autoClaim** program. For information about running **autoClaim**, see the *System Management: Acquisitions and Serials User's Guide*.

Values: You can assign any of the following values for this environment variable:

- **1** - Log debug messages.
- **2** - Log information messages.
- **3** - Log warning messages.
- **4** - Log error messages.
- **5** - Log critical messages

11.15 CLAIM_BATCH_TYPE

Environment Variable: CLAIM_BATCH_TYPE

Purpose: Specifies the destination of the log messages written by **autoClaim**. You can choose to output log messages to standard error (**stderr**), syslog, or a file (**autoClaim.log**). If you do not set this environment variable, Virtua will write log messages to **autoClaim.log**. For information about running **autoClaim**, see the *Virtua System Management: Acquisitions and Serials User's Guide*.

Values: You can assign any of the following values for this environment variable:

- **1** - Log messages to standard error (**stderr**).
- **2** - Log messages syslog.
- **3** - Log messages to **autoClaim.log**.

11.16 ADD_AAP_LINKS_ONLY

Environment Variable: ADD_AAP_LINKS_ONLY

Purpose: Specifies whether or not you want to index *only* additional access points when running the script **ReProcessBibs.ksh**. For additional information on running this script, see the *Virtua System Management: Cataloging User's Guide*.

Values: You can assign any of the following values for this environment variable:

- **N** - Perform all indexing when running **ReProcessBibs.ksh**, including AAP, keyword, and call number indexing.
- **Y** - Index *only* additional access points when running **ReProcessBibs.ksh**. This means that records will NOT be processed for keyword and call number indexing.

11.17 LC_TIME

Environment Variable: LC_TIME

Purpose: Specifies the date format (an aspect of the UNIX locale settings) that the server uses for the 1) availability notice (**availabilitynotice.utf**), 2) in the circbackup report file, and 3) when a due date is sent via SIP (3mdriver.exe).

Sample Value: en_GB, which causes the date to be in the format dd-mm-yy.

11.18 LC_ALL (Locale Environment Variable)

Environment Variable: LC_ALL

Purpose: For Sun Solaris servers, specifies the locale environment to be used by the system. The locale determines which languages are supported and how information is displayed on the server. If you run Virtua on the Solaris platform and do not set this environment variable, you may have problems viewing the contents of files and other raw data on the server that contain non-Latin scripts such as Hebrew and Cyrillic.

Note: Setting this environment variable has no effect on the display of characters in the Virtua client or InfoStation.

Values: Solaris supports 123 locales. For details, see the following URL: http://docs.oracle.com/cd/E19455-01/806-0169/6j9hsml3c/index.html

- For use with Virtua, use a UTF-8 locale such as **en_US.UTF-8**.

  Hint: To get a list of the available locales, type: `locale -a`

Sample Usage: export LC_ALL=en_US.UTF-8

Hint: Make sure that other LC categories such as LC_TIME are either not set or set to the same localename value.

11.19 KEYWORD_INDEXING_OFF

Environment Variable: KEYWORD_INDEXING_OFF

Purpose: Toggles keyword indexing on and off

Usage:.

- To disable keyword indexing: export KEYWORD_INDEXING_OFF=y
- To re-enable keyword indexing: unset KEYWORD_INDEXING_OFF

11.20 HEADING_INDEXING_OFF

Environment Variable: HEADING_INDEXING_OFF

Purpose: Toggles heading indexing on and off for records loaded

Usage:.

- To disable keyword indexing: export HEADING_INDEXING_OFF=y
- To re-enable keyword indexing: unset HEADING_INDEXING_OFF

12. Appendix C - Mapping Codes

This appendix lists the map codes for:

- Sort maps - Mappings from the internal character set to the sort character set.
- Input maps - Mappings from the external character set to the internal character set
- Output maps - Mappings from the internal character set to the external character set.

12.1 Sort Maps

Name	Description
sort2	UTF-8 to sort2
euro3	UTF-8 to sort for EUROPA3
sort1	ANSI sort 1 for older VTLS libraries
impr	ANSI improved sort2
mexico	ANSI sort for Mexico
swiss	ANSI sort for Switzerland
latvia	ANSI sort for Latvia
scand	ISO sort for Scandinavia
catlun	ISO sort for Catalan
catlunUK	ISO sort for Catalan based on UK
polish	ISO sort for Poland
port	ISO sort for Portugal
turk	ISO sort for Turkey
slov	ISO sort for Slovakia
german	ISO sort for Germany
sort2-intl	Extends Sort2. Characters from other scripts are copied in the output without change.
greek-intl	Sort2 + Greek Sort + copy characters in other scripts
chinese-sort-tsn	Sort2 + Chinese Sort by TSN + copy characters in other scripts
vietnamese	Sort for Vietnamese
inuktitut	Sort for Inuktitut
chinese-radical-stroke	Sort2 + Chinese Sort by radical + stroke count + copy characters in other scripts

Name	Description
chinese-stroke-radical	Sort2 + Chinese Sort by stroke count + radical + copy characters in other scripts
swiss-rero	The sort mapping table that RERO uses.

For information about how sort maps are used in Virtua, see the appendix "Sort Mapping" in this document.

Note: In addition to the maps listed in this table, the **lang/sort** directory on your server contains the following files, which are used to generate the **sort2-intl** mapping tables:

- sort2-arabic.mu
- sort2-cyrillic.mu
- sort2-greek.mu
- sort2-latin.mu
- sort2-intl_3.m4
- map-controls.md
- skip-modifiers.md
- skip-symbols.md

All of these files are used as part of the **sort2-intl** map. Do not choose any one of them as the sort map for your system. Instead, if you want to use the mappings in these files, choose **sort2-intl**.

12.2 Input Maps

Note: Regardless of the input mapping Source Code being used, the Target Code is always 2, i.e., Unicode using the UTF-8 transformation format, which is the character set of the Virtua database.

Source Code	Name	Description
10	usmarc	MARC21 to UTF-8
11	ansi8	ANSI extended for UCL + ANSI 8 + ANSI Z39.47 to UTF-8
12	euro3	EUROPA 3 to UTF-8
13	wlatin1	Windows Latin1 to UTF-8
14	pc8	PC-8 to UTF-8
15	wala	Windows ALA to UTF-8

Source Code	Name	Description
16	warab	Windows Arabic to UTF-8
17	whebrew	Windows Hebrew to UTF-8
18	wcyrillic	Windows Cyrillic to UTF-8
19	wlatin2	Windows Latin 2 to UTF-8
20	iso6937-2	ISO 6937/2 to UTF-8
21	cp850	Microsoft CP 850 to UTF-8
22	ubd-brunei	ISO6937/2 + Arabic to UTF-8
23	greek	Greek to UTF-8
24	big5-tamkang	Big5 (the variant used by Tamkang University) to UTF-8
25	ansi8+hebrew	ANSI-8 + Hebrew to UTF-8. This is used for libraries that have ANSI Z39.47 for the Latin character sets and also have the Hebrew script
26	utf8	This special script (originally created for a Swedish customer) is used for mapping UTF-8 character sets, where some diacritics are coded as separate, to the Virtua database UTF-8 character set, where those diacritics are combined.
28	swiss-ansi8	Swiss ANSI8 to UTF-8.
30	gbk	GBK to UTF-8.
31	tis620	TIS620 Classic Thai to UTF-8.
32	ISO-5426	International Serials Data System (ISDS) interchange character set
33	cccii	CCCII to UTF8
34	gb18030	GB 18030 to UTF8
35	big5-HKSCS	big5-HKSCS to UTF8

12.3 Output Maps

Note: Regardless of the output mapping Target Code being used, the Source Code is always 2, i.e., Unicode using the UTF-8 transformation format, which is the character set of the Virtua database. So if you should specify 2 as the Target Code in your output mapping, no data will be mapped since the Source Code and the Target Code will be the same.

Target Code	Name	Description
10	usmarc	UTF-8 to MARC-8
11	ansi8	UTF-8 to ANSI8
12	euro3	UTF-8 to Europa3
13	wlatin1	UTF-8 to Windows Latin 1
15	wala	UTF-8 to Windows ALA
16	warab	UTF-8 to Windows Arabic
17	whebrew	UTF-8 to Windows Hebrew
18	wcyrillic	UTF-8 to Windows Cyrillic
19	wlatin2	UTF-8 to Windows Latin 2
20	iso6937-2	UTF-8 to 6937/2
24	big5-tamkang	UTF-8 to Big5 (the variant used by Tamkang University)
27	wincat	UTF-8 to US Windows (Cataloging version)
28	swiss-ansi8	UTF-8 to Swiss ANSI8.
29	wincat-east	UTF-8 to East European Windows (Cataloging version)
31	tis620	UTF-8 to Classic TIS620
32	ISDS	UTF-8 to ISDS interchange (ISO-5426)
33	cccii	UTF8 to CCCII
34	gb18030	UTF8 to GB 18030
35	big5-HKSCS	UTF8 to big5-HKSCS

13. Appendix D - Character Sorting and Mapping

This appendix explains the concepts behind character sorting and character mapping in the Virtua system. This appendix covers the following topics:

⇒ Introduction to Hexadecimal Sorting
⇒ Understanding Hexadecimal Numbers
⇒ Determining the Sort Order of a Character
⇒ Using Sort Maps

Note: See the *System Management: OPAC Reference Guide* for information about applying sort maps to keyword indexes.

13.1 Introduction to Hexadecimal Sorting

Each character in a character set has an associated hexadecimal number that can be used to determine the character's sort order. When Virtua displays a sorted results set such as the one on the List of Titles screen, the results are sorted based on this number. For example, the letter **C** has a hexadecimal number of 0043. By default, it will sort between the characters with hexadecimal numbers of 0042 (**B**) and 0044 (**D**).

You can determine the sort order of a character by comparing the character's hexadecimal number with the hexadecimal number of other characters. For a brief explanation of hexadecimal numbering as it pertains to character sorting, see the section "Understanding Hexadecimal Numbers" in this document.

While many characters are sorted by their hexadecimal number, this method of sorting will not work in some instances. For example, uppercase letters in the Basic Latin character set have hexadecimal codes ranging from 0041 (**A**) to 005A (**Z**). Lowercase letters have hexadecimal codes ranging from 0061 (**a**) to 007A (**z**). If you sort these characters by their hexadecimal number, the uppercase letter **Z** sorts before the lowercase letter **a**.

For these characters to sort logically, the lowercase letter needs to be mapped to the uppercase letter, meaning that **a** sorts the same as **A**. Virtua uses a sort map to determine which characters are used for sorting. You can determine the sort order of characters in your database by using the sort map files provided with your installation. For information on using sort maps to determine a character's sort order, see the section "Using Sort Maps" in this document.

Note: In addition to dictating the sort order of a results set, sort maps determine which characters match the characters in a search term. For example, the search term **MAGINOT** will return results with **Maginot** in an indexed field. This result is possible because the lowercase letters are mapped to uppercase letters, thereby making your search term case-insensitive. Sort maps can also help you find terms with diacritics. For example, you can find results with **résumé** in an indexed field by searching for **resume** if, in your sort map, **é** is mapped to **e**.

13.2 Understanding Hexadecimal Numbers

To analyze sort maps, you need to understand the hexadecimal numbering system. Specifically, you need to be able to compare two hexadecimal numbers to determine which number is greater.

Hexadecimal numbering is a base-16 system, meaning that there are 16 possible symbols for each digit in a number. Conventional numbering systems are base-10.

The following sections compare the base-16 hexadecimal system with the base-10 decimal system with which you are familiar.

13.2.1 Counting

The standard base-10 decimal numbering system uses the following numbering scheme:

First digit	0
Second digit	1
Third digit	2
Fourth digit	3
Fifth digit	4

Sixth digit	5
Seventh digit	6
Eighth digit	7
Ninth digit	8
Tenth digit	9

The highest single-digit value in a base-10 system is **9**. If you count to the next number after nine, you must replace **9** with a **0** and add **1** to the next column (*Figure 13-1*).

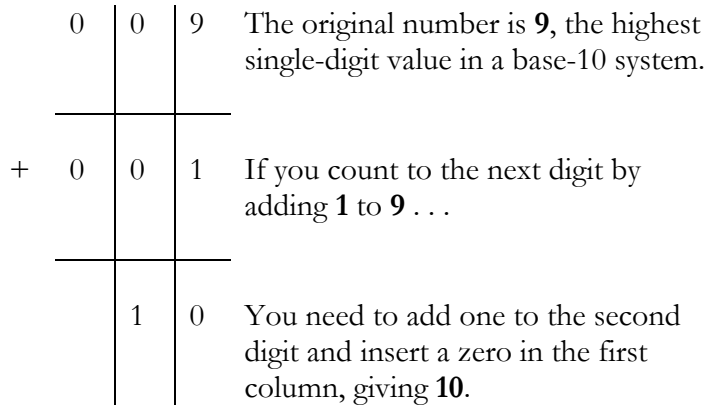


Figure 13-1 Counting in a Base-10 System

The base-16 hexadecimal system uses the following numbering scheme:

First digit	0
Second digit	1
Third digit	2
Fourth digit	3
Fifth digit	4
Sixth digit	5
Seventh digit	6
Eighth digit	7
Ninth digit	8
Tenth digit	9
Eleventh digit	A
Twelfth digit	B
Thirteenth digit	C
Fourteenth digit	D
Fifteenth digit	E
Sixteenth digit	F

The highest single-digit value in a base-16 system is **F**. If you count to the next number after **F**, you must replace **F** with a **0** and add **1** to the next column (*Figure 13-2*).

	0	0	F	The original number is **F**, the highest single-digit value in a base-16 system
+	0	0	1	If you count to the next digit by adding **1** to **F** . . .
	0	1	0	You need to add one to the second digit and insert a zero in the first column, giving **10**.

Figure 13-2. Counting in a Base-16 System

In this case, do not think of **10** as ten. **10** in base-16 hexadecimal format is actually the equivalent of the base-10 decimal **16**. Consider each digit as an individual symbol.

13.2.2 Comparing

Compare base-16 numbers in the same manner as base-10 numbers. For the number you want to compare, you need to ask the following questions:

1. **Does one number have more digits than the other?** - If one number has more digits than the other, that number is the larger number. If the number of digits is the same, proceed to the next step.
2. **Is the first digit of one number greater than the first digit of the other number?** - If the first digit of one number is greater than the first digit of the other number, that number is the larger number. If the digits are equal, do the same test on the next digit until a digit in one number is larger than the digit in the same place of the other number.

For example,

In a base-10 system . . .

200 *is less than* **1000**

400 *is greater than* **399**

5870 *is greater than* **5809**

You can use the same system to compare two base-16 numbers

For example,

1A6 *is greater than* **E9**	In this example, the first number has more digits than the second number, therefore, the first number is larger than the second.
C55 *is less than* **D00**	In this example, the first digit of the second number is greater than the first digit of the first number. The second number is larger.
2DD0 *is greater than* **2DC9**	In this example, the first two digits of the first and second numbers are equal, but the third digit of the first number is greater than the third digit of the second number. The first number is greater than the second number.

13.2.3 Determining the Character Sort Order Using Hexadecimal Numbers

As discussed above, each character has an associated hexadecimal number. This number determines the sort order of the character in relation to other characters. Generally, *lower* hexadecimal number are sorted at the top of a results set.

In a sort map, the hexadecimal number can appear in two forms:

- **ASCII Format** - Used to represent the ASCII range of characters (00 to 7F). Codes for characters in the ASCII range appear in the following format:

  0x##

  . . . where ## is the character's two-digit hexadecimal number.

  For example, in a sort map, the code for the letter **F** in ASCII format is **0x46**. The Unicode UTF-16 number for this character is **0046**.

Note: The UTF-16 format is used in this case for comparison with the Unicode charts. Virtua does NOT use UTF-16 internally. Virtua uses the UTF-8 encoding of Unicode.

- **Unicode Format** - Used to represent values not in the ASCII range (i.e., all values above 7F). Codes for characters in the Unicode® range appear in the following format:

  U+0x####

  . . . where #### is the character's four-digit hexadecimal number.

  For example, in a sort map, the code for **æ** is **U+0x00E6**. The hexadecimal number for this character is **00E6**.

  Note: The UTF-16 format is used in this case for comparison with the Unicode charts. Virtua does NOT use UTF-16 internally. Virtua uses the UTF-8 encoding of Unicode.

Characters mapped to ASCII codes always sort before characters mapped to Unicode characters. Within each format, you can compare characters by examining their Unicode UTF-16 hexadecimal number. For example, if you want to determine the sort order of characters mapping to the following values:

0x4D
U+0x03B2
0x4C
U+0x00C7

- 0x4D and 0x4C are in the ASCII range and will be sorted before the Unicode characters U+0x03B2 and U+0x00C7.
- The Unicode UTF-16 hexadecimal codes for 0x4D and 0x4C are 004D and 004C. Since 004C is a lower hexadecimal number than 004D, 004C sorts before 004D.
- The Unicode UTF-16 hexadecimal codes for U+0x03B2 and U+0x00C7 are 03B2 and 00C7. Since 00C7 is a lower hexadecimal number than 03B2, 00C7 sorts before 03B2.

Given the above comparisons, the sort order of the four characters is as follows:

0x4C
0x4D
U+0x00C7
U+0x03B2

For information on finding the hexadecimal number for a character, see the section "Using Sort Maps" in this document.

Note: Listings in a sort map that do NOT begin with the prefix **U+0x** and have a hexadecimal value above **007F** are using the UTF-8 encoding representation of a character. You cannot compare the hexadecimal numbers with UTF-8 encoding with numbers in ASCII and Unicode UTF-16 format. For information about converting these numbers to Unicode UTF-16 format, see *The Unicode Standard, Version 3.0* available at *www.unicode.org*.

13.3 Determining the Sort Order of a Character

There are three steps you need to take to determine the sort order of a character.

1. Find the hexadecimal number of the character. If you do not know this number, you can look it up using code charts at *www.unicode.org*.
2. Find the hexadecimal number of the character in the sort map used in your database.
   - If you find the character's hexadecimal number, the sort map lists the code of the character to which the character is mapped for sorting.
   - If you do not find the hexadecimal number, the character is either ignored or it is sorted by its hexadecimal number, depending on the default mapping of the sort map.
3. If the character is mapped to another character's hexadecimal number, you can view that character by looking it up in the code charts at *www.unicode.org*.

Each of these steps is discussed in detail in the following section.

13.4 Using Sort Maps

Virtua sort maps determine how characters are sorted in a results set on the List of Titles and Browse screens in the Virtua client. Using the hexadecimal code for each character, the sort map determines how each character sorts in a results list.

Note:
- There is only ONE sort map defined for each database.
- Sort maps are loaded using **LoadSortMap.sh**; see the section "Loading Sort Maps" for information about this script.

The following sort maps are available to load to your database:

- Catalan (catlun_3.mu)
- UK Catalan (catlunUK_3.mu)
- Chinese (chinese-sort-tsn_3.mu)
- Europa 3 (euro3_3.mu)
- German (german_3.mu)
- Greek (greek-intl_3.mu)
- Improved SORT2 (impr_3.mu)
- Inuktitut (inuktitut_3.mu)
- Latvian (latvia_3.mu)
- Polish (polish_3.mu)
- Portuguese (port_3.mu)
- Scandinavian (scand_3.mu)
- Slovakian (slov_3.mu)
- SORT1 (sort1_3.mu)
- SORT2 International (sort2-intl_3.mu)
- SORT2 (sort2_3.mu)
- Spanish - Mexican (mexico_3.mu)
- Swiss (swiss_3.mu)
- Swiss-Rero
- Thai (thai_3.mu)
- Turkish (turk_3.mu)
- Vietnamese (vietnamese_3.mu)

You can find these files in the **lang/sort/** directory. For information about the directory structure of the default Virtua installation, see the section "Directory Structure of the Virtua Server Installation" in this document.

Note: Although you use ***.mu** files to determine the sort order of characters, ***.md** files are actually loaded to the database to map characters. For example, for the German sort map, you would use the easier-to-read **german.mu** file to determine how characters are sorted. However, you would load the **german.md** file to the database to actually choose the German sort map.

In Figure 13-3, we display an excerpt from the Polish sort map file (**polish.mu**).

```
U+0x0309 -> SKIP    # 0xE0
U+0x1ea2 -> PUT 0x41    # 0xE0 0x41
U+0x1ea3 -> PUT 0x41    # 0xE0 0x61
U+0x1eba -> PUT 0x47    # 0xE0 0x45
U+0x1ebb -> PUT 0x47    # 0xE0 0x65
U+0x1ee6 -> PUT 0x62    # 0xE0 0x55
U+0x1ee7 -> PUT 0x62    # 0xE0 0x75

U+0x0300 -> SKIP    # 0xE1
U+0x00c0 -> PUT 0x41    # 0xE1 0x41
U+0x00E0 -> PUT 0x41    # 0xE1 0x61
U+0x00c8 -> PUT 0x47    # 0xE1 0x45
U+0x00e8 -> PUT 0x47    # new
U+0x00cc -> PUT 0x4c    # 0xE1 0x49
U+0x00ec -> PUT 0x4c    # new
U+0x00d2 -> PUT 0x54    # 0xE1 0x4f
U+0x00f2 -> PUT 0x54    # new
U+0x00d9 -> PUT 0x62    # 0xE1 0x55
U+0x00f9 -> PUT 0x62    # 0xE1 0x75
U+0x1ec1 -> PUT 0x45    # 0xE1 0xE3 0x65
U+0x1ed3 -> PUT 0x4f    # 0xE1 0xE3 0x6F
```

Figure 13-3. Sort Map - Polish.mu

13.4.1 Understanding Sort Maps

Each line in the sort map file consists of the following parts:

- **Character code** - The hexadecimal number for the character plus a prefix.
- **Separator** (**->**) - Separates the hexadecimal number from the mapping action.
- **Mapping** - Specifies how the character is mapped.
- **Comment (Optional)** - Notes about the mapping. Comments do not have an effect on mapping and can generally be ignored.

Character Code	Separator	Mapping	Comment
U+0x00E0	->	PUT 0x41	# 0xE1 0x61

All characters listed in a sort map are used in the sort according to the mapping definition. In the above example, the Unicode character U+0x00E0 (**à**) is sorted using the hexadecimal value 0041, which is the hexadecimal number for the ASCII character **A**.

Characters not defined in the sort map use the default mapping. For information on determining the default mapping for a sort map, see the section "Default Mapping" in this document.

13.4.1.1 Character Code

The character code in a sort map consists of two parts:

1. **Prefix** - The first characters in the code before the hexadecimal number. In an ASCII format character code, the prefix is **0x**. In a Unicode format character code, the prefix is **U+0x**. The prefix is not used in determining the sort order of a character.
2. **Hexadecimal number** - The unique hexadecimal identifier for the character. The hexadecimal number is located after the **x** in the character code. For example, in the character code U+0x00D2, the hexadecimal number is 00D2. You can use this number to find the mapping for a character.

Use the hexadecimal number to search for a character in the sort map.

13.4.1.2 Mapping

Located after the separator (**->**), the mapping determines how a character is mapped. There are two possible mapping values:

- **SKIP** - The character is not used in the sort. Below is an example of the mapping for a skipped character:
  U+0x0300 -> SKIP
- **PUT** [hexadecimal code] - The character is mapped to the specified hexadecimal code. This code can be either the code for a different character or the code for the character itself. This is the code that determines how the character is sorted. Below is an example of a character mapped to another character and a character mapped to itself:

  Character mapped to another character:
  U+0x1EC1 -> PUT 0x45

  Character mapped to itself:
  U+0x0023 -> PUT 0x23

13.4.1.3 Default Mapping

Each sort map includes a default sort definition. The default definition specifies how characters without explicitly defined mappings are mapped. All sort maps includes *one* of two default definitions:

- **. -> SKIP** - Any character not explicitly mapped in the sort map is ignored for sorting purposes.
- **. -> PUT .** - Any character not explicitly mapped in the sort map is mapped to itself, meaning that the character is sorted using its own hexadecimal number.

13.4.1.4 Finding the Hexadecimal Number for a Character

Every Unicode character has an associated hexadecimal number. You can find the hexadecimal number for a character using code charts such as those available at *www.unicode.org*.

To use a code chart to find the hexadecimal number for a character,

1. Find the code chart for your character set, for example, Hebrew. For a list of Unicode code charts, go to *www.unicode.org/charts*.

2. Locate the character in the code chart.

   The code chart lists the hexadecimal number for the character (*Figure 13-4*).

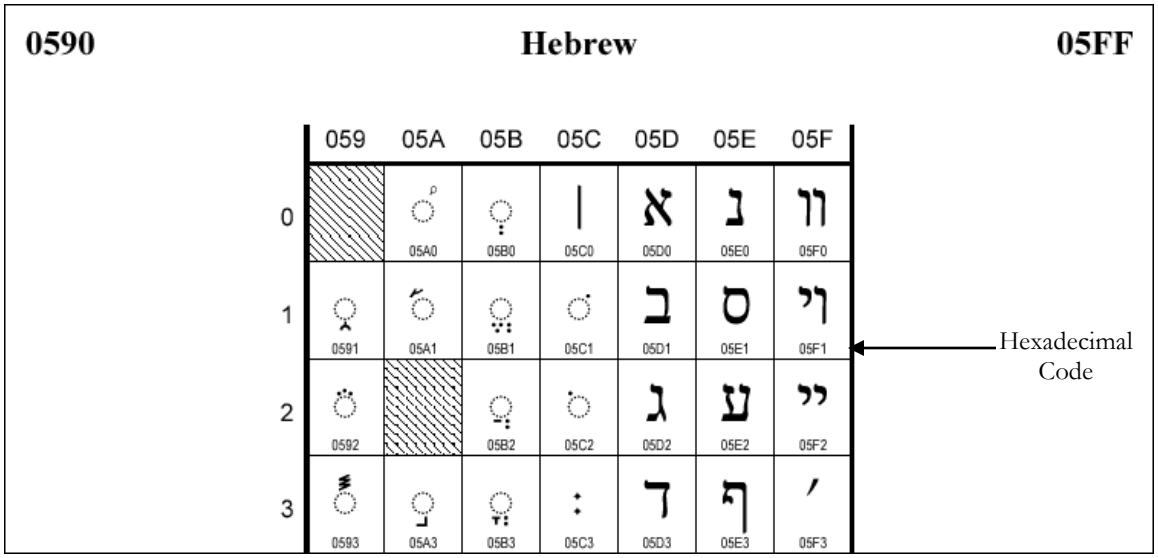


Figure 13-4. Unicode Code Chart - Hebrew Character Set

3. In a text editor, open the sort map used in your database.

   Note: If you do not know which sort map your database uses, you can view a copy of the sort map used in the database by logging in to the database as the **dbadmin** user and running the following command:

   map.exe -g 2 3 1 sortmap.txt

   This command saves the entire sort map to a file named **sortmap.txt**. Sometimes the name of the sort map is listed in the header of this file.

4. Using the find feature of your text editor, find the hexadecimal number for the character in the character code of a line.
   - If the mapping for the character is to the same hexadecimal number as the character for which you searched, the character maps to itself. This means that you can determine where the character is in the sort order by comparing the character's hexadecimal number with that of other mapped characters. For information about comparing hexadecimal numbers, see the section "Understanding Hexadecimal Numbers" in this document.

- If the mapping for the character is to a different hexadecimal number than that of the character for which you searched, the character is sorted using the character specified by the mapped hexadecimal number. You can use the code charts to find the character associated with the hexadecimal number.

Tip: Unicode code charts group characters by blocks of character sets, however, characters that are shared by multiple character sets are only listed in one block. For more information on how codes are grouped in a Unicode code chart, go to *www.unicode.org/unicode/standard/where*.

13.4.1.5 Example: Finding the Sort Mapping for a Character

You can determine the sort order of any character by . . .

1. Using a code chart to find the hexadecimal number for the character.
2. Using the sort map table to find the character's mapping.
3. Determining the character associated with that hexadecimal number (if the character is mapped to another hexadecimal number).

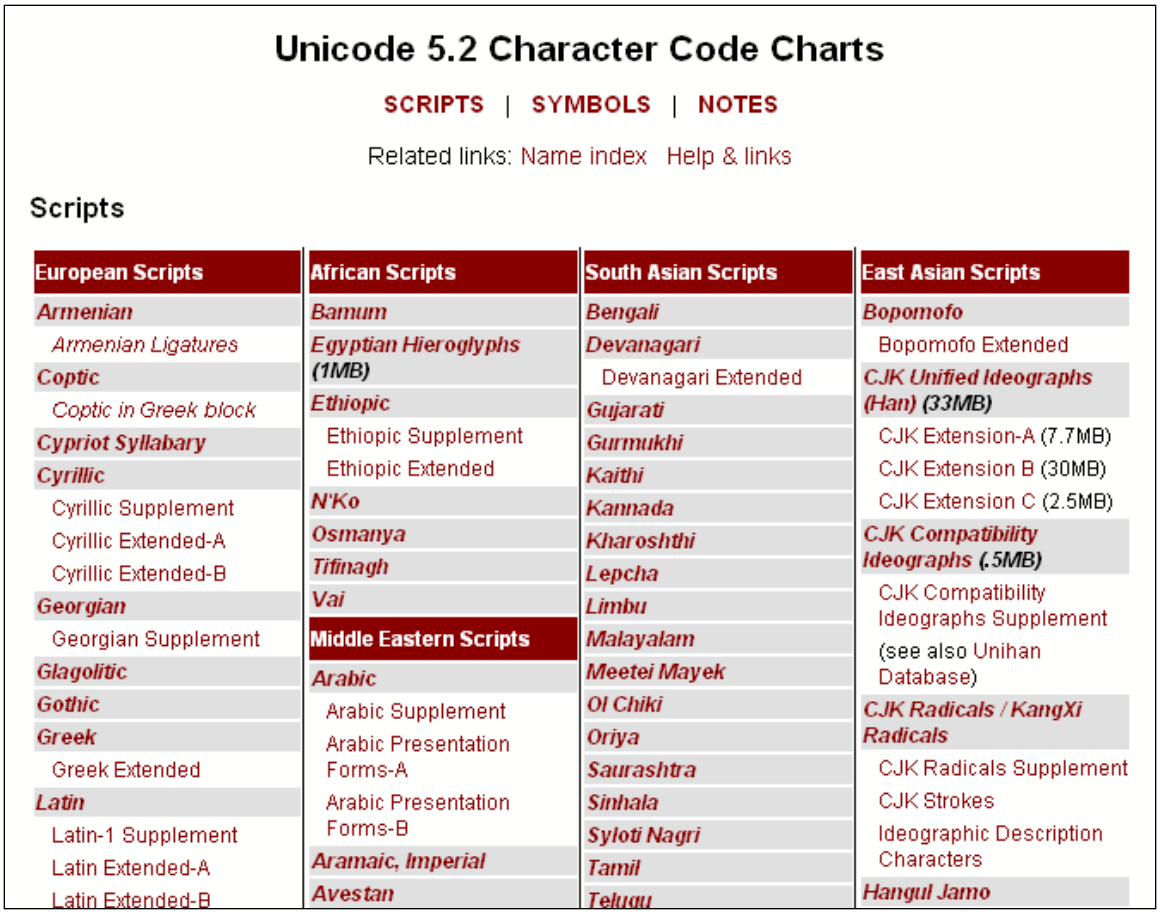
This section uses the German sort map to determine the sort order of the following characters:

- **æ**
- **ş**

13.4.1.5.1 Finding the Hexadecimal Number for Each Character

To find the hexadecimal number for each character,

1. Access the list of code charts at *www.unicode.org/charts* (*Figure 13-5*).



Unicode 5.2 Character Code Charts

SCRIPTS | SYMBOLS | NOTES

Related links: Name index Help & links

Scripts

European Scripts	African Scripts	South Asian Scripts	East Asian Scripts
Armenian	Bamum	Bengali	Bopomofo
Armenian Ligatures	Egyptian Hieroglyphs (1MB)	Devanagari	Bopomofo Extended
Coptic	Ethiopic	Devanagari Extended	CJK Unified Ideographs (Han) (33MB)
Coptic in Greek block	Ethiopic Supplement	Gujarati	CJK Extension-A (7.7MB)
Cypriot Syllabary	Ethiopic Extended	Gurmukhi	CJK Extension B (30MB)
Cyrillic	N'Ko	Kaithi	CJK Extension C (2.5MB)
Cyrillic Supplement	Osmanya	Kannada	CJK Compatibility Ideographs (.5MB)
Cyrillic Extended-A	Tifinagh	Kharoshthi	CJK Compatibility Ideographs Supplement
Cyrillic Extended-B	Vai	Lepcha	(see also Unihan Database)
Georgian	Middle Eastern Scripts	Limbu	CJK Radicals / KangXi Radicals
Georgian Supplement	Arabic	Malayalam	CJK Radicals Supplement
Glagolitic	Arabic Supplement	Meetei Mayek	CJK Strokes
Gothic	Arabic Presentation Forms-A	Ol Chiki	Ideographic Description Characters
Greek	Arabic Presentation Forms-B	Oriya	Hangul Jamo
Greek Extended	Aramaic, Imperial	Saurashtra	
Latin	Avestan	Sinhala	
Latin-1 Supplement		Syloti Nagri	
Latin Extended-A		Tamil	
Latin Extended-B		Telugu	

Figure 13-5. List of Code Charts at Unicode.org

2. Click the link to your character set. In this example, the character sets are Latin-1 Supplement for **æ** and Latin Extended-A for **ş**.

> **Tip:** Each Unicode character is listed once in the Unicode code charts. This means that if a character used in your character set is listed in the code chart for a preceding character set, the character will not be listed in the chart for your character set. If you cannot find a character in the code chart for your character set, look for it in the preceding code charts. For more information on how codes are grouped in a Unicode code chart, go to *www.unicode.org/unicode/standard/where*.

3. Browse the code chart for each character.
4. Copy the hexadecimal number associated with each character. Figure 13-6 displays **æ** in the code chart for Latin-1 Supplement.

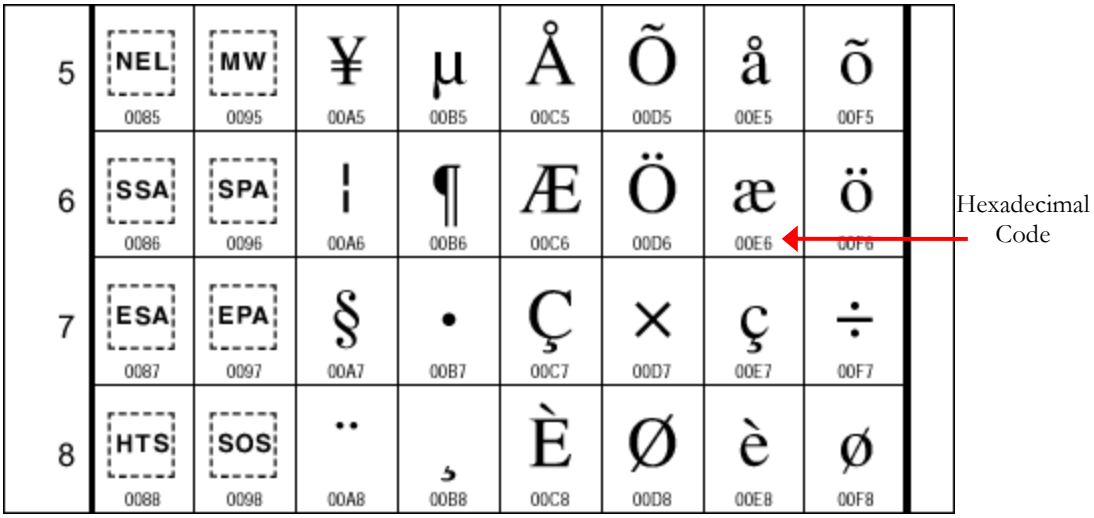


Figure 13-6. Unicode Code Chart

The code charts show that the hexadecimal number for . . .

- **æ** is 00E6
- **ş** is 015F

13.4.1.5.2 Using the Sort Map Table to Find the Character's Mapping

To find the mapping for æ and ş,

1. Open the sort map for your database in a text editor.
2. Using the Find mechanism of your text editor, search for the hexadecimal number of each character until you find the hexadecimal number in the Character Code portion of a line item. Figure 13-7 shows the Character Code entry for **æ**.

```
U+0x02bb -> SKIP    # 0xB0
U+0x0142 -> PUT 0x4c    # 0xB1
U+0x00f8 -> PUT 0x4f    # 0xB2
U+0x0111 -> PUT 0x44    # 0xB3
U+0x00fe -> PUT 0x54 0x48    # 0xB4
U+0x00e6 -> PUT 0x41 0x45    # 0xB5
U+0x0153 -> PUT 0x4f 0x45    # 0xB6
U+0x02dd -> SKIP    # 0xB7
U+0x0131 -> PUT 0x49    # 0xB8
```

Figure 13-7. Sort Map

The mapping entries for **ş** and **æ** are:

- **æ**: U+0x00E6 -> PUT 0x41 0x45
- **ş**: U+0x015F -> PUT 0x53

The mapping for **æ** indicates that the character will be sorted using *two* characters: 0041 and 0045.

The mapping for **ş** indicates that the character will be sorted using the hexadecimal number 0053.

Since the first hexadecimal number of the mapping for **æ** is less than the hexadecimal number of the mapping for **ş**, **æ** is sorted *before* **ş**.

13.4.1.5.3 Finding the Mapped Character

The hexadecimal numbers to which **æ** and **ş** are mapped are related to other characters. You can find out which characters **æ** and **ş** are mapped to using the Unicode code charts.

Code charts are sorted in numeric order by each character's hexadecimal number (*Figure 13-8*).

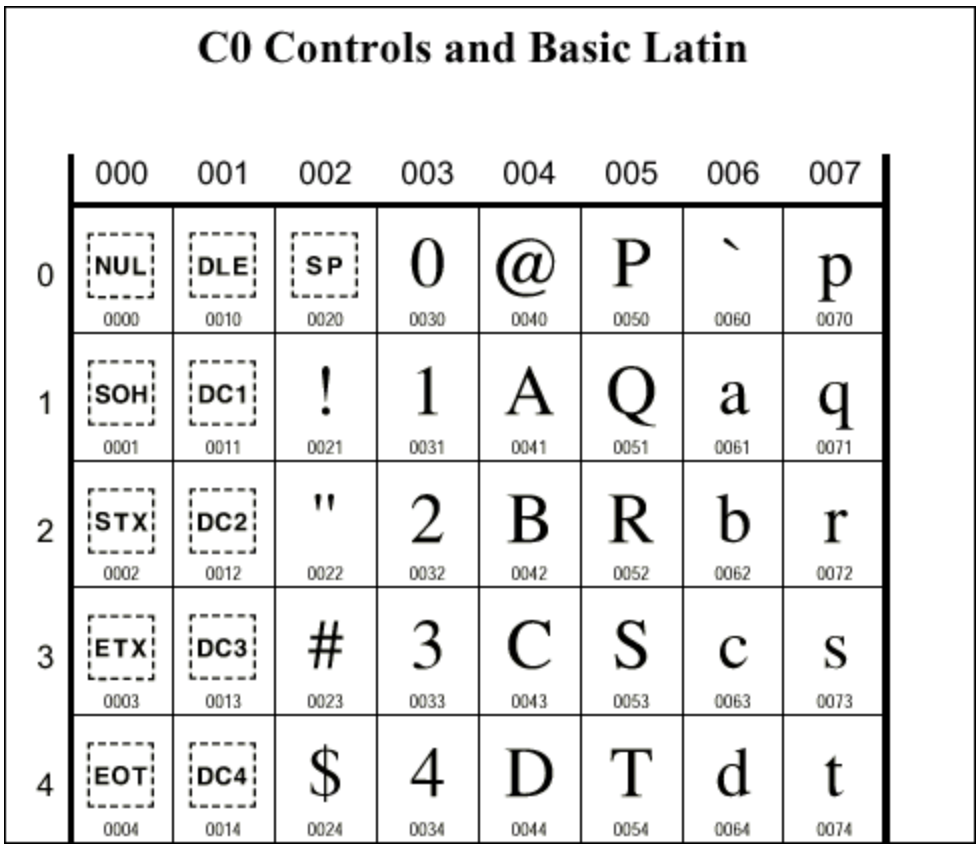


Figure 13-8. Unicode Code Chart - Basic Latin

Using the hexadecimal number of each mapping, you can find that the character for the hexadecimal number . . .

- 0041 is **A**
- 0045 is **E**
- 0053 is **S**

This means that . . .

- **æ** maps to **AE**
- **ş** maps to **S**

13.4.2 Loading Sort Maps

The script **LoadSortMap.sh** is used to load a new sort map to the database. In most cases, this script does not need to be run. However, the script should be run if you change an existing sort map or want to use a different sort map.

Hint: If you have altered a sort map, you must load the updated version to your database before running this script.

To run LoadSortMap.sh,

1. Log in to your server as **dbadmin**.

2. In the /usr/vtls directory, type the following:

```
LoadSortMap.sh [sort_map_name]
```

where **[sort_map_name]** is the name of the sort map you wish to load (see the section "Sort Maps" in this guide for a list of sort map names).

The script loads your sort map. The next time you sort items in the Virtua client or Chamo, the mapping defined in the specified sort map will be used to sort results lists.

Note:
- If you do not specify a sort map name, **LoadSortMap.sh** will load the default sort map, sort2-intl.
- If you type an invalid sort map name, the script will prompt you with a list of valid sort map names.

13.4.3 After Running LoadSortMap.sh

After loading your new sort map, you need to run the following scripts to populate the database tables with the newly sorted data. If you do not do this, no changes will take place in your database.

- **Re_CreateHeadingSort.sh**
- **KeywordIndex.exe** (which may not need to be run, depending on which sort map is being loaded and whether an associated parameter is set.)
- **CallIndexForm.sh**
- **ReIndexPatrons.sh**
- **PopulateBibFields.exe**

You can find documentation for each of the first four scripts in the *System Management: OPAC User's Guide*, and for **PopulateBibFields.exe** in the *System Management: Cataloging User's Guide*.

14. Appendix E - Troubleshooting Scripts

14.1 Working with Redo Log Groups

14.1.1 Switching the Current Redo Log Group

The script **SwitchCurrentRedoLogGroup.sh** will switch the current redo log group. It is generally used for Oracle troubleshooting purposes only.

Usage=`./SwitchCurrentRedoLogGroup.sh`

14.1.2 Displaying the Redo Log Switches

The script **DisplayRedoLogSwitches.sh** will display the number of times the Oracle redo log groups switched each hour for the past 24 hours.

For maximum performance, Oracle redo logs should switch no more than four times per hour during peak database usage. If you notice a large number of switches per hour, contact Innovative Customer Support and request that the sizes of the Oracle Redo Log be increased.

Note: If you need help determining the optimal size of the redo log files, in the **initvtls.ora**, set the fast_start_mttr_target parameter to 120, and then query the optimal_logfile_size column in the **v$instance_recovery** view.

Usage=`./DisplayRedoLogSwitches.sh`

14.2 Reporting on Performance Problems

To help with analyzing performance issues, run **GenerateStatsPackReport.sh**. The script generates a detailed report of internal database metrics regarding structures and usage. To capture applicable data, run the script during peak operating hours.

Usage=`./GenerateStatsPackReport.sh`

14.3 Checking Oracle for Block Corruption

Virtua provides a script **CheckOracleForBlockCorruption.sh**, which scans the Oracle data files for both physical and logical block corruption. If any block corruption is detected, the Oracle object and block number are reported.

Important: This script is designed to be used only with Oracle 11*g* and up.

This script must be run with the database OPEN or MOUNTED. It may take up to one hour to run and can be resource intensive. It is recommended that the script be run during off-peak hours to limit the impact on a production environment.

If block corruption is detected or reported, contact Innovative Customer Support immediately!

Usage: `CheckOracleForBlockCorruption.sh [parallel_degree]`

15. Appendix F - Changes in this Guide

15.1 Changes for Version 16.1

No changes were made.

Index

/

/dev/null · 17, 18

2

2709ToXML.exe · 74

9

99dmpXX_X script · 64

A

Ad Hoc reporting · 67
ADD_AAP_LINKS_ONLY · 109
AddDiagnosticText.exe · 74
AddItemCallNumberPrefix.sh · 74
addItemStatus.sh · 74
AddTablespaceDataFile.sh · 32, 74
AL32UTF8 · 104
AnalyzeCTXBibliographicIndex.sh · 74
application server · 36
arch directory · 9
archive logging · 47
  configuring · 47
  determining of enabled · 50
  disabling · 50
  duplexing · 48
  enabling · 49
  setting a file location · 48
archive logs · 9
  determining the destination · 48
  determining the format of filenames · 49
  duplexing · 48
  filling up · 48
authClr.sh · 75
autoClaim · 75, 108
autoClaim.log · 108
availabilitynotice.utf · 109
available guides · 1

B

background process
  detecting · 18
  killing · 19
  starting · 15
backup files
  managing physical media · 58
  storing · 54
  validating · 56
backups · *See* database backups
BatchDeleteItems.exe · 75
BatchDeleteVendors.exe · 75
Binary Large Object (BLOB) format · 8
block corruption, checking for · 134
blockpat.exe · 75

C

CallIndexForm.sh · 72, 75, 132
CallIndexFormItemsOnly.sh · 75
callNumberKey.exe · 75
Chamo
  directory structure · 4
ChangeAllowRequest.sh · 75
ChangeItemClass.sh · 75
ChangeItemPrice.sh · 75
ChangeItemStatus.sh · 76
ChangeLoanPeriod.sh · 76
ChangeLocation.sh · 76
ChangeLocationByBarCode.sh · 76
ChangeLocationByCallNumberRange.sh · 76
ChangeLocationByItemId.sh · 76
ChangeLocationId.sh · 76
ChangeOracleUserPassword.sh · 30, 76
ChangePermAuthorityToProv.kh · 76
ChangeReserveItemClass. sh · 76
changes in the guide · 135
ChangeShelfLocation. sh · 76
character maps · 5, 69
  codes · 111
  input · 5
  input, determining which are loaded · 69
  loading · 70
  output · 5
  output, determining which are loaded · 69
  sort · 5
character sorting · 115
CheckAllFRBRLinks.sh · 77
CheckCircParameters.sh · 77
CheckedOutItems.sh · 77
CheckForReciprocal1xx5xxAuthorities.sh · 77
CheckOracleForBlockCorruption.sh · 77, 134
CheckTablespaceFileSize.sh · 7, 32, 63, 77
child process · 40

circbackup.exe · 77
CircReport.exe · 78
CLAIM_BATCH_LEVEL · 108
CLAIM_BATCH_TYPE · 108
clas99 test database
  creating · 65
  installing key · 68
CleanUpRequestDirectories.ksh · 78
cold backups · 51
  procedures for · 51
  tarring the database directory · 51
CombineFRBR.exe · 78
CombineFRBRRecords.sh · 78
comparing hexadecimal numbers · 118
connections to the database, displaying active · 28
control files · 10, 11
ConvertMARCFileCharacterSet.exe · 78
ConvertMARCRecordsToRDA.exe · 78
ConvertMARCRecordsToRDA.sh · 78
copying production database to clas99 · 64, 66
  disk space required · 66
  errors to ignore · 67
corruption, checking Oracle files for · 134
cr_CTX_Indexes.sh · 78
Cr_Spfile.sh · 54, 80
create_adhoc_views.sh · 63, 79
create_intermedia.pl · 17, 79
Create024PermalinkTagInAuthorityRecords.exe · 79
Create024PermalinkTagInBibRecords.exe · 79
CreateBibIdsFromItemIds.sh · 79
CreateCustomAdHocView.sh · 80
CreateDirectoryObject.sh · 34, 80
CreateEnvVariableScript.sh · 14, 80
CreateItemBarcodesFromItemIds.sh · 80
CreateItemIdsFromBarcodes.sh · 80
CreateItemIdsFromBibIds.sh · 80
CreateRMANOnlineBackup.sh · 52, 55, 80
CreateRMANRestoreScript.sh · 55
CreateSubjectThesauri_1.sh · 80
CreateSubjectThesauri_2.sh · 81
cron job · 20
  about · 20
  creating · 21
  unique to each UNIX user · 20
  viewing · 20
crontab · 20, 21
  adding an entry · 21
  format of entries · 20
crontab process · 16
crontab, viewing entries · 20, 21
ctl01 directory · 10, 11
ctl02 directory · 10, 11
ctl03 directory · 10, 11
ctxsys user · 31

D

data directory · 8
data files
  about · 7
  adding to a tablespace · 32
  checking · 32
  resizing within a tablespace · 33
  working with · 32
data loss · 46
Data Pump tool · 59
  exporting and importing a database · 61
data recovery · 45
database
  exporting · 61
  importing · 61
  restoring · 55
database backup files
  managing physical media · 58
database backups · 45
  cold backups · 51
  consequences of not having a good backup strategy · 46
  dbv · 56
  hot backups · 52
  importance of · 46
  options for · 47
  recommendations for · 59
  recommendations for archiving on tape · 58
  recovery · 45
  storing on separate disks · 58
  strategy for · 46
  validating backup files · 56
database connections, determining · 83
database connections, displaying active · 28
database server · 26
database tables
  displaying parallel · 29
  viewing with disabled logging · 28
database version, determining · 43, 84
date format, changing · 109
dbadmin user · 20
dbv · 56
  not available for exports · 56
  not available for redo log files · 56
  using · 56
dbv utility · 59
DecodeBibLeaderData.sh · 81
Delete_Circ_Trans_Log_OFFLINE.sh · 81
Delete_Circ_Trans_Log_ONLINE.sh · 81
Delete_Operation_Log.sh · 81
DeleteExpiredPatronBlocks.sh · 81
DeleteExpiredPatrons.sh · 81
DeleteInactivePatrons.sh · 82
DeleteItemsByBarcode.sh · 82
DeleteItemsByDueDate.sh · 82
DeleteItemsByLocation.ksh · 82
DeleteItemsByStatus.sh · 82

DeleteOldFines.sh · 82
DeleteOldItems.sh · 82
DeleteOldPurchaseRequests.sh · 82
DeleteOrphanFRBRLinksAndBibs.sh · 83
DeletePatronPasswords.sh · 83
DeletePatrons.sh · 83
directory structure
  Chamo installation · 4
  InfoStation installation · 4
  non-standard · 3
  Oracle database files · 6
    arch directory · 9
    ctl01 directory · 10
    ctl02 directory · 11
    ctl03 directory · 11
    data directory · 8
      large tablespaces · 8
      lg1 · 8
      md1 · 8
      medium tablespaces · 8
      sm1 · 8
      small tablespaces · 8
    impexp directory · 66
    indx directory · 8
      large indexes · 8
      lg1 · 8
      md1 · 8
      medium indx · 8
      sm1 · 8
      small indexes · 8
    initvtls.ora · 7
    intr directory · 10
    lob directory · 8
    migrate directory · 9
    rdo1 directory · 9
    rdo2 directory · 10
    system directory · 10
    undo directory · 10
  Virtua applications software · 3, 4
    lang directory · 5
    sql directory · 4
    src directory · 5
  Virtua server installation · 3
DisableArchiveLogging.sh · 50, 83
disabling archive logging · 50
disk failure · 45
Display ParallelTables.sh · 84
DisplayActiveDBConnections.sh · 28, 83
DisplayNoLoggingTables.sh · 28, 83
DisplayOracleRelease.sh · 27, 83
DisplayParallelTables.sh · 29
DisplayRedoLogSwitches.sh · 84, 133
DisplayVirtuaDBRelease.sh · 43, 84
DocumentBatchDeleter.sh · 84
DocumentBatchLoader.sh · 84
DocumentLoader.sh · 84
DpExport_Database.sh · 61, 63, 84
DpImport_Database.sh · 63
drop_CTX_Indexes.sh · 84
duplexed log files · 9, 10
DuplicateOptions.exe · 85

E

echo · 13
EDIFACT · 107
EDIFACT_NEW_LINE · 107
EnableArchiveLogging.sh · 49, 85
enabling archive logging · 49
environment variables · 12
  ADD_AAP_LINKS_ONLY · 109
  CLAIM_BATCH_LEVEL · 108
  CLAIM_BATCH_TYPE · 108
  EDIFACT_NEW_LINE · 107
  EXE_DIR · 103
  getting the value of · 13
  HEADING_INDEXING_OFF · 110
  IIRS_SYS · 107
  importance of · 12
  LC_TIME · 109
  NLS_LANG · 104
  ORACLE_HOME · 103
  ORACLE_SID · 104
  PATH · 104
  setting as part of a script · 14
  setting the value of · 13
  unsetting the value of · 14
  used by Virtua · 102
  V_LOG_LEVEL · 105
  V_LOG_TYPE · 106
  V_LOG_TYPE_PARAM · 106
  VIRTUA_PASSWORD · 105
  VIRTUA_USER · 105
  VTLS_TEMP · 107
EXE_DIR · 103, 107
executables · 5, *See* scripts and executables
ExportPatronBarcodeTable.sh · 85
extract_bibs_by_creation_date.sh · 85
extract_deleted_bibs.pl · 85
Extract_FRBR_Records.sh · 85
extract_patrons_by_modify_date.sh · 85
extract_patrons_in_error_state.pl · 85
ExtractAuthorityRecordsUsingControlNumbers.sh · 85
ExtractForDiscovery.sh · 86

F

find command · 22
finding text · 22
FindMissingFRBRLinks.sh · 86
FixAuthConflictErrors.sh · 86
flash recovery area, setting up · 53
FRBR records
  extracting · 85

indexing · 95
managing · 77, 86
FRBRQualifyFromVTLS.sh · 86
fyRollover.exe · 86

G

GatherSchemaStats.sh · 33, 86
GatherTableStats.sh · 33, 86
GenerateStatsPackReport.sh · 86, 134
get_bibs.pl · 86
Get_FixedField · 86
Get_Leader · 87
Get_Tag · 87
GetExipirationDate.exe · 44, 86
GetJournalTitleIdsUsingTHL.sh · 87
getURLs.sh · 87
globalChange1.ksh · 87
globalChange2.ksh · 87
Grandfather-Father-Son rotation · 58
grep · 18

H

HandleRequestDeactivationStatus.sh · 87
HEADING_INDEXING_OFF · 110
hexadecimal numbering · 115
about · 116
comparing · 118
counting · 116
hot backups · 52

I

IdentifyBibTitleWithReturnChars.sh · 87
IdentifyDuplicatePatronBarcodes.sh · 87
identifying processes · 39
identifying the databases that are running · 27
IIRS_SYS · 107
impexp directory · 64, 66
ImportRDATranslations.sh · 88
Index24PermalinkTagInAuthorityRecords.exe · 88
indexes
large · 8
medium · 8
small · 8
indx directory · 8
InfoStation
directory structure · 4
InfoStation, identifying crontab entries · 21
initialization parameters · 7
initvtls.ora · 7, 47, 48, 53
determining whether in use · 54
log_archive_dest · 48
log_archive_duplex_dest · 48
log_archive_format · 49
log_archive_min_succeed_dest · 48
input maps · 70
codes · 111, 112
determining which are loaded · 69
loading · 70
installing key for clas99 · 68
InterMedia · 10
intr directory · 10
IsArchiveLoggingEnabled.sh · 50, 88
IsSpfileInUse.sh · 54, 88
ItemStatusMonitor.exe · 88
itivaReport.sh · 88

J

jloadmaps · 70, 88
running · 72
using to load character maps · 70

K

key for clas99, installing · 68
KEYWORD_INDEXING_OFF · 110
KeywordIndex.exe · 16, 18, 72, 88, 132
KeywordIndexParallelJob.exe · 89
kill command · 19
killing
a process · 19
psdriver.exe · 19, 40

L

lang directory · 5
LC_ALL · 109
LC_TIME · 109
lg1 indexes directory · 8
lg1 tablespaces directory · 8
license keys
expiration date of · 44
features enabled with · 43
updating · 44
verifying · 44
working with · 43
List of Titles screen · 115
List853Holdings.sh · 89
loading sort map · 131
subsequent tasks · 132
LoadPatronLanguageCode.sh · 89
LoadRecordConverterFiles.sh · 89
LoadSortMap.sh · 89, 122, 131
LoadSynonymFile.sh · 89
LoadThesaurus.ksh · 89

lob directory · 8
LOCAL0 · 39
locale, setting in Solaris environment · 109
log level · 37
  default · 37, 106
  recommended level · 38
  setting · 38, 105
log_archive_dest parameter · 48, 50
log_archive_duplex_dest parameter · 48
log_archive_format parameter · 49
log_archive_min_succeed_dest parameter · 49
logging server output · 37
logging status, checking for tables · 28

M

making changes to the database · 25
map.exe · 69, 89
  using to determine which input and output maps are loaded · 69
MapAynAlifCharsInAuthorityRecords.sh · 89
MapAynAlifCharsInBibRecords.sh · 90
MapChars.exe · 72
  running · 73
MapSharpSCharacterInAuthorityRecords.sh · 90
MapSharpSCharacters.exe · 90
MARCBibUpdate.exe · 90
MarcView.exe · 90
Max. Simultaneous Sessions limit Profiler setting · 41
maximum number of simultaneous connections · 41
md1 indexes directory · 8
md1 tablespaces directory · 8
migrate directory · 9
migration directory · 9
modifying
  database directly · 25
MoveStateRecords.exe · 90
multiplexed control files · 10, 11

N

NLS_LANG · 104
  setting the character set · 104
nohup · *See* scripts and executables, running as background processes
NOLOGGING mode, Virtua tables and · 83

O

offline backups · *See* cold backups
online backups · *See* hot backups
OptimizeAllIndexes.sh · 90
Oracle · 24
  Data Pump tool · 61
  directory structure
    database files · 6
  do NOT directly modify data in the database · 25
  performing basic tasks · 25
  recommended knowledge · 25
  redo log groups, working with · 133
  RMAN tool · 52
  tablespaces and data files · 7
  working with · 24
Oracle Listener
  displaying status of · 30
  starting · 30
  stopping · 30
  working with · 29
Oracle users
  ctxsys · 31
  default passwords for · 31
  refreshing privileges · 31, 92
  sys · 31
  system · 31
Oracle utilities, authorization to use · 24
Oracle version, determining · 27
ORACLE_HOME · 103
ORACLE_SID · 73, 104
OracleListener.sh · 90
output maps · 70
  codes · 111, 114
  determining which are loaded · 69
  loading · 70

P

passwords
  changing
    for Innovative-created Oracle users · 30
    for Oracle users · 30
PATH · 104
performance, analyzing · 134
PermAuthIdsToProv.ksh · 90
PopulateBibFields.exe · 91, 132
PopulateChamoIndexQueue.sh · 91
PopulateCollectionGroupFilters.sh · 91
PopulateFRBRLocationFilters.sh · 91
PopulateItemClassFilters.sh · 91
PopulateMarc21FormatFilter.sh · 91
PopulateMissingLocationFilters.sh · 91
PopulatePostalCode.sh · 91
PopulateRequestGroup.sh · 91
PopulateUDCBrowse.sh · 91
PrintAuthoritybyState.ksh · 92
process
  detecting · 18
  killing · 19
prod_db_copy script · 66
prod_db_stats script · 66
production database, copying to clas99 · 64, 66
  disk space required · 66

errors to ignore · 67
ps command · 18, 19, 27, 39
psdriver.exe · 35, 92
identifying processes · 39
child process · 40
parent process · 40
process ID · 40
killing · 19, 40
limit on simultaneous connections · 41
log levels · 37
logging server output · 37
refuses connections when license key expires · 44
running · 36
running from a script · 36
setting the log level · 38
specifying a process · 36
starting two processes · 35

R

rdo1 directory · 9, 10
rdo2 directory · 9, 10
Re_CreateHeadingSort.sh · 17, 72, 92, 132
Re_indexForUnicodeNormalization.sh · 92
recommendations for database backups · 59
RecordConverter.exe · 92
recovery · 45
recovery tool · 52
redirecting messages
to /dev/null · 17, 18
redirecting output · 16
stderr · 16, 17
stdout · 16, 17
stdout and stderr · 18
redo log files · 9, 47
redo log groups, working with · 133
redo log switches, displaying · 84
RefreshSequences.sh · 92
RefreshSyns.sh · 31, 92
ReIndexPatrons.sh · 92, 132
ReIndexUserHeadings.sh · 92
removeAddItemStatus.ksh · 93
RemoveBlankItemCallNum.sh · 93
RemoveCRLF.exe · 93
RemoveDeactivatedRequests.sh · 93
removeItemStatus.ksh · 93
RemovePatronsByDeletionDate.sh · 93
RemoveStreetDateStatus.sh · 93
RemoveTabCharacterFromBibRecords.exe · 93
RepairUTF8InAuthorityRecords.sh · 93
RepairUTF8InBibRecords.exe · 94
RepairUTF8InBibRecords.sh · 94
RepairXMLFile.exe · 94
ReplaceSubstringInItemCallNumber.sh · 94
ReProcess240Titles.sh · 94, 96, 98
ReProcessAuths.ksh · 94
ReProcessBibs.ksh · 94, 109
ReProcessBibsToIndexVitalPid.sh · 94
ReProcessForPublisherHeadings.sh · 95
ReProcessForPublisherUserHeadings.sh · 95
ReProcessFRBRWorksForSubjects.sh · 95
ReProcessPatrons.sh · 95
RequestDaemon.sh · 95
RequestRefresh.exe · 95
ResetChamoAdminPassword.sh · 95
ReSetTempCircCount.ksh · 95
ResizeTablespaceDataFile.sh · 33, 96
Restore_clas$DB_DIGITS.sh · 52, 55, 96
RestoreAuthorityNoteModifyDate.exe · 96
RestorePatronsToGoodStanding.sh · 96
restoring the database · 55
RetrieveAuthFixedTagData.sh · 96
RetrieveAuthLeaderData.sh · 96
RetrieveAuthVariableTagData.sh · 96
RetrieveBibFixedTagData.sh · 96
RetrieveBibLeaderData.sh · 96
RetrieveBibVariableTagData.sh · 97
rlint.exe · 97
RMAN online backups · 52
RMAN tool · 52

S

Schedule_FloatingMV_Refresh.sh · 97
script for setting environment variables · 14
scripts · 5
scripts and executables
redirecting output · 16
running as background processes · 15
specifying the path to · 103
tips for running · 15
used for troubleshooting · 133
SetAutoAddPermalinksFlag.sh · 97
SetAutoShelfLocationReset.sh · 97
SetConsiderAllRequestedItemsFlag.sh · 97
SetIndexInvalidISBNFlag.sh · 97
SetMaskAndKeywordIndexBibs.sh · 98
SetNoNISOSerialDisplay.sh · 98
SetPatronBarcodesAreUniqueFlag.sh · 98
SetPreventBrowseOf2ndItemLevelCallNumber.sh · 98
SetServerKeyCode.exe · 44, 68, 98
SetTag240IsSeparateTitleFlag.sh · 94, 98
setting the log level of psdriver.exe · 38
SetUpRequestsDirectories.sh · 98
SetUseKeywordSortMapTableFlag.sh · 98
shutting down the database · 26
shutdown immediate · 26
using stopdb · 26
simultaneous sessions, limits for · 41
sm1 indexes directory · 8
sm1 tablespaces directory · 8
Solaris environment, locales and · 109
sort map · 70
codes · 111

loading · 70, 122, 131
tasks after loading · 132
using · 122
spfile · 54
creating · 54
determining whether one is in use · 54
sql directory · 4
SQL scripts, authorization to run · 4
SQL*Plus
caveat against using · 24
src directory · 5
specifying the path to · 103
standard error · *See* stderr
standard out · *See* stdout
startdb · 26
starting
database · 26
Oracle Listener · 30
startps, redirects output to /dev/null · 36
statistics, gathering · 33, 134
stderr · 17, 18, 106, 108
stdout · 17, 18
stopdb · 26
StopIndexingSubfieldEInCorporateHeadings.sh · 99
stopping
database · 26
Oracle Listener · 30
SwitchCurrentRedoLogGroup.sh · 99, 133
sys user · 31
sys_01.dbf · 10
syscli user · 31
syslog · 39, 106, 108
system directory · 10
system files · 10
sys_01.dbf · 10
temp_01.dbf · 10
users_01.dbf · 10
System Management: Acquisitions and Serials User's Guide · 1, 108
System Management: Cataloging User's Guide · 1
System Management: Circulation User's Guide · 1
System Management: OPAC User's Guide · 1
System Management: Reporting User's Guide · 1
system tablespace file · 10
system user · 31

T

tablespaces
about · 7
adding data files to · 32
checking · 32
large · 8
medium · 8
resizing data files within · 33
small · 8
working with · 32
tape drive · 58
tape recommendations · 58
temp_01.dbf · 10
temporary tablespace file · 10
test database
creating · 65
installing key · 68
testState.sh · 63, 99
Thai parsing · 107
ToMARC21.exe · 99
troubleshooting scripts · 133

U

undo directory · 10
UNIX
environment variables · *See* environment variables
minimum knowledge for working with · 12
tips for working with · 12
working with · 12
UnlockMARCRecord.sh · 99
unlockProfilerAccount.sh · 99
update_circ_transation_log.pl · 100
Update_Patron_015b.sh · 85, 100
UpdateDueDate.sh · 99
UpdateOverdueFines.exe · 99
UpdatePatronTypeByAge.sh · 99
user limit constraints · 41
users
refreshing privileges for Virtua's Oracle · 31
users tablespace file · 10
users_01.dbf · 10
UTF-8 files, viewing on Solaris servers · 109

V

V_LOG_LEVEL · 38, 105
V_LOG_TYPE · 39, 106
V_LOG_TYPE_PARAM · 106
validating backup files · 56
vendor records, deleting · 75
version of database, determining · 43, 84
version of Oracle, determining · 27
Virtua client
connecting to psdriver.exe · 35
reaching the user limit · 41
Virtua Profiler/Introduction and Global Settings User's Guide · 41
VIRTUA_PASSWORD · 105
VIRTUA_USER · 105
vload.exe · 100, 107
VTLS_REPORT_PRIVS error, meaning of · 67
VTLS_TEMP · 107

W

WhichSortMapsLoaded.sh · 100
wrapper.pl · 21
write2709.exe · 100
WriteAuthIdsFile.ksh · 100
WriteBibIdsFile.ksh · 100
WriteHoldingIdsFile.ksh · 100
WriteItemIdsFile.ksh · 100
WriteMaskedBibIdsFile.sh · 100
WritePatronIdsFile.ksh · 100
WritePermAuthIds.ksh · 101
WritePermAuthorityFile.ksh · 101
WritePermSubjectIds.ksh · 101

X

XMLToMARC.exe · 101