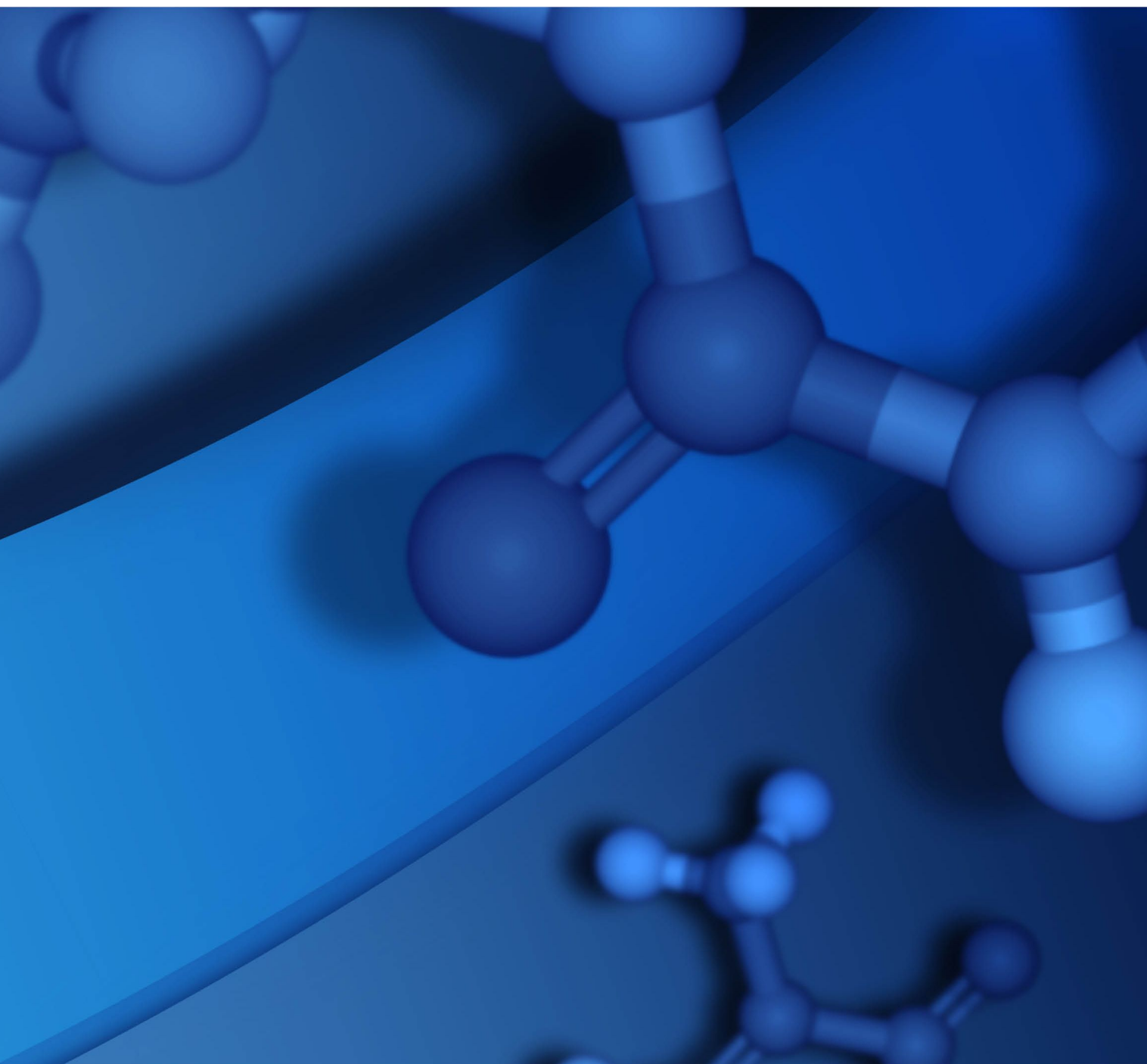


ADMINISTRATION GUIDE

BIOVIA DIRECT 2021



Copyright Notice

©2020 Dassault Systèmes. All rights reserved. 3DEXPERIENCE, the Compass icon and the 3DS logo, CATIA, SOLIDWORKS, ENOVIA, DELMIA, SIMULIA, GEOVIA, EXALEAD, 3DVIA, 3DSWYM, BIOVIA, NETVIBES, IFWE and 3DEXCITE, are commercial trademarks or registered trademarks of Dassault Systèmes, a French "société européenne" (Versailles Commercial Register # B 322 306 440), or its subsidiaries in the U.S. and/or other countries. All other trademarks are owned by their respective owners. Use of any Dassault Systèmes or its subsidiaries trademarks is subject to their express written approval.

Acknowledgments and References

To print photographs or files of computational results (figures and/or data) obtained by using Dassault Systèmes software, acknowledge the source in an appropriate format. For example:

"Computational results were obtained by using Dassault Systèmes BIOVIA software programs. BIOVIA Direct was used to perform the calculations and to generate the graphical results."

Dassault Systèmes may grant permission to republish or reprint its copyrighted materials. Requests should be submitted to Dassault Systèmes Customer Support, either by visiting <https://www.3ds.com/support/> and clicking **Call us** or **Submit a request**, or by writing to:

Dassault Systèmes Customer Support
10, Rue Marcel Dassault
78140 Vélizy-Villacoublay
FRANCE

Contents

Chapter 1: About This Guide	1
Introduction to Direct	1
Audience and Prerequisite Knowledge	1
Related BIOVIA Direct documents	2
Chapter 2: Setting the Cartridge C\$DIRECT2021 Environment	3
The BIOVIA Chemical Environment	3
Overview of BIOVIA Global Chemical Environment	4
Writing Global Environment Information to a File	4
Editing the Global Environment Information	5
Loading Global Environment Files	5
Using Multiple Processors with SSS and RSS Searches	6
Customizing Environment Information	7
The BIOVIA Periodic Table	7
Format of the BIOVIA Ptable	7
The Direct Default Ptable	7
Customize the BIOVIA Ptable	10
The BIOVIA Salts Definition File	11
Customize the Salts Definition	16
Customize the Sgroupfields Definition	16
Allowed Sgroup Field Names and Data Types	17
Modifying the Sgroupfields Definition	18
Customize the Isotopedata Definition	18
Chapter 3: Creating and Managing Indexes and Tables	20
Creating Molecule Indexes and Tables	20
Molecule Index Without Generic Structures	20
Markush Molecule Index	21
Creating Molecule Tables	22
Examples	22
Create a Table to Contain Molecules	23
Add a Domain Index to the Molecule Table	23
Domain Index Creation	24
Biopolymer Searching in Direct	25
Specifying the Cutoff Size for SCSR Template Expansion	26

FLEXMATCH Searching	27
Substructure Searching	28
Computing Sequence Text	29
Fingerprint Searching	30
Creating a Fingerprint Index	30
Fingerprint Similarity Searching	30
Creating and Managing Reaction Tables and Indexes	31
Creating Reaction Tables	31
Create a Table to Contain Reactions	31
Insert Reactions into a Table	32
Example	32
Example	32
Add a Domain Index to the Reactions Table	32
Example	32
If CREATE INDEX Fails	33
Structure of a Domain Index	33
Example Objects	33
Permissions Required for BIOVIA Direct Users	37
Dropping and Recreating Domain Indexes	37
Storing Structures that Cannot be Indexed or Searched	38
Chapter 4: Maintaining Tables and Indexes	41
Routine Index Maintenance	41
Scan for Duplicate Molecules	41
Scan for Duplicate Reactions	43
Scan the Substructure Search Keys	46
Update the Inverted Key Index	47
Update a Fastsearch Index	47
Validate the Fastsearch Index	47
Rebuild a Domain Index	48
Rebuild the Fastsearch Index	49
Rebuild the FLEXMATCH or RXNFLEXMATCH Index	49
Rebuild the Molecular Formula Index	49
Rebuild the Substructure Search Keys	49
Truncate a Domain Index	50
Analyze a Domain Index	50

Recreate Invalid Domain Indexes	50
Optional Index Management Tasks	51
Add Fastsearch to a Domain Index	51
Add Substructure Search Keys to a Domain Index	52
Add a Modification Date Column to a Table	52
Add a molfile or Chime String Column to a Table	52
Add a rxnfile or Chime String Column to a Table	53
Add a Molecular Formula Column to a Table	54
Add a Molecule Name Column to a Table	55
Add a Molecular Weight Column to a Table	55
Add a Molecule Text Keys Column to a Table	56
Add an InChI key or SMILES String Column to a Table	57
Alter Duplicate Checking Behavior	58
Alter Inverted Keys Chunk Size	58
Recompute the Molecular Formula	59
Recompute the Molecular Weight Values	59
Validate the Molecular Formula Index	59
Validate Molecular Weights	60
Remove Fastsearch from a Domain Index	60
Remove Substructure Search Keys from a Domain Index	60
Rename a Domain Index	60
Drop a Domain Index	60
Enable NEMA Keys for Exact-match Searching	60
Disable NEMA Keys for Exact-match Searching	61
Running Routine Index Maintenance Procedures Automatically	61
Example	62
Atom-To-Atom Maps	67
Using the Automapper	67
Create a Status Information Table	67
Use the Automapper	67
Lock and Commit	68
Examine the Automap Status	68
Chapter 5: Local Partitioned Index	70
Creating a Partitioned Domain Index	70
Creating Indexes on Large Partitions	71

Adding a New Partition	71
Splitting a Partition	72
Rebuilding a Partition	72
Exchanging Table and Partition	72
Direct Index Maintenance Procedures	73
Parallelization of Index Maintenance	73
Parallelizing Index Partition Creation and Searching	73
Structure of Partitioned Domain Index	74
Limitations with Partitioned Domain Indexes	74
Chapter 6: Upgrading Indexes	77
Upgrading a Direct Table with an Existing Domain Index	77
Upgrading a Direct Table without an Existing Domain Index	79
Chapter 7: Exporting and Importing	80
Export and Import	80
If Import Fails to Create the Domain Index	81
Importing Data and the Domain Index	82
Chapter 8: Logging Information	85
Logging Index Operations	85
Logging to a File	86
Log levels	86
Create the MDL_DC_LOGGING Table	87
Change the Logging Level	87
Modify the Logging Table	87
Logging and Performance	88
Direct Error Messages	88
Chapter 9: Replication	90
Direct and Replication	90
Direct Replication Scripts	90
Configure Chemical Tables for Replication	90
Streams Replication	91
Streams Configuration and Administration	91
Limitations on Replication	92
Chapter 10: Command Reference	93
Cartridge Management Functions and Procedures	93
MDLAUX.CARTRIDGESHEMA	93

MDLAUX.CREATEBOOSTFILE	93
MDLAUX.CREATELOGFILE	94
MDLAUX.ERRORS	95
MDLAUX.GETENVFILE	96
MDLAUX.GETTEXTPROCID	96
MDLAUX.GETPROPERTY	97
MDLAUX.INDEXPARAMETERS	97
MDLAUX.KILLEXTPROC	98
MDLAUX.LOGTABLE	99
MDLAUX.NODROP	100
MDLAUX.NOSTRUCT	100
MDLAUX.PENDINGFASTSEARCH	101
MDLAUX.DELETEDFASTSEARCH	101
MDLAUX.PREPAREINDEXEXPORT	101
MDLAUX.RECREATEFASTSEARCH	105
MDLAUX.RECREATEINDEXES	107
MDLAUX.RECREATEKEYS	108
MDLAUX.REGENAAMAPS	110
MDLAUX.REMOVEBOOSTFILE	112
MDLAUX.SCANINDEX	112
MDLAUX.SETENVFILE	115
MDLAUX.SETFLAGS('NODATE')	116
MDLAUX.SETFLAGS('ForceFingerprintSearch')	117
MDLAUX.SETFLAGS('FORCEV3000')	117
MDLAUX.SETFLAGS('FORCEV2000')	117
MDLAUX.SETFLAGS('USEDANDT')	118
MDLAUX.SETPROPERTY	118
MDLAUX.SETUP	121
MDLAUX.SHOWBOOSTFILES	122
MDLAUX.UPDATEPENDINGFASTSEARCH	122
MDLAUX.UPDATEPENDINGINVERSIONS	124
MDLAUX.UPGRADEINDEXES_PREPARE and MDLAUX.UPGRADEINDEXES_UPGRADE	125
Parameters for Index Creation and Maintenance	127
Parameters for CREATE INDEX	128
Parameters for ALTER INDEX REBUILD	134

Parameters for ALTER INDEX	141
Chapter 11: Performance Tuning	144
Direct Performance	144
Fastsearch Index Cache and Searching	144
Oracle Real Application Clusters	145
Known Issues with Oracle RAC	145
Appendix A: Direct Index Tables	146
Description of the Direct Index Tables	146
Appendix B: Direct SDFile Import Utility	148
Syntax of the importSDFile Utility	148
Processing Options of the importSDFile Utility	148
Create the Intermediate Table	148
Import as MOLFILE	149
Import as 2D	149
Specify a Table in Another Schema	149
Oracle O/S Authentication	149
Rename SDFile Fields	150
NOWAIT Option	150
ERRORS N Option	150
REPORTFIELD Option	150
Numeric and Date Field Processing	151
Error Handling	151
SDFile Import Example	152
Appendix C: Direct RDFile Import Utilities	154
The Extended RDfile Import Utility	154
Extended RDFile Import Utility Operation	154
Tables and Columns	155
Using importRDFileEx	156
Table Definitions File	156
The RDFile Import Utility	164
RDFile Import Utility Operation	165
Tables and Columns	165
tableprefix Parameter	165
Handling of Specialized Conventions	166
Syntax of the importRDFile Program	166

Processing Options of the importRDFFile Program	166
Create the Intermediate Table	166
Import Structures in Character(-char) Mode	167
Oracle O/S Authentication	167
NOWAIT Option	167
Numeric and Date Field Processing	167
Control/C Interrupt	168
Error Handling	168
Tables Created by RDFFile Import Utility	169
Duplicate Checking	169
Domain Index Creation	170
Appendix D: SDFFile and RDFFile Export Utilities	171
Overview of the Export Utilities	171
Usage	172
ExportSDFFile	172
Designating the Molecule Column	172
Specifying the Record Source	172
Controlling Record Order	172
Controlling Which Records are Included	172
Renaming Columns	173
Formatting Columns	173
Oracle O/S Authentication	173
ExportRDFFile	173
Specifying a MOLREGNO or RXNREGNO Column	174
Specifying an extreg Column	174
Specifying a Molecule RDFFile Versus a Reaction RDFFile	174
Designating the Molecule Column	174
Specifying the Record Source	174
Controlling Record Order	174
Controlling Included Records	175
Renaming Columns	175
Formatting Columns	175
Oracle O/S Authentication	175
SQL File Syntax	176
Error Handling	176

Appendix E: The Direct Default Isotopedata Definition File	178
Appendix F: UniProt Converter	267
About the UniProt Converter	267
Running the UniProt Converter	268
Usage	268
Format of Fields File	269
Operation	269
Additional Information on UniProt Structure Conversion	270
FEATURE TYPES	271
Ignored Types	272
Structure Modifications	272
Sgroups	273
Abbreviation Classes	273
Known Limitations	273
Record Encoding Multiple Structures	273
Creating a Sample Proteins Database	274
Appendix G: Direct Boost Files	275
Storing in Boost Files	275
Adding Boost Files to a Molecule or Reaction Index	275
Location of Boost Files	276
Size of Boost Files	277
Boost Files Creation Time	277
Recreating Boost Files	277
Displaying Boost Files	278
Removing Boost Files	279
Appendix H: Upgrading Direct 8	280
Configuring Direct Global Cartridge Environment	280
Upgrading Direct 8 Molecule or Reaction Table	281
Upgrading Direct	282
Upgrade Example	285
Direct Upgrade Utility Parameters	286
Required Parameters	287
Optional Parameters	287
How To Determine If Direct Environment Files Are Customized	291
PTABLE	292

SALTS	292
SGROUPFIELDS	292
Converting a Previous Direct PTable or SALTS File to Direct 2021 Format	293
Common Warnings Issued by the directupgrade Utility	295
Appendix I: Direct 8 to Direct 2021 Chemistry Update	296
Audience	296
List of Changes	296
Valence Differences	296
Aromaticity	297
Tautomers	299
Stereochemistry of Allenes and Biaryls	301
Tetrahedral Stereochemistry	304
Geometric Stereoisomers	306
A-Line / Nonstandard Type Behavior	308
Molecular Weight Differences	309

Chapter 1:

About This Guide

This guide explains how to administer BIOVIA Direct on:

- Host computers that run the Red Hat Linux operating system
- Host computers that run the SUSE Linux operating system
- Intel or Intel-compatible host computers that run the Microsoft Windows Server operating system

It also explains how to administer the tables that Direct uses.

Introduction to Direct

Direct is an Oracle Database Server Data Cartridge that allows you to store and search chemical objects in an Oracle RDBMS. It includes an Oracle Database Server Domain Index. Direct extends the capabilities of Oracle Database Server. It provides transparent access to your chemical data.

Direct uses the same underlying chemistry and chemical perception algorithms as Pipeline Pilot Server. This chemical perception is a harmonization of the traditional Direct and BIOVIA Pipeline Pilot chemistry. More information on the Pipeline Pilot chemistry is located on your Pipeline Pilot Server Home Page. Go to *Help Center (Developers) > Administrators > Reference Guides > Collections > Chemistry Collection*.

Audience and Prerequisite Knowledge

This guide is for the people who are responsible for administering Direct at your site. This might include administrators for:

- BIOVIA Direct
- Other BIOVIA products
- Microsoft Windows Server
- Linux
- BIOVIA databases
- Oracle RDBMS, especially Oracle SQL*Plus and Oracle Database Server

This guide does not assume that you have any prior knowledge of Direct.

If you are using Oracle Database Server replication functionality, this guide assumes that you are thoroughly familiar with Oracle Database Server replication.

Related BIOVIA Direct documents

You need these BIOVIA Direct documents in order to successfully install and administer Direct:

Document	Purpose
<i>Direct System Requirements</i> <i>Direct Installation & Configuration Guide (Microsoft Windows)</i> <i>Direct Installation & Configuration Guide (Linux)</i> <i>BIOVIA Direct Release Notes</i>	Installing and configuring Direct for your platform
<i>Direct Developers Guide</i> <i>Direct Reference</i>	Developing applications using Direct
<i>BIOVIA Chemical Representation Guide</i>	Reference information on BIOVIA chemical representation
<i>Error Messages Guide</i>	Contains Direct and related-Oracle error messages
<i>CTFile Formats</i>	Describes the file formats used to represent structures. This file is included in the <i>BIOVIA Direct_2021_Documentation.zip</i> file.

Chapter 2:

Setting the Cartridge C\$DIRECT2021 Environment

BIOVIA Direct manages molecules and reactions that are stored in Oracle tables.

This chapter explains how to set the chemical environment that allows you to use Direct 2021.

The BIOVIA Chemical Environment

The BIOVIA chemical environment establishes *business rules* for your company for the representation of chemical structures. Although the default chemical environment is sufficient to represent most chemical structures, you can customize the chemical environment to represent the following types of structures:

- Structures that are annotated with *attached data* (Sgroup data). For more information and examples, see *Attached Data* in *BIOVIA Chemical Representation*.
- Structures that are partially undefined. For more information, see *Tetrahedral Stereochemistry* in *BIOVIA Chemical Representation*.

The following components comprise the customizable chemical environment in Direct:

- BIOVIA periodic table (Ptable)

The Ptable contains information about the atom symbols that Direct allows. Only BIOVIA molfiles that contain symbols listed in the Ptable can be stored in Oracle Database Server.

Note: If you have customized your previous Ptable, you need to incorporate your customization into the updated Ptable for Direct. For more information, see [Customize the Ptable](#).

- Salts definition

The Salts definition specifies salts for the purpose of FLEXMATCH searches when the SAL switch is Off. For more information, see [Customize the Salts definition](#). For information on FLEXMATCH, see *Exact Search (Flexmatch)* in *BIOVIA Chemical Representation*.

- Sgroupfields definition

The Sgroupfields definition specifies the Sgroup fields for the database. If your company business rules require attached data, you must define Sgroup fields so that structures can be correctly registered and searched. For more information, see [Customize the Sgroupfields Definition](#).

Note: The default Sgroup fields definition file is empty. This means that any Sgroup field names are permitted for registration and searching.

- Isotope atomic masses (Isotopedata definition)

The Isotopedata definition specifies the exact atomic masses for common isotopes. These are used by the MONOISOTOPICMASS operator when computing the mono-isotopic molecular weight for a molecule. If your company uses non-standard isotopic masses you may customize the values used by Direct. For more information, see [Customize the Isotopedata Definition](#).

The following definition files are also part of the chemical environment, but should not be customized:

- The **molecule 2D key definition** specifies the molecule keys that are used in substructure and similarity searches.
- The **molecule subset key definition** specifies which of the 2D substructure search keys comprise a key subset. The subset is used only by the MOLKEYS operator, and is not used for searching and registration.

Note: The procedures that are explained in this chapter assume that the Oracle Database Server schema for Direct is C\$DIRECT2021. For information about creating the Oracle Database Server schema that owns Direct, see the *BIOVIA Direct Installation and Configuration Guide* for your platform.

Overview of BIOVIA Global Chemical Environment

The Direct *global* chemical environment applies to all databases at your company. After you install Direct, you must customize the components of the global chemical environment that are required for your company business rules for chemical structure representation so that existing databases will work with Direct.

The global chemical environment is initially defined when you install BIOVIA Direct in Oracle Database Server. The environment files are stored as CLOBs in an Oracle table in the schema C\$DIRECT2021.

You customize each component of the global chemical environment by:

- Writing environment information to a text file
- Editing the file to reflect information that is specific to your site
- Loading the edited file into the global environment

BIOVIA provides these functions to assist you in performing these tasks:

- See [MDLAUX.GETENVFILE](#)
- See [MDLAUX.SETENVFILE](#)

Notes:

If you have previously installed BIOVIA software, but have not installed Direct 2021, you might already have a customized Ptable file or Salts definition file that you might want to use with Direct. Upgrade the previous versions of the customized Ptable or Salts definition files to the Direct format by using the `directupgrade` utility, then point to the locations of the upgraded files.

For more information on global environment files, see [Load Global Environment Files](#), [Customize the Ptable](#), and [Customize the Salts Definition](#).

Writing Global Environment Information to a File

This section explains how to extract global environment definitions and write them to a file.

1. Start SQL*Plus and log in as the Oracle schema user C\$DIRECT2021.

For Windows:

```
>sqlplus C$DIRECT2021/user password
```

For Linux:

```
$sqlplus 'C$DIRECT2021'  
password:abcde
```

2. For each component of the global chemical environment that you want to change, call the function MDLAUX.GETENVFILE to write the current Direct environment information to disk:

```
Syntax:  SELECT MDLAUX.GETENVFILE (' ENV_ITEM' , ' ENV_ITEM_FILE_NAME' )  
FROM DUAL ;
```

where:

ENV_ITEM is a variable that stands for the environment component

`ENV_ITEM_FILE_NAME` is a variable that stands for the full pathname of the environment item file that you will create.

For example, to extract the Salts definition to the file `salts.dat`, enter:

```
SQL> SELECT MDLAUX.GETENVFILE ('SALTS', '/usr/direct/mfiles/salts.dat')
FROM DUAL;
```

3. Exit SQL*Plus:

```
SQL> exit
```

Verify that all users have write permissions to the directory in which the files are created.

See also

BIOVIA Direct Reference Guide

[MDLAUX.GETENVFILE](#)

Editing the Global Environment Information

Use a text editor to open and edit the global environment information. Set the global environment items to values that are specific for your site.

See also

[Customizing Environment Information](#)

BIOVIA Chemical Representation Guide

Loading Global Environment Files

1. Start SQL*Plus and log in as the Oracle schema user C\$DIRECT2021.
2. For each component of the global chemical environment, call the function `MDLAUX.SETENVFILE` to set the Direct chemical environment to the values that are written in the specified file, for example:

```
SQL> SELECT MDLAUX.SETENVFILE ('ENV_ITEM', 'ENV_ITEM_FILE_NAME') FROM
DUAL;
```

where:

- `ENV_ITEM` is a variable that stands for the environment component
- `ENV_ITEM_FILE_NAME` is a variable that stands for the full pathname of the environment item file that contains the environment information.
- For example, to use a customized Ptable with Direct, call `MDLAUX.SETENVFILE` as shown:

```
SQL> SELECT MDLAUX.SETENVFILE('PTABLE',
'/usr/direct/mfiles/ptable.dat') FROM DUAL;
```

Note: Call `MDLAUX.SETENVFILE` with the filename argument set to `NULL` or an empty string to reset the environment file to the Direct default.

3. Exit SQL*Plus:

```
SQL> exit
```

Verify that the Oracle Database Server `extproc` process has read permissions to the directory in which the file `ENV_ITEM_FILE_NAME`, or `/usr/direct/mfiles/ptable.dat` in the preceding example, is created.

The contents of the environment file `/usr/direct/mfiles/ptable.dat` are copied to a CLOB within a table in the Direct schema.

When you load a new environment file, such as the `ptable`, into the Direct cartridge schema it will only be used when creating new domain indexes. It will not be used with existing domain indexes. The existing domain indexes contain private copies of the environment files that were in the cartridge when they were created or previously rebuilt with the 'environment' option. If you wish to use the updated environment with an existing domain index you must rebuild that domain index using the parameter '*environment*'. For example:

```
alter index moltable_molidx rebuild parameters ('environment');
```

ALTER INDEX will rebuild those parts of the domain index that are affected by any changes to the environment files since the time the indexes were created or previously rebuilt with the '*environment*' option. If no changes to the index data are required the ALTER INDEX statement will return immediately, it will not rebuild the index.

You must execute this command for each molecule or reaction domain index that you want to use the new environment files.

See [Parameters for ALTER INDEX](#) for more information on rebuilding the domain index.

See also

[MDLAUX.SETENVFILE](#)

Using Multiple Processors with SSS and RSS Searches

Direct can use multiple processors when performing molecule substructure (SSS) or reaction substructure (RSS) searches. Using multiple processors typically speeds up SSS searches by 10%, however, results will vary depending on the query with some queries showing a much larger improvement. Using multiple processors with a Fastsearch boost file typically speeds up SSS searches by 33%, with some queries showing very large improvements.

The bottleneck for most substructure searches is the speed of Oracle data block I/O, so adding more processors does not greatly affect performance. Additional processors are more effective if the I/O time can be reduced, which can be done either by:

- Allocating sufficient memory to the Oracle buffer cache so that the entire Fastsearch data are held in memory
- Using a Fastsearch boost file

The Direct administrator specifies the number of processors to use with the `MDLAUX.SETPROPERTY ('NTHREADS', 'number-of-processors')` command. For example, to allow all substructure searches to use four (4) processors connect to SQL*Plus as the C\$DIRECT2021 user and execute the command:

```
SQL> EXECUTE MDLAUX.SETPROPERTY('NTHREADS', '4');
```

To disable use of multiple processors, enter a value of NULL as the 'number-of-threads':

```
SQL> EXECUTE MDLAUX.SETPROPERTY('NTHREADS', NULL);
```

See also

[MDLAUX.SETPROPERTY](#)

Customizing Environment Information

This section explains how to customize the environment files that you have extracted from a global chemical environment.

After you have created a customized environment item definition, or know the location of the main definition that is used by other BIOVIA software, load it into the Direct chemical environment as explained in [Load Global Environment Files](#).

The BIOVIA Periodic Table

The BIOVIA periodic table, or Ptable, defines the atom symbols that you can register to your database and provides for optional overrides to the atomic weights defined in Pipeline Pilot Server.

The Ptable comprises three sections:

- Overrides for atomic weights of standard elements in Pipeline Pilot Server (elements H-Lr).
- Additional man-made elements.
- Pseudoatoms, atom symbols that do not correspond to any of the chemical elements.

Format of the BIOVIA Ptable

Each element entry in the Ptable includes 11 items of information on one line. Each item is separated by space from the previous item.

- Atom symbol: must start with a letter and can be followed by up to two more letters or numbers
- Atom name (not used by Direct)
- Van der Waals radius: in angstroms (not used by Direct)
- Covalent radius: in angstroms (not used by Direct)
- First legal oxidation state (1..n, or 0 if none)
- Second legal oxidation state (1..n, or 0 if none)
- Third legal oxidation state (1..n, or 0 if none)
- Fourth legal oxidation state (1..n, or 0 if none)
- Fifth legal oxidation state (1..n, or 0 if none)
- Atomic weight
- Exact mass of the most common isotope

Comments are allowed in the Ptable. They must have a # as the first character of the line. Any text following the # is ignored.

If the atom symbol is specified as one of the elements H through Lr, only the Van der Waals radius, covalent radius, atomic weight and exact mass of the most common isotope are used to override the built-in Pipeline Pilot Client values. The atom name and oxidation states are not overridden.

When specifying pseudoatom symbols, if a symbol might stand alone, set all oxidation states to zero. Otherwise, set them to the values 1 through 5. This prevents implicit hydrogens from being added.

The Direct Default Ptable

```
# Direct periodic table
#
# Each line has the following fields,
# for standard elements only atomic weight or exact mass can be modified:
#
```

```

# symbol name VDW_radius covalent_radius 5X_legal_oxidation_states atomic_
weight exact_mass_most_common_isotope
#
# Standard elements
# Lines are comments unless atomic weight or exact mass is modified
# H Hydrogen 1.09 0.32 1 0 0 0 0 1.007940000 1.007825032
# He Helium 1.40 1.60 0 0 0 0 0 4.002602000 4.002603250
# Li Lithium 1.82 1.31 1 0 0 0 0 6.941000000 7.016004000
# Be Beryllium 1.53 0.91 2 0 0 0 0 9.012182000 9.012182100
# B Boron 1.92 0.82 3 0 0 0 0 10.811000000 11.009305500
# C Carbon 1.70 0.77 4 2 0 0 0 12.010700000 12.000000000
# N Nitrogen 1.55 0.75 3 0 0 0 0 14.006700000 14.003074010
# O Oxygen 1.52 0.73 2 0 0 0 0 15.999400000 15.994914620
# F Fluorine 1.47 0.72 1 0 0 0 0 18.998403200 18.998403200
# Ne Neon 1.54 1.12 0 0 0 0 0 20.179700000 19.992440180
# Na Sodium 2.27 1.66 1 0 0 0 0 22.989770000 22.989769670
# Mg Magnesium 1.73 1.36 2 0 0 0 0 24.305000000 23.985041900
# Al Aluminum 1.84 1.18 3 0 0 0 0 26.981538000 26.981538440
# Si Silicon 2.10 1.10 4 0 0 0 0 28.085500000 27.976926530
# P Phosphorus 1.80 1.06 3 5 0 0 0 30.973761000 30.973761510
# S Sulfur 1.80 1.02 2 4 6 0 0 32.065000000 31.972070690
# Cl Chlorine 1.75 0.99 1 3 5 7 0 35.453000000 34.968852710
# Ar Argon 1.88 1.54 0 0 0 0 0 39.948000000 39.962383120
# K Potassium 2.75 2.06 1 0 0 0 0 39.098300000 38.963706900
# Ca Calcium 2.31 1.74 2 0 0 0 0 40.078000000 39.962591200
# Sc Scandium 2.11 1.44 3 0 0 0 0 44.955910000 44.955910200
# Ti Titanium 2.00 1.32 3 4 0 0 0 47.867000000 47.947947100
# V Vanadium 2.00 1.21 5 4 3 2 0 50.941500000 50.943963700
# Cr Chromium 2.00 1.26 6 3 2 0 0 51.996100000 51.940511900
# Mn Manganese 2.00 1.26 7 6 4 2 3 54.938049000 54.938049600
# Fe Iron 2.00 1.26 2 3 0 0 0 55.845000000 55.934942100
# Co Cobalt 2.00 1.21 2 3 0 0 0 58.933200000 58.933200200
# Ni Nickel 1.63 1.15 2 3 0 0 0 58.693400000 57.935347900
# Cu Copper 1.40 1.17 2 1 0 0 0 63.546000000 62.929601100
# Zn Zinc 1.39 1.25 2 0 0 0 0 65.409000000 63.929146600
# Ga Gallium 1.87 1.26 3 2 0 0 0 69.723000000 68.925581000
# Ge Germanium 2.11 1.27 4 0 0 0 0 72.640000000 73.921178200
# As Arsenic 1.85 1.20 3 5 0 0 0 74.921600000 74.921596400
# Se Selenium 1.90 1.17 2 4 6 0 0 78.960000000 79.916521800
# Br Bromine 1.85 1.14 1 3 5 7 0 79.904000000 78.918337600
# Kr Krypton 2.02 1.60 0 0 0 0 0 83.798000000 83.911507000
# Rb Rubidium 3.03 2.21 1 0 0 0 0 85.467800000 84.911789300
# Sr Strontium 2.49 1.86 2 0 0 0 0 87.620000000 87.905614300
# Y Yttrium 2.00 1.66 3 0 0 0 0 88.905850000 88.905847900
# Zr Zirconium 2.00 1.41 4 0 0 0 0 91.224000000 89.904703700
# Nb Niobium 2.00 1.31 3 5 0 0 0 92.906380000 92.906377500
# Mo Molybdenum 2.00 1.30 2 3 4 5 6 95.940000000 97.905407800
# Tc Technetium 2.00 1.27 7 0 0 0 0 98.000000000 97.907216000
# Ru Ruthenium 2.00 1.16 2 3 4 6 8 101.070000000 101.904349500
# Rh Rhodium 2.00 1.25 2 3 4 0 0 102.905500000 102.905504000
# Pd Palladium 1.63 1.26 2 4 0 0 0 106.420000000 105.903483000
# Ag Silver 1.72 1.34 1 0 0 0 0 107.868200000 106.905093000
# Cd Cadmium 1.58 1.48 2 0 0 0 0 112.411000000 113.903358100

```

# In	Indium	1.93	1.44	3	0	0	0	0	114.8180000000	114.903878000
# Sn	Tin	2.17	1.40	2	4	0	0	0	118.7100000000	119.902196600
# Sb	Antimony	2.06	1.41	3	5	0	0	0	121.7600000000	120.903818000
# Te	Tellurium	2.06	1.37	2	4	6	0	0	127.6000000000	129.906222800
# I	Iodine	1.98	1.33	1	3	5	7	0	126.904470000	126.904468000
# Xe	Xenon	2.16	1.31	0	0	0	0	0	131.2930000000	131.904154500
# Cs	Cesium	3.43	2.46	1	0	0	0	0	132.905450000	132.905447000
# Ba	Barium	2.68	2.01	2	0	0	0	0	137.3270000000	137.905241000
# La	Lanthanum	2.00	1.81	3	0	0	0	0	138.9055000000	138.906348000
# Ce	Cerium	2.00	1.65	3	4	0	0	0	140.1160000000	139.905434000
# Pr	Praseodymium	2.00	1.71	3	4	0	0	0	140.907650000	140.907648000
# Nd	Neodymium	2.00	1.64	3	0	0	0	0	144.2400000000	143.910083000
# Pm	Promethium	2.00	1.63	3	0	0	0	0	145.0000000000	144.912744000
# Sm	Samarium	2.00	1.62	2	3	0	0	0	150.3600000000	151.919728000
# Eu	Europium	2.00	1.85	3	3	0	0	0	151.9640000000	152.921226000
# Gd	Gadolinium	2.00	1.61	3	0	0	0	0	157.2500000000	157.924101000
# Tb	Terbium	2.00	1.59	3	4	0	0	0	158.925340000	158.925343000
# Dy	Dysprosium	2.00	1.59	3	0	0	0	0	162.5000000000	163.929171000
# Ho	Holmium	2.00	1.58	3	0	0	0	0	164.930320000	164.930319000
# Er	Erbium	2.00	1.57	3	0	0	0	0	167.2590000000	165.930290000
# Tm	Thulium	2.00	1.56	2	3	0	0	0	168.934210000	168.934211000
# Yb	Ytterbium	2.00	1.56	2	3	0	0	0	173.0400000000	173.938858100
# Lu	Lutetium	2.00	1.56	3	0	0	0	0	174.9670000000	174.940767900
# Hf	Hafnium	2.00	1.41	4	0	0	0	0	178.4900000000	179.946548800
# Ta	Tantalum	2.00	1.31	5	0	0	0	0	180.947900000	180.947996000
# W	Tungsten	2.00	1.30	2	3	4	5	6	183.8400000000	183.950932600
# Re	Rhenium	2.00	1.28	1	2	4	6	7	186.2070000000	186.955750800
# Os	Osmium	2.00	1.26	2	3	4	6	8	190.2300000000	191.961479000
# Ir	Iridium	2.00	1.27	2	3	4	6	0	192.2170000000	192.962924000
# Pt	Platinum	1.72	1.30	2	4	0	0	0	195.0780000000	194.964774000
# Au	Gold	1.66	1.34	1	3	0	0	0	196.966550000	196.966552000
# Hg	Mercury	1.55	1.49	1	2	0	0	0	200.5900000000	201.970626000
# Tl	Thallium	1.96	1.48	1	3	0	0	0	204.383300000	204.974412000
# Pb	Lead	2.02	1.47	2	4	0	0	0	207.2000000000	207.976636000
# Bi	Bismuth	2.07	1.46	3	5	0	0	0	208.980380000	208.980383000
# Po	Polonium	1.97	1.46	2	4	0	0	0	209.0000000000	208.982416000
# At	Astatine	2.02	1.45	1	3	5	7	0	210.0000000000	209.987131000
# Rn	Radon	2.20	1.90	0	0	0	0	0	222.0000000000	222.017570500
# Fr	Francium	3.48	1.80	1	0	0	0	0	223.0000000000	223.019730700
# Ra	Radium	2.83	2.00	2	0	0	0	0	226.0000000000	226.025402600
# Ac	Actinium	2.00	1.81	3	0	0	0	0	227.0000000000	227.027747000
# Th	Thorium	2.00	1.65	4	0	0	0	0	232.038100000	232.038050400
# Pa	Protactinium	2.00	1.66	4	5	0	0	0	231.035880000	231.035878900
# U	Uranium	1.86	1.42	3	4	5	6	0	238.028910000	238.050782600
# Np	Neptunium	2.00	1.61	3	4	5	6	0	237.0000000000	237.048167300
# Pu	Plutonium	2.00	1.61	3	4	5	6	0	244.0000000000	244.064198000
# Am	Americium	2.00	0.92	3	4	5	6	0	243.0000000000	243.061372700
# Cm	Curium	2.00	0.91	3	0	0	0	0	247.0000000000	247.070347000
# Bk	Berkelium	2.00	0.90	3	4	0	0	0	247.0000000000	247.070299000
# Cf	Californium	2.00	0.89	3	0	0	0	0	251.0000000000	251.079580000
# Es	Einsteinium	2.00	0.88	3	0	0	0	0	252.0000000000	252.082970000
# Fm	Fermium	2.00	0.87	3	0	0	0	0	257.0000000000	257.095099000
# Md	Mendelevium	2.00	0.86	2	3	0	0	0	258.0000000000	258.098425000

```

# No  Nobelium      2.00 0.85 2 3 0 0 0 259.000000000 259.101020000
# Lr  Lawrencium    2.00 0.84 3 0 0 0 0 262.000000000 262.109690000
#
# Additional elements and pseudoatoms
Rf  Rutherfordium  1.58 0.84 1 2 3 4 5 261.000000000 0.000000000
Db  Dubnium        1.58 0.84 1 2 3 4 5 262.000000000 0.000000000
Sg  Seaborgium     1.58 0.84 1 2 3 4 5 266.000000000 0.000000000
Bh  Bohrium        1.58 0.84 1 2 3 4 5 264.000000000 0.000000000
Hs  Hassium        1.58 0.84 1 2 3 4 5 277.000000000 0.000000000
Mt  Meitnerium     1.58 0.84 1 2 3 4 5 268.000000000 0.000000000
Ds  Darmstadtium   1.58 0.84 1 2 3 4 5 280.000000000 0.000000000
Rg  Roentgenium    1.58 0.84 1 2 3 4 5 280.000000000 0.000000000
Cn  Copernicium    1.58 0.84 1 2 3 4 5 285.000000000 0.000000000
Uut  Ununtrium     1.58 0.84 1 2 3 4 5 284.000000000 0.000000000
Fl  Flerovium      1.58 0.84 1 2 3 4 5 289.000000000 0.000000000
Uup  Ununpentium   1.58 0.84 1 2 3 4 5 288.000000000 0.000000000
Lv  Livermorium    1.58 0.84 1 2 3 4 5 293.000000000 0.000000000
Uus  Ununseptium   1.58 0.84 1 2 3 4 5 294.000000000 0.000000000
Uuo  Ununoctium    1.58 0.84 1 2 3 4 5 294.000000000 0.000000000
Pol  PolymerBead   1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Mod  Modification  1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Rgp  R              1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
H2   Hydrogen      2.18 0.64 0 0 0 0 0 2.015880000 2.015650064
H+   Hydrogen      1.09 0.32 0 0 0 0 0 1.007280000 0.000000000
Ala  Alanine       1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Arg  Arginine      1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Asn  Asparagine    1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Asp  Asparagine    1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Cys  Cystine       1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Gln  Glutamine     1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Glu  GlutamicAcid  1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Gly  Glycine       1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
His  Histidine     1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Ile  Isoleucine    1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Leu  Leucine        1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Lys  Lysine         1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Met  Methionine     1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Phe  Phenylalanine  1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Pro  Proline        1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Ser  Serine         1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Thr  Threonine      1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Trp  Tryptophan     1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Tyr  Tyrosine       1.80 0.79 1 2 3 4 5 0.000000000 0.000000000
Val  Valine         1.80 0.79 1 2 3 4 5 0.000000000 0

```

Customize the BIOVIA Ptable

The BIOVIA periodic table or Ptable defines additional atom symbols that you can register to your database and customized atomic weights used when Direct calculates the molecular weight of a molecule.

- Elements H through Lr are defined in the BIOVIA periodic table. Pipeline Pilot can be used to change their atomic weights or other properties used by Pipeline Pilot Server but not by Direct. These

elements are included in the output from MDLAUX.GETENVFILE but are commented out. To change the weight or exact mass of one of these elements, remove the # character from the front of the line and make the desired modification.

- The default Direct Ptable matches the default Pipeline Pilot Server Ptable and adds the man-made elements Rf through Uuo, as well as pseudo-atom symbols found in previous Direct Ptables: Pol, Mod, H2, H+, and Ala through Val.
- Symbols D and T are now properly recognized as isotopes of Hydrogen and handled internally as ^2H , ^3H . Thus, the need for a special pseudoatom in the ptable has been eliminated.
- Symbol X is present in the Pipeline Pilot Server libraries used by Direct and is not present in the default Ptable.
- Symbols R and Z are present in the PPS libraries used by Direct and are not present in the default Ptable.

You can customize the default Ptable to add your own pseudoatom symbols or to use site-specific atomic weights for natural and man-made elements.

To Customize a Ptable Definition

1. Extract the Ptable from the chemical environment to a text file.
Use the procedures explained in [Write Global Environment Information to a File](#).
2. Load the revised Ptable into the Direct global environment.
Use the procedure explained in [Load Global Environment Files](#).

To use a previous Direct 6, 7, or 8 customized Ptable definition

1. Upgrade the previous Ptable using the procedures in [Converting a Previous Direct PTABLE or SALTS File to Direct 9 Format](#).
2. Load the upgraded Ptable into the Direct global environment using the procedure under [Load Global Environment Files](#).

The BIOVIA Salts Definition File

The salts definition is an SDFfile containing counterion definitions and an optional counterion removal code. Each counterion is a separate record in the SDFfile. The counterion removal code is an optional data field entry following the molfile counterion definition. For an SDfile definition, see *BIOVIA CTFfile Formats*.

Use BIOVIA Draw to define each counterion and save as a molfile. Then use an editor to add the counterion molfile to the SDFfile. Terminate each record in the SDFfile with four dollar sign characters on a line by themselves, for example:

```
Dichloride
ACCLDraw04091209072D

  2  0  0  0  0  0  0  0  0  0  0999 v2000

      5.0000   -4.0625    0.0000 c1  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0
      4.9688   -5.2500    0.0000 c1  0  5  0  0  0  0  0  0  0  0  0  0  0  0
M  CHG  2    1   -1    2   -1
M  END
$$$$
```

If no counterion removal code is specified, the counterion is removed from the query only if it is a completely separate fragment. The file that BIOVIA ships with Direct uses only this default.

You can also specify that the counterion should be removed if it is a completely separate fragment, or if it is attached to a fragment apart from the main structure. In this case the entire fragment including the counterion is removed. To specify this option, include a data field named `FragmentFlag` with a value of `F` in the SDF file, for example:

```
Nitrate-1
ACCLDraw04091209252D

  4  3  0  0  0  0  0  0  0  0  0999 v2000
    5.1250   -5.0313    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
    5.1250   -3.9439    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0  0  0
    6.1479   -5.6218    0.0000 O   0  5  0  0  0  0  0  0  0  0  0  0  0  0
    4.1021   -5.6218    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0  0  0
  1  2  2  0  0  0  0
  1  3  1  0  0  0  0
  1  4  2  0  0  0  0
M  CHG  1  3  -1
M  END
>  <FragmentFlag>
F

$$$$
```

where:

`<FragmentFlag>` specifies the field name. `F` is the value for the field. The blank line following the value is required.

The Direct Default Salts Definition File

```
Dichloride
ACCLDraw04091209072D

  2  0  0  0  0  0  0  0  0  0  0999 v2000
    5.0000   -4.0625    0.0000 Cl  0  5  0  0  0  0  0  0  0  0  0  0  0  0
    4.9688   -5.2500    0.0000 Cl  0  5  0  0  0  0  0  0  0  0  0  0  0  0
M  CHG  2  1  -1  2  -1
M  END
$$$$

Chloride
ACCLDraw04091209072D

  1  0  0  0  0  0  0  0  0  0  0999 v2000
    5.0000   -4.0625    0.0000 Cl  0  5  0  0  0  0  0  0  0  0  0  0  0  0
M  CHG  1  1  -1
M  END
$$$$

Bromide
ACCLDraw04091209192D

  1  0  0  0  0  0  0  0  0  0  0999 v2000
    3.0313   -2.6875    0.0000 Br  0  5  0  0  0  0  0  0  0  0  0  0  0  0
M  CHG  1  1  -1
```



```

M END
$$$
Fluoride
ACCLDraw04091209202D

1 0 0 0 0 0 0 0 0 0 0999 v2000
3.3438 -2.5000 0.0000 F 0 5 0 0 0 0 0 0 0 0 0 0
M CHG 1 1 -1
M END
$$$
Lithium
ACCLDraw04091209202D

1 0 0 0 0 0 0 0 0 0 0999 v2000
3.0938 -3.5000 0.0000 Li 0 3 0 0 0 0 0 0 0 0 0 0
M CHG 1 1 1
M END
$$$
Sodium
ACCLDraw04091209202D

1 0 0 0 0 0 0 0 0 0 0999 v2000
2.5313 -2.9688 0.0000 Na 0 3 0 0 0 0 0 0 0 0 0 0
M CHG 1 1 1
M END
$$$
Potassium
ACCLDraw04091209212D

1 0 0 0 0 0 0 0 0 0 0999 v2000
2.4375 -2.7500 0.0000 K 0 3 0 0 0 0 0 0 0 0 0 0
M CHG 1 1 1
M END
$$$
Dicarbonate
ACCLDraw04091209222D

8 6 0 0 0 0 0 0 0 0 0999 v2000
5.1250 -5.0313 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
5.1250 -3.8501 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
6.1479 -5.6218 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0
4.1021 -5.6218 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0
4.0709 -9.4656 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0
6.1166 -9.4656 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0
5.0938 -7.6939 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
5.0938 -8.8750 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
1 2 2 0 0 0 0
1 3 1 0 0 0 0
1 4 1 0 0 0 0
8 5 1 0 0 0 0
8 6 1 0 0 0 0
8 7 2 0 0 0 0
M CHG 4 3 -1 4 -1 5 -1 6 -1

```

```

M END
$$$
Carbonate
ACCLDraw04091209222D

4 3 0 0 0 0 0 0 0 0 0999 v2000
5.1250 -5.0313 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
5.1250 -3.8501 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
6.1479 -5.6218 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0
4.1021 -5.6218 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0
1 2 2 0 0 0 0
1 3 1 0 0 0 0
1 4 1 0 0 0 0
M CHG 2 3 -1 4 -1
M END
$$$
Sulfate
ACCLDraw04091209232D

5 4 0 0 0 0 0 0 0 0 0999 v2000
5.1250 -5.0313 0.0000 S 0 0 3 0 0 0 0 0 0 0 0 0
4.5625 -3.9751 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
6.1479 -5.6218 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0
4.1021 -5.6218 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0
5.8054 -3.9779 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
1 2 2 0 0 0 0
1 3 1 0 0 0 0
1 4 1 0 0 0 0
1 5 2 0 0 0 0
M CHG 2 3 -1 4 -1
M END
$$$
Nitrate-1
ACCLDraw04091209252D

4 3 0 0 0 0 0 0 0 0 0999 v2000
5.1250 -5.0313 0.0000 N 0 0 0 0 0 0 0 0 0 0 0 0
5.1250 -3.9439 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
6.1479 -5.6218 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0
4.1021 -5.6218 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
1 2 2 0 0 0 0
1 3 1 0 0 0 0
1 4 2 0 0 0 0
M CHG 1 3 -1
M END
$$$
Nitrate-2
ACCLDraw04091209252D

4 3 0 0 0 0 0 0 0 0 0999 v2000
5.1250 -5.0313 0.0000 N 0 3 0 0 0 0 0 0 0 0 0 0
5.1250 -3.9439 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
6.1479 -5.6218 0.0000 O 0 5 0 0 0 0 0 0 0 0 0 0

```

```

      4.1021    -5.6218    0.0000 o  0  5  0  0  0  0  0  0  0  0  0  0
    1  2  2  0  0  0  0
    1  3  1  0  0  0  0
    1  4  1  0  0  0  0
M  CHG  3    1    1    3   -1    4   -1
M  END
$$$$
Nitrite

```

ACCLDraw04091209262D

```

    3  2  0  0  0  0  0  0  0  0  0999 v2000
      5.1250    -5.0313    0.0000 N  0  0  0  0  0  0  0  0  0  0  0
      5.1250    -3.9439    0.0000 o  0  0  0  0  0  0  0  0  0  0  0
      5.1021    -6.0593    0.0000 o  0  5  0  0  0  0  0  0  0  0  0
    1  2  2  0  0  0  0
    1  3  1  0  0  0  0
M  CHG  1    3   -1
M  END
$$$$

```

Perchlorate

ACCLDraw04091209322D

```

    5  4  0  0  0  0  0  0  0  0  0999 v2000
      3.7188    -3.6875    0.0000 c1 0  0  0  0  0  0  0  0  0  0  0
      5.0229    -3.6907    0.0000 o  0  0  0  0  0  0  0  0  0  0  0
      3.7188    -4.8686    0.0000 o  0  0  0  0  0  0  0  0  0  0  0
      3.7271    -2.6907    0.0000 o  0  0  0  0  0  0  0  0  0  0  0
      2.6021    -3.7156    0.0000 o  0  5  0  0  0  0  0  0  0  0  0
    1  2  2  0  0  0  0
    1  3  2  0  0  0  0
    1  4  2  0  0  0  0
    1  5  1  0  0  0  0
M  CHG  1    5   -1
M  END
$$$$

```

Hydrochloride

ACCLDraw04091209342D

```

    2  1  0  0  0  0  0  0  0  0  0999 v2000
      3.2813    -2.3125    0.0000 c1 0  0  0  0  0  0  0  0  0  0  0
      4.4624    -2.3125    0.0000 H  0  0  0  0  0  0  0  0  0  0  0
    1  2  1  0  0  0  0
M  END
$$$$

```

Hydrate

ACCLDraw04091209342D

```

    3  2  0  0  0  0  0  0  0  0  0999 v2000
      3.2813    -2.3125    0.0000 o  0  0  0  0  0  0  0  0  0  0  0
      4.4624    -2.3125    0.0000 H  0  0  0  0  0  0  0  0  0  0  0
      2.6907    -3.3354    0.0000 H  0  0  0  0  0  0  0  0  0  0  0
    1  2  1  0  0  0  0

```

```
1 3 1 0 0 0 0  
M END  
$$$$
```

Customize the Salts Definition

An exact match search can retrieve compounds that are salts, hydrates, or parent compounds of your query. However, to be perceived as the counterion of a salt during a search, an atom or fragment must be listed in the Salts definition. If you want to extend the definition of a salt to include more counterions, you can customize the default Salts definition.

Note: You need to customize the Salts definition only if your business rules use a single chemical structure to represent the salt. Other data models might represent salts differently. For example, you might want to store only the parent compound as a chemical structure and store information on counterions and hydrates in a separate database field. You might also define a salt as a substance with the chemical structures that comprise the salt (parent compound, counterion, water of hydration) in different rows within an Oracle database table.

To customize a Salts definition

1. Extract the Salts definition from the chemical environment to a text file.
Use the procedures explained in [Write Global Environment Information to a File](#).
For information on the Salts file format, see [The BIOVIA Salts Definition File](#) and the default salts definition file section.
For example, you have the ability to define the **F** flag. You do that by including a field in the SDF file named *FragmentFlag*, if it has a value of **F** the entire fragment containing the salt is ignored.
2. Load the revised Salts file into the Direct environment.
Use the procedures explained in [Load Global Environment Files](#) or [Customizing Environment Information](#).

To use a previous Direct 6, 7, or 8 customized Salts definition

1. Upgrade the previous salts definition file using the procedures in [Converting a Previous Direct PTABLE or SALTS File to Direct 9 Format](#).
2. Load the upgraded salts definition into the Direct global environment using the procedure [Load Global Environment Files](#).

Customize the Sgroupfields Definition

You can associate data with all or part of a structure. This attached data is also called Sgroup data. It can be numeric or text data. Attached data is chemically significant, that is, attached data can be used to differentiate chemical structures.

Each instance of attached data on a structure is associated with an Sgroup field, which defines what the data means.

The default Sgroupfields definition for Direct is empty. This allows for registration and searching of any data Sgroup field. No action is required unless you want to constrain the ability to register Sgroups. If this is the case, then the definition needs to be set.

Once defined, Sgroup data that contains molecules will cause warnings or errors, such as MDL-0066: Molecule contains an invalid Sgroup data field.

Normally, you would define Sgroup fields in the following circumstances:

- Your business rules for chemical structure representation require attached data. For example, you might want to differentiate structures by attached data on isotopic purity. For examples, see *Attached Data* in the *BIOVIA Chemical Representation Guide*.
- You want your users to be able to register and search structures from third-party applications that create their own molfiles. If these molfiles contain attached data, you must add the associated Sgroup field for each type of attached data to the Sgroupfields definition in your database.

To customize an Sgroupfields definition

Start with the following general steps:

1. Extract the Sgroupfields definition from the chemical environment to a text file.

Use the procedures explained in [Write Global Environment Information to a File](#).

IMPORTANT! The default Sgroupfields definition for Direct is empty. No action is required unless you want to constrain the ability to register Sgroups. If this is the case, then the definition needs to be set.

For example:

```
SELECT MDLAUX.GETENVFILE('SGROUPFIELDS', 'c:\temp\sgroupfields.txt')
FROM DUAL;
```

2. Use the guidelines in *Attached Data* in the *BIOVIA Chemical Representation Guide* to decide the categories of attached data that you want in your database. Each category of attached data must have an associated Sgroup field and data type (text or numeric).
3. Edit the file extracted in step 1, or create a new file so that it contains the desired Sgroup field definitions. See [Modifying the Sgroupfields Definition](#).

To configure BIOVIA Draw for your users, use the procedure in the *Attached Data* chapter of the *BIOVIA Chemical Representation Guide*.

4. Load the revised Sgroupfields file into the Direct environment. Use the procedure explained in [Load Global Environment Files](#) or [Customizing Environment Information](#).

For example (connected to Oracle as the cartridge schema):

```
SELECT MDLAUX.SETENVFILE('SGROUPFIELDS', 'c:\temp\edited_
sgroupfields.txt') FROM DUAL;
```

Allowed Sgroup Field Names and Data Types

MDL_STARATOM_NAME	TEXT
MDL_RESIDUE_ATTACHMENT_ORDER	text
POLYMER_TYPE	text
POLYMER_STEREO	
Isotopic_Purity	n

Note: Sgroup field name definition files which have been created by convertcrg also contain a third entry on each line. This entry specifies the field number, or ID, to use when generating 2D molecule keys for a molecule. If you are modifying an SGROUPFIELDS file which contains these entries, do not remove any FIELDID=n sections, where n represents a number, in the entry. However, do not add FIELDID=n to any new Sgroup field that you add to the file.

A file created by convert.rcg will look like the following example:

MDL_STARATOM_NAME	TEXT	FIELDID=1
MDL_RESIDUE_ATTACHMENT_ORDER	text	FIELDID=2
POLYMER_TYPE	text	FIELDID=3
POLYMER_STEREO	TEXT	FIELDID=4
Isotopic_Purity	NUMERIC	FIELDID=5

Modifying the Sgroupfields Definition

The format of the data Sgroup field names definition is:

```
field-name      field-type      [IGNORECASE]
```

- Specify one field definition per line. Empty lines are ignored.
- Specify the field name using 30 or fewer characters. It may be entered as upper or lower case, case does not matter.
- Specify the field type as either the word NUMERIC or the word TEXT. You may also abbreviate to just N or T. You can also use lower case n or t.
- Add the keyword IGNORECASE (or ignorecase) if you want text searches to case-insensitive. When you specify IGNORECASE, a search will consider SomeTextData as equal to sometextdata.

For example if your company business rules now require two new Sgroup fields names MIXTURE and PURITY, where PURITY is a number from 0 to 100 and MIXTURE is text which is not case-sensitive, the two added lines in the Sgroupfields definition file would look like:

```
mixture text ignorecase
purity numeric
```

Customize the Isotopedata Definition

The isotopedata definition is a file containing information about common isotopes: their exact mass, isotopic composition and half-life. Only the exact-mass is used by Direct in the MONOISOTOPICMASS operator.

To read the default file, see [Appendix E: The Direct Default Isotopedata Definition File](#).

The format of the file is a single line header followed by one line for each isotope. Each line contains the integer atomic number, the atom symbol, the mass number of the isotope, the exact atomic mass, the isotopic composition and the half life.

If your company requires a different value for the exact atomic mass of one or more isotopes you can customize the isotopedata definition.

To Customize a Isotopedata Definition

1. Extract the Isotopedata definition from the chemical environment to a text file.
Use the procedures explained in [Write Global Environment Information to a File](#).
2. Edit the file and modify only the value in the fourth column, *AtomicMass*, for the isotope to be changed. For example, you could replace the value for carbon 10, 10.0168531, with a new value 10.0168600.

3. Load the revised Isotopedata into the Direct global environment.
Use the procedure explained in [Load Global Environment Files](#) or [Customizing Environment Information](#).

Chapter 3:

Creating and Managing Indexes and Tables

Note: Use of partitioned tables and indexes requires Oracle Enterprise Edition.

Creating Molecule Indexes and Tables

The following sections explain how to create and drop molecules and their domain indexes using Direct.

Molecule Index Without Generic Structures

This type of index is applicable to most corporate registry systems. This index can only store molecules that:

- are not Markush structures.
- do not contain Rgroups or Rgroup members.

The index only stores 2D molecules, models with 3D coordinates are flattened to 2D (z coordinate set to zero) upon registration. Biopolymer sequence molecules can be stored in the index.

Create the index either without specifying an index type, or by specifying a type of MOLECULES.

```
create index molindex on moltable (ctab)
indextype is C$DIRECT2021.mxixmdl;
```

or

```
create index molindex on moltable (ctab)
indextype is C$DIRECT2021.mxixmdl
parameters ('molecules');
```

Oracle Database Server calls Direct to create a domain index. Direct, in turn, creates the objects that are explained in this section. For information on the PARAMETERS string, see [Parameters for Index Creation and Maintenance](#).

Note: The molecule column must be of type BLOB.

Use the:

- MOL operator to insert molecules into a table.
- SSS operator to perform a substructure search. This search can use either Fastsearch technology or inverted key screening. You can also use the SSSTIMEOUT and SSSHIGHLIGHT ancillary operators.
- FLEXMATCH operator to perform an exact match or a flexible match, which allows tautomers to match, salts to match, and so on. You can also use the FLEXMATCHHIGHLIGHT and FLEXMATCHTIMEOUT ancillary operators .
- SIMILAR operator to perform a similarity search which uses substructure search keys to compute similarity. You can also use the SIMILARITY ancillary operator.
- FMLALIKE and FMLAMATCH operators to perform formula searches.

For example, to create a domain index which stores all indexing data in tablespace USERS, issue CREATE INDEX as shown:


```
SQL> CREATE INDEX MOLTABLE_IX ON MOLTABLE (COL)
      2> INDEXTYPE IS C$DIRECT2021.MXIXMDL
      3> PARAMETERS ('TABLESPACE=USERS');
```

where:

INDEX is the name of the domain index that you want to create.

MOL_TABLE is the name of the table on which you want to create the domain index.

MOL_COLUMN is the name of a BLOB column that holds the molecules to be indexed.

PARAMETERS are the parameters that you specify, such as, for example, the maximum number of rows that the table is expected to contain.

When you specify an INDEXTYPE of C\$DIRECT2021.MXIXMDL, Direct creates a domain index.

You can create synonyms for functions and operators. The INDEXTYPE of MXIXMDL must always be prefixed with the Direct schema name C\$DIRECT2021.

Note: If you run CREATE INDEX or ALTER INDEX on a table that contains many rows of data, the operation can be relatively slow. Status information during index building or rebuilding is written to the local index log table. To view the status information, open another connection to Oracle Database Server and call MDLAUX.LOGTABLE. This function queries the local index log table to determine the status of the operation. Entries are written to the local index log table and committed periodically.

Markush Molecule Index

A generic molecule index can store entire libraries of 2D molecules in one record, as a generic (Markush) molecule structure. This type of index is used as a compact way to store a large number of molecules that have varying substituents on a common core. This index can store both generic and non-generic molecules. It is a superset of the molecule index described in Molecule index without generic structures.

Note: Because of additional indexes for Markush features, substructure searches of normal molecule will be slightly slower than with a non-generic molecule index.

Create the index by specifying a type of GENERICS:

```
create index molindex on moltable (ctab)
indextype is C$DIRECT2021.mxixmdl
parameters ('generics')
```

Use the:

- MOL operator to insert molecules into a table.
- SSS operator with GENERICS in its third argument to perform a substructure search which selects records in the index that contain generic structures. This search uses Fastsearch technology. Key screening is not available with generic substructure searching. You can also use the SSSTIMEOUT ancillary operator .
- SSS operator without a third argument to perform a normal molecule substructure search. This search can use either Fastsearch technology or inverted key screening. You can also use the SSSTIMEOUT and SSSHIGHLIGHT ancillary operators.
- FLEXMATCH operator with GENERICS as its third argument to perform a generic exact-match search. The normal FLEXMATCH switches are ignored, and the search always uses IGNORE=DAT (data Sgroups must be ignored because only Fastsearch is used for matching). You can also use the FLEXMATCHTIMEOUT ancillary operator.

- FLEXMATCH operator without GENERICS in its third argument to perform an exact match or a flexible match that allows tautomers to match, salts to match, and so on. The query cannot be a generic structure, and the search does not select any generic structures as hits. You can also use the FLEXMATCHHIGHLIGHT and FLEXMATCHTIMEOUT ancillary operators.
- OVERLAP operator to perform a generic overlap search, which finds records that have at least one enumerated specific structure in common with the query. You can also use the PCTOVERLAP and OVERLAPTIMEOUT ancillary operators.
- SIMILAR operator to perform a similarity search that uses substructure search keys to compute similarity. The query cannot be a generic structure, and the search does not select any generic structures as hits. You can also use the SIMILARITY ancillary operator.
- FMLALIKE and FMLAMATCH operators to perform formula searches.

Creating Molecule Tables

Note: Before users can create and manage tables and indexes in Direct, the Oracle Database Server user SYS must log in AS SYSDBA and run the script `mdl sysactions.sql`, which is provided by BIOVIA. The script performs these tasks:

- Grant EXECUTE on DBMS_LOCK to PUBLIC;
Direct uses the DBMS_LOCK package to manage access by multiple users to domain index maintenance routines.
- Grant SELECT on ALL_SECONDARY_OBJECTS to PUBLIC;
This might be required at your site. Some versions of Oracle Database Server do not permit all users to access the ALL_SECONDARY_OBJECTS view. Direct uses this view to determine whether Oracle Database Server has added secondary objects to the SYS . SECOBJ\$ table.
- Create procedure MDLINSERTSECOBJ and grant EXECUTE on it to PUBLIC;
This procedure ensures that domain index secondary tables are properly defined in the SYS . SECOBJ\$ table after using the Oracle Database Server Import utility or after a call to MDLAUX . RECREATEINDEXES.

When you create a molecule table or an index on a molecule table, remember that:

- Molecules are stored as BLOBs.
- The average size of a molecule BLOB is approximately 2KB.
- Most molecules are smaller than 4KB.

Note: BLOBs larger than 4 KB are not stored in the table at all, but in a separate segment.

BLOBs which are less than or equal to 4 KB in size can be stored in-line, that is, physically adjacent to the rest of the columns in the row or out-of-line in a different segment from the rest of the columns. Molecules tend to be fairly small, which makes it feasible to store them in-line. To determine the table storage that a molecule table requires, use the preceding guidelines. For more information, see your Oracle Database Server documentation.

Examples

The examples that follow show you how to use SQL*Plus commands to manage Direct.

Before you can use SQL commands to manage Direct, verify that you have configured your Direct environment correctly. Also verify that you have all of the privileges that are required. For information

on configuration and required privileges, see the *BIOVIA Direct Installation and Configuration Guide* for your platform.

Perform any tasks that are required in the schema that you plan to use.

Create a Table to Contain Molecules

When you create a table to contain molecules, include a column of type BLOB. The example that follows shows you how to issue a SQL statement that creates a molecule column:

```
CREATE TABLE TABLE (IDENT VARCHAR2(30), MOL_COLUMN BLOB);
```

where:

- TABLE represents the table that you create
- MOL_COLUMN represents the molecule column name

Oracle Database Server creates a table that you can use to store and search molecule data.

Add a Domain Index to the Molecule Table

Note: Creating a domain index is an optional task. However, when you create an index on your molecule column, search speeds increase significantly if the table contains more than a few hundred rows of data.

IMPORTANT!

SQL commands such as ALTER TABLE ... MOVE that reorganize or move the contents of the molecule or reaction table might result in modified ROWIDs for some or all rows in the molecule or reaction table. The domain index depends on the values of the ROWIDs of these rows and as a result, might become invalid after the operation. If it is necessary to use commands like ALTER TABLE... MOVE, which alter the record ROWIDs, you must perform a domain index rebuild after to restore proper functionality to the domain index.

Because the domain index depends on the ROWIDs of the records, when you copy the contents of a reaction or molecule table to another schema, you should create the domain index using a CREATE INDEX command, and not try to transport the data content of the existing domain index tables. The data content of domain index tables is only valid for the reaction or molecule table for which the index was originally created.

To enhance molecule search performance as the table grows larger, add a domain index to the BLOB column, as follows.

```
SQL> CREATE INDEX TABLE_MOLIX ON TABLE (MOL_COLUMN)
2> INDEXTYPE IS C$DIRECT2021.MXIXMDL
3> [PARAMETERS ('PARAMETERS')];
```

Note: The Oracle Database Server uses the term domain index to identify an Oracle Database Server index of type DOMAIN.

A domain index has an associated indextype, which is owned by a particular Oracle Database Server user. The Direct indextype is MXIXMDL. The owner of the indextype is the user C\$DIRECT2021.

Create a domain index whenever the table in which you store molecule data contains more than two to three hundred rows of data. For more information about domain indexes, see [Create a Domain Index](#).

For example:

```
SQL> CREATE INDEX CORPDB_MOLIX ON CORPDB_MOL (CTAB) INDEXTYPE IS
C$DIRECT2021.MXIXMDL;
```

Domain index maintenance is performed using ALTER INDEX and four MDLAUX procedures. For information about domain index maintenance and the procedures and functions that BIOVIA supplies for such purposes, see [Maintaining Tables and Indexes](#).

You can also specify parameters that control how the molecule index is created. For more information about the use of parameters when creating a molecule index, see [Parameters for Index Creation and Maintenance](#) and the sections that follow.

Periodically as information is added to the tables, you will need to update pending inversions and pending Fastsearch data to ensure that search speed is optimal. For more information on performing such index maintenance, see [Update a Fastsearch Index](#) and [Update the Inverted Key Index](#).

Domain Index Creation

During the indexing process, Direct fetches each structure from the structure table and processes it to create entries in the secondary objects which make up the domain index. For information, see [Structure of a Domain Index](#). The overall process consists of the following steps:

1. Fast scan of the structure table to locate all rows containing non-NULL, non-empty structure data, the ROWID and a sequence-generated internal ID number are inserted into the *IXNAME_CONV* table.
2. Slower scan of the structure table ordered by ID value from *IXNAME_CONV* table. Each structure is fetched and indexing information is generated and stored in secondary tables. Fastsearch data are staged in temporary files during this step, with periodic merges from the temporary files into the *IXNAME_FSIX* table.
3. Final merge of Fastsearch data into the *IXNAME_FSIX* table.

You can locate the failed records from the ROWIDs stored for each of the records using MDLAUX.LOGTABLE, for example:

```
SQL> column msg format a75
SQL> select msg from table(mdlaux.logtable('test','asc'));
MSG
-----
-----
Building new index
Table contains 100 rows
COMMITCOUNT=50000 (default)
Inserted ROWIDs into conversion table in 0.1 seconds
MERGECOUNT=3500000 (default)
TEMPDIR is operating system default
FSUPDCACHE=33554432 (index environment value)
FSUPDBLOCKSIZE=65536 (index environment value)
MDL-1934: Unable to read binary molecule: Inflate failed, status = -3.
vcCompress::Decompress: Pipeline Pilot exception rethrown
MDL-1050: Unable to unpack molecule at ROWID=AAExCHAAGAAEpHmAAA
MDL-2241: Molecule will not be indexed at ROWID=AAExCHAAGAAEpHmAAA
Processing row 100
Merging final FASTSEARCH data
Creating Oracle indexes
Index build complete with 1 row failing indexing, processed 100 rows
in 1.1
seconds
```

You can either delete those records or update them using chemical structures obtained from the original source for the records.

If an Oracle I/O error occurs during any of the steps, index creation is terminated. Oracle may still create the index. However, it will not be valid and should be dropped if it exists.

If an error occurs while processing chemistry during step 2, in most cases index creation will not be terminated. Messages describing the problem will be placed on the message stack and also written to the `IXNAME_LOG` table where they may be viewed after index creation completes using `MDLAUX.LOGTABLE`. No data will be written to the index for this record, and the entry containing the record's ROWID will be removed from the `IXNAME_LOG` table.

The failing record will not be found with an indexed structure search, however a non-indexed search may or may not find the record and may also generate errors. The structure in the record should be corrected and updated as soon as possible.

Biopolymer Searching in Direct

Note: Biopolymer searching applies only to molecules. Biopolymer sequences for reactions are not supported.

Over the years, chemical structure databases have used many formats for storing biopolymer molecules. The two most important formats in use today are full CTAB and SCSR:

- Full CTAB stores every atom and bond of the biopolymer molecule in the database. A protein with 100 amino acids will have around 900 atoms and 900 bonds which are stored in the molecule connection table (CTAB) inside the database. Sequence monomer information is encoded using abbreviation Sgroups (superatoms) which wrap each monomer and its leaving groups.
- SCSR (Self Contained Sequence Representation) stores one atom for each monomer in the biopolymer, plus the atoms and bonds for each distinct monomer that is present in the biopolymer. A protein with 100 amino acids, of which 20 are distinct, will have 100 SCSR template atoms and about 100 bonds between those atoms, plus about 180 additional atoms and bonds defining the monomers.

The HELM format is becoming popular as a way of entering biopolymers into the database, but it is not suitable for searching and must be converted to a standard molecule connection table during structure registration.

The SCSR format is generally preferred because it allows for storage of larger sequences. It is also easier to interconvert SCSR with the other formats including HELM.

While these molecule connection table formats are interconvertible, generally speaking a structure in one format will not match the logically identical structure in a different format in an exact match or substructure search. Chemists creating databases using a modern version of BIOVIA Draw will generally use SCSR format. If their database also contains legacy structures in full CTAB format, a search using one format will usually not match the same biopolymer stored in the other format.

Starting with the BIOVIA Direct 2018 release, "small" sequences entered in SCSR format are stored as both SCSR structures and as full CTAB structures, which provides a partial solution to this searching problem. As long as all of the sequences stored in the database are below the expansion cutoff (the value of "small"), structure searches will work regardless of the format used to store the structures.

The definition of "small" is provided by the user when the BIOVIA Direct domain index is created, either as a maximum monomer count or a maximum molecular weight. In practice, we have found that performance becomes unacceptable when it exceeds about 100 monomers.

Storing both formats requires more storage space and processing time but allows queries and targets that do not exceed the threshold (the value of "small") to match each other regardless of the original structure format.

When an input structure is stored in SCSR format, all nucleic acid monomers and any amino acid monomers that have been modified (not present in the standard list of global SCSR templates) or are cross-linked are expanded on the fly into full CTAB units during indexing and searching. On-the-fly expansion has been available since BIOVIA Direct version 8. Starting with BIOVIA Direct 2018, all SCSR input structures below a certain size cutoff have all monomers expanded into full CTAB format, and the expanded full CTAB structure is stored in the index so that on-the-fly expansion is not required during matching.

Starting with the BIOVIA 2020 release, options are available to ignore the 3' and 5' terminal phosphates in substructure and exact matches, allowing nucleic acids represented with traditional SCSR template classes to match structures created from HELM format as long as the sugars, bases, and connecting phosphates are the same.

To see how queries match or do not match targets, see the [Flexmatch Searching](#) tables.

See also

[Specifying the Cutoff Size for SCSR Template Expansion](#)

[FLEXMATCH Searching](#)

[Substructure Searching](#)

[Computing Sequence Text](#)

Specifying the Cutoff Size for SCSR Template Expansion

You specify the size cutoff for SCSR template expansion when creating an index by adding a new parameter named `SEQUENCE_EXPAND_LIMIT`. You can also change the expansion limit at any time by rebuilding the index and adding this parameter with a different value. You can view the current value for the parameter by selecting from the SQL function `MDLAUX.INDEXPARAMETERS`.

By default there is no automatic expansion of standard (not modified) SCSR templates during registration or searching. This results in the same searching behavior as Direct 2017 R2 and earlier versions.

To enable automatic expansion, choose a threshold below which you want all SCSR templates expanded into full chemistry. You can specify the threshold as either a number of monomer units or as a molecular weight. For peptide and nucleic acid sequences we recommend keeping the threshold below 100 monomers or 10000 atomic mass units (AMU). Registration performance and index size are both negatively affected with larger thresholds.

To specify the threshold as a number of monomers use:

```
CREATE INDEX indexname ON tablename (columnname) INDEXTYPE
IS C$DIRECT2021.MXIXMDL PARAMETERS ('SEQUENCE_EXPAND_LIMIT=nMONOMERS');
```

Replace `n` with the desired number of monomers, for example 60. The suffix `MONOMERS` is optional. For example:

```
create index biomol_molix on biomol (ctab) indextype is
c$direct2021 parameters ('sequence_expand_limit=50');
```

To specify the threshold as a molecular weight use the suffix `AMU` instead:

```
CREATE INDEX indexname ON tablename (columnname) INDEXTYPE
IS C$DIRECT2021.MXIXMDL PARAMETERS
('SEQUENCE_EXPAND_LIMIT=nMONOMERS');
```

Replace m with the molecular weight limit, for example:

```
create index biomol_molix on biomol (ctab) indextype is
c$direct2021 parameters ('sequence_expand_limit=5000amu');
```

To change the threshold on an existing index use ALTER INDEX REBUILD:

```
ALTER INDEX indexname REBUILD PARAMETERS
('SEQUENCE_EXPAND_LIMIT=nMONOMERS');
```

or

```
ALTER INDEX indexname REBUILD PARAMETERS
('SEQUENCE_EXPAND_LIMIT=mAMU');
```

To remove the threshold from an index and revert to the Direct 2017 R2 and earlier behavior where SCSR templates are not expanded, use the parameter value NONE:

```
ALTER INDEX indexname REBUILD PARAMETERS
('SEQUENCE_EXPAND_LIMIT=NONE');
```

FLEXMATCH Searching









The following two tables show the difference in FLEXMATCH searches for Direct 2021 with either small queries (below the size cutoff for expansion) or large queries (above the size cutoff for expansion). Large queries (above the size cutoff for expansion) in Direct 2021 behave the same as all queries do in previous versions of Direct.

Exact is an exact match, i.e. FLEXMATCH(CTAB, 'query', 'MATCH=ALL').







Flexible is a non-exact match, e.g. FLEXMATCH(CTAB, 'query', 'IGNORE=MAS').

GREEN indicates that matching works correctly between query and target, RED indicates that matching either fails with an error or does not provide logically equivalent molecules as hits.

Query smaller than size cutoff for expansion

	Target as full CTAB	Target as SCSR
Query as full CTAB, exact		
Query as SCSR, exact		
Query as full CTAB, flexible		
Query as SCSR, flexible		

Previous versions of Direct, or query larger than size cutoff for expansion

	Target as full CTAB	Target as SCSR
Query as full CTAB, exact		
Query as SCSR, exact		
Query as full CTAB, flexible		

Substructure Searching

Substructure queries which are in SCSR format and consist entirely of unmodified amino acid monomers (i.e. monomers which are present the standard list of SCSR global templates) are handled as a text substring search. The sequence text is generated for the query and a text column within the domain index is queried. This will find all targets regardless of size, as long as Direct could compute the sequence text for the target when it was registered. See [Computing Sequence Text](#) for more information, it is not always possible to generate sequence text for full CTAB input structures.

















The following two tables show substructure searching in Direct 2021 and in previous versions.

"Small" refers to sequence molecules which are below the expansion cutoff.

"Large" refers to sequence molecules which are above the expansion cutoff.

Green indicates that matching works correctly between query and target, red indicates that matching does not provide logically equivalent molecules as hits, and yellow indicates that matching provides some logically equivalent molecule as hits but may miss some valid hits. No color indicates that the search is not applicable (query larger than target).

Direct 2018

	"Small" target as full CTAB	"Small" target as SCSR	"Large" target as full CTAB	"Large" target as SCSR
Non-sequence query (assumed small)				
"Small" sequence query as full CTAB				
"Small" sequence query as SCSR				
"Large" sequence query as full CTAB				
"Large" sequence query as SCSR				

Previous Versions of Direct

	"Small" target as full CTAB	"Small" target as SCSR	"Large" target as full CTAB	"Large" target as SCSR
Non-sequence query (assumed small)				
"Small" sequence query as full CTAB				
"Small" sequence query as SCSR				
"Large" sequence query as full CTAB				
"Large" sequence query as SCSR				

When using an SCSR or full CTAB sequence structure as a substructure query, monomer leaving groups are removed from the input structure before executing the search. Thus the terminal hydroxyl is removed at the end of each chain allowing the query to match a target anywhere in the sequence and not just at the end of it. In a normal SSS query the presence of the OH at the end of the query would require a matching OH in the target, i.e. the end of a chain.

In addition to the terminal hydroxyl, other leaving groups are also removed. Hydrogen leaving groups would not normally be attached and are only removed if present.

Queries which are not in SCSR format and which do not contain abbreviation Sgroups (superatoms) to encode monomer information are not interpreted as biopolymer sequences and will not have any leaving groups removed. This can be useful when you want to match the N or C terminus of a chain, and is essential when using query features such as atom lists, markush, etc.

If you have structures which should be interpreted as biopolymer sequences but which are not in one of the two sequence formats, you can use tools in Pipeline Pilot 2021 to perceive the monomers and generate the SCSR format for the structure. If the structures can be represented as HELM strings you can also convert HELM into SCSR using Direct functions. Please contact Customer Support for assistance.

Computing Sequence Text

Direct computes the sequence text representation during registration for structures which are in one of the two sequence formats - SCSR or full CTAB with abbreviation Sgroups (superatoms). To compute the correct ordering of monomer letters, the full CTAB format must usually contain sequence ID numbers, the relative position of monomers within the sequence. Only in very simple cases can the sequence text be computed for sequences which do not contain sequence IDs. When the text cannot be computed a warning generated during registration or index creation:

```
MDL-2421: warning: Unable to generate sequence text for molecule without
sequence IDs
```

The structures are registered, but will not be retrieved with the `SEQUENCESEARCH` operator or with the `SSS` operator when the query is a completely unmodified SCSR sequence.

You can use tools in Pipeline Pilot 2021 to re-perceive the sequence information for these structures and generate sequence ID values. If you have the HELM string representation for these structures you can also convert HELM into SCSR using Direct functions. Please contact Customer Support for assistance.

Fingerprint Searching

Direct supports an indexed similarity search type, which allows the user to create an index using either Accord or Scitegic fingerprints. The structure of the index is identical to that used for the traditional substructure key similarity search with differences in the length of the bit vector and the meaning of each bit.

Creating a Fingerprint Index

CREATE INDEX and ALTER INDEX REBUILD support a FINGERPRINT parameter that creates a second inverted key index for the user-defined fingerprint. Use the form FINGERPRINT=type[,size]. The supported types are Accord and Scitegic fingerprint.

- Accord: Set the type to accord. The fingerprint keys are similar to Accord, but with differences in chemical perceptions such as aromaticity. For this type, do not specify the size, which is always 384 bits. For example:

```
create index testix on test (ctab) indextype is c$direct2021.mxixmdl
parameters ('fingerprint=accord');
```

- Scitegic fingerprint: Set the type to one of the standard names such as ecfp_6 or fcfp_4. The Scitegic fingerprint keys are folded to the desired size (number of bits), and then rounded up to a multiple of 32. If a size is not specified, 512 bits is used by default. For example, the following creates an index with ECFP_4 fingerprints stored in 1024 bits (1000 rounded up to a multiple of 32):

```
create index testix on test (ctab) indextype is c$direct2021.mxixmdl
parameters ('fingerprint=ecfp_4,1000');
```

Notes:

- Methods that update pending inversions will operate on the normal substructure and similarity keys, and the fingerprint keys.
- Methods that rebuild keys will only rebuild the substructure and similarity keys. To rebuild the fingerprint keys by themselves, use the FINGERPRINT=type[,size] parameter in an ALTER INDEX REBUILD statement.

Fingerprint Similarity Searching

To perform a fingerprint similarity search, add the keyword FINGERPRINT to the argument to the MOLSIM or SIMILAR search query. For example:

```
select cdbregno, similarity(1) from test where similar(ctab,
'/tmp/caffeine.mol', '80 fingerprint', 1) =1 order by similarity(1);

select cdbregno from test where molsim(ctab, '/tmp/caffeine.mol', 'normal
fingerprint') >= 80;
```

Note: The SUB and SUPER keywords also support fingerprint similarity.

An application can force the use of the FINGERPRINT search at the Oracle session level or for all users of the cartridge whether or not the fingerprint keyword is not present. For example:

```
execute mdlaux.setflags('ForceFingerprintSearch');
```

```
execute mdlaux.setproperty('ForceFingerprintSearch', 'True'); -- Execute as
user C$DIRECT2021
```

Notes:

- The use of substructure keys for similarity cannot be forced.
- Once 'ForceFingerprintSearch' has been specified, all similarity searches will use the user-defined fingerprint type.

Creating and Managing Reaction Tables and Indexes

The following sections explain how to create and drop reaction tables and their domain indexes using Direct.

Creating Reaction Tables

Note: Before users can create and manage tables and indexes in Direct, the Oracle Database Server user SYS must log in AS SYSDBA and run the script `mdl sysactions.sql`, which is provided by BIOVIA. The script performs the following tasks.

- Grant EXECUTE on DBMS_LOCK to PUBLIC
Direct uses the DBMS_LOCK package to manage access by multiple users to domain index maintenance routines.
- Create procedure MDLINSERTSECOBJ and grant EXECUTE on it to PUBLIC;
This procedure ensures that domain index secondary tables are properly defined in the SYS.SECOBJ\$ table after using the Oracle Database Server Import utility or after a call to MDLAUX.RECREATEINDEXES.

When you create a reaction table or an index on a reaction table, remember that:

- Reactions are stored as a BLOB.
- The average size of a reaction BLOB is approximately 4KB.

Note: A BLOB larger than 4 KB is not stored in the table at all, but in a separate segment.

BLOBs which are less than or equal to 4 KB in size can be stored in-line, that is, physically adjacent to the rest of the columns in the row, or out-of-line, in a different segment from the rest of the columns. To determine the table storage that a reaction table requires, use the preceding guidelines. For more information, see your Oracle Database Server documentation.

Create a Table to Contain Reactions

When you create a table to contain reactions, include a column of type BLOB. The example that follows shows you how to issue a SQL statement that creates a reaction column:

```
CREATE TABLE TABLE (IDENT VARCHAR2(30), RCTAB BLOB);
```

where:

- *TABLE* represents the table that you create.
- *RCTAB* represents the reaction column name.

Example:

```
CREATE TABLE CORPDB_RXN (CORPID VARCHAR2(12), REFNUM NUMBER, RCTAB BLOB);
```

Oracle Database Server creates a table that you can use to store and search reaction data. The RCTAB column is used to contain reactions.

Insert Reactions into a Table

To insert reactions, use the RXN operator to convert a rxnfile to a packed reaction BLOB.

Example

```
\INSERT INTO CORPDB_RXN VALUES ('ABC-123', 123, RXN('/usr/rxns/rxn123.rxn'));
```

For more information about the RXN operator, see the *BIOVIA Direct Developers Guide* and the *BIOVIA Direct Reference Guide*.

You can also insert packed reaction BLOBs directly from another table. You do not need to use the RXN operator in this case, because the reaction is already correctly packed into a BLOB:

Example

```
INSERT INTO CORPDB_RXN VALUES ('ABC-456', 456, (SELECT RCTAB FROM TEST_RXN  
WHERE TESTID = 456));
```

Add a Domain Index to the Reactions Table

To enhance reaction search performance as the table grows larger, add a domain index to the BLOB column, as follows.

```
SQL> CREATE INDEX TABLE_RXNIX ON TABLE (RCTAB)  
2> INDEXTYPE IS C$DIRECT2021.RXIXMDL  
3> [PARAMETERS ('PARAMETERS')];
```

The Oracle Database Server uses the term domain index to identify an Oracle Database Server index of type DOMAIN.

The reaction column must be of type BLOB.

You can create synonyms for functions and operators. However, you cannot create a synonym for schema.RXIXMDL. The INDEXTYPE of RXIXMDL must always be prefixed with the Direct schema name C\$DIRECT2021.

If you run CREATE INDEX or ALTER INDEX on a table that contains many rows of data, the operation can be relatively slow. Status information during index building or rebuilding is written to the local index log table. To view the status information, open another connection to Oracle Database Server and call MDLAUX.LOGTABLE. This function queries the local index log table to determine the status of the operation. Entries are written to the local index log table and committed periodically.

A domain index has an associated indextype which is owned by a particular Oracle Database Server user. The Direct indextype is RXIXMDL. The owner of the indextype is the user C\$DIRECT2021.

Create a domain index whenever the table in which you store reaction data contains more than two to three hundred rows of data. For more information about domain indexes, see [Create a domain index](#).

Example

```
SQL> CREATE INDEX CORPDB_RXNIX ON CORPDB_RXN (RCTAB)  
INDEXTYPE IS C$DIRECT2021.RXIXMDL;
```

Domain index maintenance is performed using ALTER INDEX and four MDLAUX procedures. For information about domain index maintenance and the procedures and functions that BIOVIA supplies for such purposes, see [Maintaining Indexes and Tables](#).

You can also specify parameters which control how the reaction index is created. For more information about the use of parameters when creating a reaction index, see [Structure of a Domain Index](#), and the sections that follow.

Periodically as information is added to the tables, you need to update pending inversions and pending Fastsearch data to ensure that search speed is optimal. For more information on performing such index maintenance, see [Update a Fastsearch index](#), [Domain Index Creation](#) and [Update the inverted key index](#).

If CREATE INDEX Fails

If the `CREATE INDEX` command fails due to error, Oracle Database Server still creates the index, but marks it `INVALID`. If this occurs, drop the index as shown:

```
DROP INDEX INDEX;
```

To examine any errors that caused the error, issue the statement that follows:

```
SELECT MDLAUX.ERRORS FROM DUAL;
```

Resolve the error. For more information on the error, see the *BIOVIA Direct Error Messages Guide* and your Oracle Database Server Error Messages documentation.

Structure of a Domain Index

When you use Direct to create an instance of a domain index, Direct also creates these objects:

- Sequence
- Tables
- Indexes

These objects enable faster molecule structure, reaction structure, and formula searching. The [Example objects](#) section briefly explains these objects.

Note: The objects are created in the schema that you specify. If you do not specify a schema, Oracle Database Server creates these objects in the current user schema.

Example Objects

This section lists the objects created by Direct. The examples that follow assume that the domain index name is `IXNAME`.

IMPORTANT! Do not attempt to insert, update, delete, or alter these objects. These objects are maintained by Direct. Do not create applications which select or otherwise use these objects. These objects might change in subsequent versions of Direct, thereby rendering your applications invalid.

Direct creates a sequence as follows:

Sequence: `IXNAME_SQNC 1 . 4294967295`

where `IXNAME_SQNC` is an Oracle Database Server sequence with values ranging from 1 through 4294967295. This sequence creates a unique integer to use in inverted keys and to provide isolation from the molecule `ROWID`.

Direct creates the tables that follow.

The `IXNAME_PROP` table contains information about the domain index.

Table:	IXNAME_PROP	(PROPERTY_NAME VARCHAR2(30) ,
		PROPERTY_VALUE VARCHAR2(4000) ,
		PROPERTY_CLOB CLOB)

where *IXNAME_PROP* is an Oracle table that contains:

- Name of property
- Value of property

The *IXNAME_CONV* table supports conversion from domain index REGNOs to molecule ROWIDs, and from molecule ROWIDs to domain index REGNOs.

Table:	IXNAME_CONV	(RID ROWID or UROWID(n), REGNO NUMBER(10))
Index:	IXNAME_CONV1	REGNO ASC
Index:	IXNAME_CONV2	RID ASC

where:

- *IXNAME_CONV* is a table that maps row identification numbers (ROWIDs) to registration numbers (REGNOs)
- *IXNAME_CONV1* and *IXNAME_CONV2* are indexes on the table *IXNAME_CONV*

The *IXNAME_SKY2* table stores sequential molecule keys for newly inserted or updated molecules.

Table:	IXNAME_SKY2	(REGNO NUMBER(10), TSTAMP TIMESTAMP(2), KEYS RAW (2000))
Index:	IXNAME_SKY21	REGNO ASC

where *IXNAME_SKY21* is the index on the table *IXNAME_SKY2*.

The *IXNAME_SRKY* table stores sequential reaction keys for newly inserted or updated reactions.

Table:	IXNAME_SRKY	(REGNO NUMBER(10), TSTAMP TIMESTAMP(2), KEYS RAW(2000))
Index:	IXNAME_SRKY1	REGNO ASC

where:

- *IXNAME_SRKY1* is the index on the table *IXNAME_SRKY*.
- *IXNAME_SRKY* is the index that stores sequential reaction keys for newly inserted or updated reactions.

The *IXNAME_IKY2* table stores inverted keys for molecules with inverted or rebuilt keys.

Table:	IXNAME_IKY2	(CHUNKNO NUMBER(n), KEYS BLOB)
--------	-------------	--------------------------------

The *IXNAME_IRKY* table stores inverted keys for reactions with inverted or rebuilt keys.

Table:	IXNAME_IRKY	(CHUNKNO NUMBER(9), RXNKEYS BLOB)
--------	-------------	-----------------------------------

The *IXNAME_FLEX* table stores FLEXMATCH fragment hashes for each component of each molecule.

Table:	IXNAME_FLEX	(REGNO NUMBER(10), MOLHASH NUMBER(10))
Index:	IXNAME_FLEX1	MOLHASH ASC
Index:	IXNAME_FLEX2	REGNO ASC

where:

- IXNAME_FLEX is the table that stores FLEXMATCH fragment hashes for each component of each molecule.
- IXNAME_FLEXnumber are indexes on the table IXNAME_FLEX

The IXNAME_RFLX table stores FLEXMATCH fragment hashes for each component of each reaction.

Table:	IXNAME_RFLX	(REGNO NUMBER(10), MOLTYPE NUMBER(2), MOLNUM NUMBER(3), MOLHASH NUMBER(10))
Index:	IXNAME_RFLX1	MOLHASH ASC
Index:	IXNAME_RFLX2	REGNO ASC

where:

- IXNAME_RFLX is the table that stores FLEXMATCH fragment hashes for each component of each reaction
- IXNAME_RFLXnumber are indexes on the table IXNAME_RFLX

The IXNAME_SGRP table stores Sgroup data for any molecules or reactions that contain data Sgroups. This table is used to speed up FLEXMATCH or RXNFLEXMATCH searches.

Table:	IXNAME_SGRP	(FIELDNAME VARCHAR2(30), REGNO NUMBER(10), MINVAL NUMBER, MAXVAL NUMBER, TEXTVAL VARCHAR2(4000))
Index:	IXNAME_SGRP1	FIELDNAME ASC
Index:	IXNAME_SGRP2	REGNO ASC

where:

- IXNAME_SGRP is a table that stores Sgroup data for use as a prescreen during FLEXMATCH or RXNFLEXMATCH searches
- IXNAME_SGRPnumber are indexes on the table IXNAME_SGRP

The IXNAME_FSUP table stores Fastsearch information for newly inserted or updated molecules or reactions.

Table:	IXNAME_FSUP	(REGNO NUMBER(10), SEQNO NUMBER(9), TSTAMP TIMESTAMP(2), FSUPDATE BLOB)
Index:	IXNAME_FSUP1	REGNO ASC

The IXNAME_FSIX table stores Fastsearch information.

Table:	IXNAME_FSIX	(SEGNO NUMBER(9), FSINDEX BLOB)
--------	-------------	---------------------------------

The IXNAME_CCLK table is used when registration duplicate checking is enabled.

Table:	IXNAME_CCLK	(CLASSCODE NUMBER(10))
Index:	IXNAME_CCLK1	CLASSCODE ASC

The IXNAME_CTAB table stores a copy of the molecules or reactions if registration duplicate checking is enabled.

Table:	IXNAME_CTAB	(REGNO NUMBER(10), CTAB BLOB)
Index:	IXNAME_CTAB1	REGNO ASC

The IXNAME_NEC table stores NEMA keys used for molecule exact match searching.

Table:	IXNAME_NEC	(REGNO NUMBER(10), CONHASH VARCHAR2(30), STEHASH VARCHAR2(30), FLAGS NUMBER(4))
Index:	IXNAME_NEC1	REGNO ASC
Index:	IXNAME_NEC2	CONHASH ASC
Index:	IXNAME_NEC3	STEHASH ASC

The IXNAME_GENX table stores prescreen data used to speed up the generic-exact match (FLEXMATCH (ctab,query, 'GENERIC')) search.

Table:	IXNAME_GENX	(REGNO NUMBER(10), ISGENRIC NUMBER(1), NUMSPECIFICS NUMBER(10), MINMOLWT NUMBER, MAXMOLWT NUMBER)
Index:	IXNAME_GENX1	REGNO ASC
Index:	IXNAME_GENX2	NUMSPECIFICS ASC, MINMOLWT ASC, MAXMOLWT

where:

- IXNAME_GENX is a table that stores data for use as a prescreen during GENEXACT searches.
- IXNAME_GENXnumber are indexes on the table IXNAME_GENX.

The IXNAME_BSQ table stores biopolymer sequence text used for substructure searching when the query does not contain any modified or cross-linked monomers.

Table:	IXNAME_BSQ	(REGNO NUMBER(10), SEQUENCE CLOB)
Index:	IXNAME_BSQ1	REGNO ASC

Permissions Required for BIOVIA Direct Users

During the installation of Direct, the installer creates the user who will own Direct, and specific functionality within Direct.

You add other users as explained in Oracle Database Server documentation.

Grant these privileges directly to each Oracle Database Server user account that accesses Direct:

- CREATE SYNONYM
- CREATE SEQUENCE

Grant this privilege if users will be creating or rebuilding indexes using PL/SQL.

Do not grant these privileges to *roles* that you assign to users. Grant them directly to users.

```
SQL> sqlplus
      connect system/password
      grant create synonym to user1;
      grant create synonym to user2;
      grant create synonym to user3;
      grant create sequence to user2;
```

The preceding example assumes that only user2 plans to use PL/SQL.

Inform users that before they can use Direct, they must execute the procedure C\$DIRECT2021.MDLAUXOP.SETUP. The MDLAUXOP.SETUP procedure sets up the synonyms that they need to use Direct.

Dropping and Recreating Domain Indexes

If a molecule or reaction domain index becomes corrupted to the point where ALTER INDEX REBUILD cannot correct it, you may need to drop the index and recreate it. This section describes how to recreate an index so that it uses the same parameters as the original index.

Before dropping and recreating the index, try to rebuild it using ALTER INDEX REBUILD. If this succeeds then you do not need to drop and recreate the index.

If ALTER INDEX REBUILD cannot start because locks cannot be obtained, try shutting down and then restarting Oracle. That will remove any transient locks, and ALTER INDEX REBUILD may then be able to proceed.

If ALTER INDEX REBUILD starts processing but then fails, for example with error ORA-28579, then recreating the index may also fail with the same error. Try rebuilding the index without fastsearch and see if that corrects the problem, for example:

```
ALTER INDEX SAMPLE2D_IX REBUILD PARAMETERS ('DISABLE=FASTSEARCH');
```

Contact BIOVIA Customer Support for assistance in this case.

If recreating the index is still necessary, determine what parameters were used when the index was created. Use MDLAUX.INDEXPARAMETERS to print out the creation parameters, there will be one per line following an initial line naming the index and telling whether it is a molecule or reaction index. For example:

```
SQL> select mdlaux.indexparameters('samplegen') from dual;

MDLAUX.INDEXPARAMETERS('SAMPLEGEN')
-----
-----
Molecule index SAMPLEGEN_IX version 9.1.0.36 (created with 9.1.0.36)
```

```
GENERIC  
PREALLOCATE=NOSTORAGECHECK
```

In this example you would recreate the index using:

```
CREATE INDEX SAMPLEGEN_IX ON SAMPLEGEN (CTAB)  
INDEXTYPE IS C$DIRECT2021.MXIXMDL  
PARAMETERS ('GENERIC PREALLOCATE=NOSTORAGECHECK');
```

If any of the following parameters are included in the output from MDLAUX.INDEXPARAMETERS, contact BIOVIA Customer Support for assistance.

The following parameters are displayed, but are defined globally and do not need to be defined using CREATE INDEX:

PTABLE, SALTS, MOL2DKEYDEFS, RXNCTRKEYDEFS, SGROUPFIELDS, MOLSKEYDEFS, and ISOTOPE DATA

If MDLAUX.INDEXPARAMETERS prints an Oracle error and does not provide any parameter output, it may still be possible to determine the index creation parameters. Contact BIOVIA Customer Support for assistance.

Storing Structures that Cannot be Indexed or Searched

During molecule or reaction registration, Direct's MOL or RXN operators raise an exception when the input is a structure which cannot be indexed and searched in Direct.

For example, a molecule containing substructure query features such as query bond types cannot be indexed or searched, an attempt to insert such a structure into a Direct molecule column will generate an error:

```
SQL> create table moltest (id varchar2(20), ctab blob);
```

Table created.

```
SQL> insert into moltest values ('QueryMol', mol('c:\querymol.mol'));  
insert into moltest values ('QueryMol', mol('c:\querymol.mol'))
```

*

ERROR at line 1:

ORA-20100: MDL-1919: Molecule failed registration check:

Error: (root) No query features allowed for registration

MDL-0633: Unable to convert molfile string to binary molecule ctab

ORA-06512: at "C\$DIRECT2021.MDLAUXOP", line 326

ORA-06512: at "C\$DIRECT2021.MDLAUXOP", line 319

ORA-06512: at "C\$DIRECT2021.MDLAUXOP", line 302

When chemical structures coming from diverse sources are registered, some structures might contain features that prevent their registration in the Direct structure column. Although the structures cannot be indexed or searched, you might want to add the records to the table. Using a trigger to intercept errors from the MOL or RXN operator will allow you to store such structures in a separate column and a NULL can be stored in the BLOB structure column. This allows a record and any associated data to be inserted in the table even though the chemical structure cannot be searched.

Here is an example of a trigger on a CLOB column which will automatically populate the BLOB molecule column when a molfile is inserted into the CLOB column. If the molecule cannot be converted to binary form by the MOL operator the molfile is still inserted into the CLOB column, however a NULL is inserted into the BLOB molecule column and the error output from the MOL operator is stored in another column named CONVERSION_ERRORS.

This example works for reactions as well, replace the MOL operator with RXN to use with a reaction database.

```
create table test_moltable (
  id varchar2(10) primary key,
  input_molfile clob,
  ctab blob,
  conversion_errors clob);

create trigger test_moltable_mol_trigger
before insert or update on test_moltable
for each row
declare
begin
  if ((inserting or updating('INPUT_MOLFILE')) and
      :new.input_molfile is not null) then
    begin
      :new.ctab := mdlaux.mol(:new.input_molfile);
      :new.conversion_errors := null;
    exception
      when others then
        :new.ctab := null;
        :new.conversion_errors := mdlaux.errors;
    end;
  end if;
end;
/
show errors;

-- phcl.mol is a molfile containing chlorobenzene
insert into test_moltable (id, input_molfile)
values ('Test1', readfile('c:\phcl.mol'));
select mdlaux.errors from dual;

-- l01.mol is a substructure query containing a link node which cannot be
registered
insert into test_moltable (id, input_molfile)
values ('Test2', readfile('c:\l01.mol'));
select mdlaux.errors from dual;

commit;

column errors format a40
select id, molwt(ctab), substr(conversion_errors,1,40) "ERRORS"
from test_moltable
order by id;
```

ID	MOLWT(CTAB)	ERRORS
Test1	112.5569	
Test2		MDL-1919: Molecule failed registration c

```
-- Locate records which are not indexed or searchable,
-- input molfile is available for all records
```

```
select id, dbms_lob.getlength(input_molfile)
  from test_moltable
 where ctab is null
 order by id;
```

ID	DBMS_LOB.GETLENGTH(INPUT_MOLFILE)
Test2	607

Chapter 4:

Maintaining Tables and Indexes

This chapter explains how to maintain your molecule or reaction table and indexes. It also explains how to maintain keys and generate and maintain indexes.

Direct does not prevent you from dropping a table while another user is performing maintenance on an index of that table.

Routine Index Maintenance

This section lists procedures you perform as part of maintaining molecule or reaction indexes. It also includes information on optional procedures that users might need to perform on their molecule or reaction indexes.

Note: BIOVIA strongly recommends that you commit records to the database prior to performing index maintenance such as `EXECUTE MDLAUX.UPDATEPENDINGVERSIONS ('mytable_ix');` MDL and Oracle errors occur if you try to perform maintenance before committing records. For example:

```
BEGIN
MDLAUX.UPDATEPENDINGVERSIONS('mytable_ix');
END;
ERROR at line 1:
```

```
ORA-20100: MDL-0346: Unable to lock table, SQL=LOCK TABLE
"DIRECTQA"."MYTABLE_IX_SKY2" IN EXCLUSIVE MODE: ORA-0060:
deadlock detected while waiting for resource
MDL-0617: ERROR inverting pending 2D molecule keys
ORA-06512: at "C$DIRECT2021.MDLAUXOP", line 2460
ORA-06512: at "C$DIRECT2021.MDLAUXOP", line 2451
ORA-06512: at "C$DIRECT2021.MDLAUXOP", line 2435
ORA-06512: at line 1
ERRORS
```

```
-----
MDL-0346: Unable to lock table, SQL=LOCK
TABLE"DIRECTQA"."MYTABLE_IX_SKY2" IN EXCLUSIVE MODE: ORA-00060: deadlock
detected while waiting for resource
MDL-0617: ERROR inverting pending 2D molecule keys
```

For more information about the functions and procedures that you can use to perform routine index maintenance, see [Cartridge Management Functions and Procedures](#).

Scan for Duplicate Molecules

To discover which molecule structures have duplicates in the database, use the FLEXMATCH search operator with each molecule (CTAB) as the query. Supply the appropriate database duplicate definition as the FLEXMATCH options. To search a table containing generic molecule structures, use the keyword 'GENERIC' instead of 'MATCH=ALL'.

You might need to use a duplicate definition that differs from MATCH=ALL. For example, you might want tautomers of a structure to be perceived as duplicates. For more information, see *Registration to Chemical Databases* in the *BIOVIA Chemical Representation Guide*.

Use the following SELECT statement, including the hints, to scan for duplicate molecules from a table, for example:

```
/*
* Scan a table for duplicate molecules, store duplicates in an output table.
* Inputs:
*   Name of molecule table
*   Name of column containing molecules (usually CTAB)
*   Name of primary key or other ID column
*   Flexmatch switches to use with search, for example 'match=all'
*   Name of output table, it will be created
* Output:
*   Rows are written to the output table when duplicates are discovered.
*   Columns of the output table are:
*       RECORD      Value of ID column for record in table
*       DUPLICATE   Value of ID column for a duplicate of RECORD
*       STATUS      'TIMEOUT' if the exact-match search timed out
*                   'ERROR'   if an error occurs, in this case there is no
*                   value for DUPLICATE
*
* STATUS will be returned as 'ERROR' if the record contains only salt
* fragments and the flexmatch switches do not include the 'SAL' switch.
*/
create or replace procedure scandupmols(
mol_table in varchar2,
ctab_column in varchar2,
id_column in varchar2,
flexmatch_switches in varchar2,
output_table in varchar2)
is
type curtyp is ref cursor;
csr curtyp;
id varchar2(4000);
ctab blob;
dupid varchar2(4000);
dupcsr curtyp;
timeout number;
status varchar2(10);
begin
-- Create the output table
execute immediate
'CREATE TABLE '||output_table||
' (RECORD VARCHAR2(4000), DUPLICATE VARCHAR2(4000), STATUS VARCHAR2(10))';
open csr for
'SELECT '||id_column||','||ctab_column||' FROM '||mol_table;
loop
fetch csr into id, ctab;
exit when csr%notfound;
-- Skip NULL or nostruct molecules, these don't have duplicates
-- (This also skips molecules containing only hydrogen atoms)
```

```

if (ctab is null) then continue; end if;
if (mdlaux.isnostruct(ctab) = 1) then continue; end if;
begin
open dupcsr for
'SELECT '||id_column||',FLEXMATCHTIMEOUT(1) FROM '||mol_table||
' WHERE FLEXMATCH('||ctab_column||',:qry, ''||
flexmatch_switches||'',1)=1'
using ctab;
loop
fetch dupcsr into dupid, timeout;
exit when dupcsr%notfound;
if (dupid <> id) then
if (timeout = 1) then
status := 'TIMEOUT';
else
status := null;
end if;
execute immediate
'INSERT INTO '||output_table||' VALUES (:id,:dupid,:sts)'
using id, dupid, status;
execute immediate 'COMMIT';
end if;
end loop;
close dupcsr;
exception
when others then
execute immediate
'INSERT INTO '||output_table||' VALUES (:id,NULL,'ERROR')'
using id;
execute immediate 'COMMIT';
end;
end loop;
close csr;
end;
/
show errors

```

A more comprehensive example script that searches a database for duplicate molecules is located in the *Direct examples directory*, (<install_directory>/direct2021/examples, where <install_directory> is typically either:

c:\BIOVIA

or

/opt/BIOVIA

The script is named `scan_duplicate_molecules.sql`. This script will ignore records which are not valid exact-match queries, and will store the duplicate information along with timeout and error status in a separate table. Comments at the top of the script describe its use.

Scan for Duplicate Reactions

Use the following SELECT statement, including the hints, to scan for duplicate reactions from a table, for example:

```
/*
* Scan a table for duplicate reactions, store duplicates in an output table.
* Input
:
*   Name of reaction table
*   Name of column containing reactions (usually RCTAB)
*   Name of primary key or other ID column
*   Flexmatch switches to use with search, for example 'match=all'
*   Name of output table, it will be created
* Output:
*   Rows are written to the output table when duplicates are discovered.
*   Columns of the output table are:
*       RECORD      Value of ID column for record in table
*       DUPLICATE   Value of ID column for a duplicate of RECORD
*       STATUS      'TIMEOUT' if the exact-match search timed out
*                   'ERROR'   if an error occurs, in this case there is no
*                   value for DUPLICATE
*
* STATUS will be returned as 'ERROR' if the record contains only salt
* fragments and the flexmatch switches do not include the 'SAL' switch.
*/
/*
* Helper function returns true if all components of a reaction are nostruct
* molecules. Such a reaction can not have duplicates.
*/
create or replace function isallnostructs(ctab in blob)
return boolean
is
ret boolean;
n number;
i number;
component clob;
begin
ret := true;
n := mdlaux.ncomponents(ctab, 1);
for i in 1..n loop
component := mdlaux.rxnmol(ctab, 1, i);
if (mdlaux.isnostruct(component) = 0) then ret := false; end if;
dbms_lob.freetemporary(component);
if (not ret) then exit; end if;
end loop;
if (ret) then
n := mdlaux.ncomponents(ctab, 2);
for i in 1..n loop
component := mdlaux.rxnmol(ctab, 2, i);
if (mdlaux.isnostruct(component) = 0) then ret := false; end if;
dbms_lob.freetemporary(component);
if (not ret) then exit; end if;
end loop;
end if;
return ret;
end;
```



```

/
show errors;
create or replace procedure scanduprxns(
rxn_table in varchar2,
ctab_column in varchar2,
id_column in varchar2,
flexmatch_switches in varchar2,
output_table in varchar2)
is
type curtyp is ref cursor;
csr curtyp;
id varchar2(4000);
ctab blob;
dupid varchar2(4000);
dupcsr curtyp;
timeout number;
status varchar2(10);
begin
-- Create the output table
execute immediate
'CREATE TABLE '||output_table||
' (RECORD VARCHAR2(4000), DUPLICATE VARCHAR2(4000), STATUS VARCHAR2(10))';
open csr for
'SELECT '||id_column||','||ctab_column||' FROM '||rxn_table;
loop
fetch csr into id, ctab;
exit when csr%notfound;
-- Skip NULL or reactions consisting entirely of nostruct molecules,
-- these do not have duplicates
if (ctab is null) then continue; end if;
if (mdlaux.hasnostructs(ctab) = 1) then
if (isallnostructs(ctab)) then continue; end if;
end if;
begin
open dupcsr for
'SELECT '||id_column||',RXNFLEXMATCHTIMEOUT(1) FROM '||rxn_table||
' WHERE RXNFLEXMATCH('||ctab_column||',:qry,'''||
flexmatch_switches||'',1)=1'
using ctab;
loop
fetch dupcsr into dupid, timeout;
exit when dupcsr%notfound;
if (dupid <> id) then
if (timeout = 1) then
status := 'TIMEOUT';
else
status := null;
end if;
execute immediate 'INSERT INTO '||output_table||' VALUES (:id,:dupid,:sts)'
using id, dupid, status;
execute immediate 'COMMIT';
end if;

```

```
end loop;
close dupcsr;
exception
when others then
execute immediate
'INSERT INTO '||output_table||' VALUES (:id,NULL,'ERROR')'using id;
execute immediate 'COMMIT';
end;
end loop;
close csr;
end;
/
show errors
```

A more comprehensive example script which will search a database for duplicate reactions is located in the Direct examples directory. See `scan_duplicate_reactions.sql`. This script will ignore records which are not valid exact-match queries, and will store the duplicate information along with timeout and error status in a separate table. Comments at the top of the script describe its use.

Scan the Substructure Search Keys

To validate the substructure search and similarity keys:

1. Start SQL*Plus and log in.
2. Call the SQL procedure `MDLAUX.SCANINDEX`.

Syntax: `EXECUTE MDLAUX.SCANINDEX('INDEX','KEYS');`

For example:

```
SQL> EXECUTE MDLAUX.SCANINDEX('SAMPLE2D_IX','KEYS');
```

Output from the scan is written into the local index log table after the table has been truncated. See [MDLAUX.LOGTABLE](#) for information on selecting rows from this table. Index errors are written to this table, as are progress messages every *n* rows, where *n* is a variable that is set to 100 or 1000, depending on the size of the table.

If there are no errors, an example of typical output in the log table is:

```
SELECT SUBSTR(msg,1,60) MSG FROM TABLE(mdlaux.logtable('sample2d_
mol','asc'));
MSG
-----
Scanning molecule index SAMPLE2D_IX on table DAVIDG.SAMPLE2D_MOL
Scanning ROWID conversion table
ROWID conversion table scan complete, there are 365 rows
Scanning molecule 2D KEYS index
Scanning row 100
Scanning row 200
Scanning row 300
Molecule 2D KEYS scan complete
```

For more information about the syntax and usage of this procedure, see [MDLAUX.SCANINDEX](#).

Error lines include the ROWID, if possible, and could also indicate whether the error occurred for reactant or product.

Update the Inverted Key Index

To update the molecule or reaction substructure search and similarity keys and similarity fingerprints, either issue the SQL command `ALTER INDEX ... REBUILD PARAMETERS ('UPDATEPENDINGINVERSIONS')`, or execute the SQL procedure `MDLAUX.UPDATEPENDINGINVERSIONS`. These commands improve search speed by incorporating recent updates into the search index.

For example:

```
SQL> EXECUTE MDLAUX.UPDATEPENDINGINVERSIONS('CORPDB_MOL_IX');
```

Tip: You must be logged in to Oracle Database Server as the owner of the domain index to execute this procedure.

Note: BIOVIA does not recommend that you use `ALTER INDEX`. The `ALTER INDEX` command prevents users from concurrently performing substructure or similarity searches. You cannot restart if a failure occurs.

Updating the inverted key index incorporates any recent updates of those keys into the search index, thereby improving search speed. BIOVIA recommends updating the inversion key index after every 2000 insertions and/or updates.

Update a Fastsearch Index

To update the molecule or reaction Fastsearch index, either issue the SQL command `ALTER INDEX ... REBUILD PARAMETERS ('UPDATEPENDINGFASTSEARCH')`, or execute the SQL procedure `MDLAUX.UPDATEPENDINGFASTSEARCH`, as shown in the example that follows.

```
SQL> EXECUTE MDLAUX.UPDATEPENDINGFASTSEARCH('CORPDB_MOL_IX');
```

Tip: You must be logged in to Oracle Database Server as the owner of the domain index to execute this procedure.

Note: BIOVIA does not recommend that you use `ALTER INDEX`. The `ALTER INDEX` command prevents users from concurrently performing substructure searches. You cannot restart if a failure occurs.

Updating the molecule Fastsearch index improves search speed by incorporating recent updates into the search index. BIOVIA recommends updating the inversion key index after every 2000 insertions and/or updates.

Validate the Fastsearch Index

To validate the Fastsearch index:

1. Start SQL*Plus and log in.
2. Call the SQL procedure `MDLAUX.SCANINDEX`.

Syntax: `EXECUTE MDLAUX.SCANINDEX ('INDEX', 'FASTSEARCH')`

For example:

```
SQL> EXECUTE MDLAUX.SCANINDEX('SAMPLE2D_IX', 'FASTSEARCH');
```

Output from the scan is written into the local index log table after the table has been truncated. For information on selecting rows from this table, see [MDLAUX.LOGTABLE](#). Index errors are written to this

table, as are progress messages every n rows, where n is a variable that is set to 100 or 1000, depending on the size of the table. If there are no errors, an example of typical output in the log table is:

```
SQL> SELECT SUBSTR(msg,1,60) MSG FROM TABLE(MDLAUX.LOGTABLE('sample2d_
mol','asc'));
MSG
```

```
-----
Scanning molecule index SAMPLE2D_IX on table DAVIDG.SAMPLE2D_MOL
Scanning ROWID conversion table
ROWID conversion table scan complete, there are 365 rows
Validating FASTSEARCH internal structure
Scanning node 1000
Scanning node 2000
Scanning node 3000
Scanning node 4000
Scanning node 5000
Scanning node 6000
Scanning node 7000
Scanning node 8000
Scanning node 9000
Scanning node 10000
Scanning node 11000
Scanning node 12000
Scanning node 13000
Scanning node 14000
Scanning node 15000
Scanning node 16000
Scanning node 17000
Scanning node 18000
Scanning node 19000
Scanning node 20000
Scanning node 21000
```

For more information about the syntax and usage of this procedure, see [MDLAUX.SCANINDEX](#).

Error lines include the ROWID, if possible, and could also indicate whether the error occurred for reactant or product.

Rebuild a Domain Index

Rebuild an index when the index becomes corrupted. To rebuild an index, you do not need to specify parameters unless you intend to change the existing parameters or parameter values. You can only rebuild an existing index.

To rebuild a domain index, issue the SQL statement and specify the parameters, as shown in this example:

```
SQL> ALTER INDEX INDEX REBUILD [PARAMETERS ('PARAMETERS')];
```

where INDEX is the name of the domain index you want to rebuild. For information on the parameters that you can specify with ALTER INDEX, see [Parameters for ALTER INDEX REBUILD](#).

Oracle Database Server calls Direct to alter a domain index. Direct alters the underlying objects as specified by the REBUILD and PARAMETERS options.

A Direct domain index can be completely or partially rebuilt. For example, to rebuild only the index information needed for FLEXMATCH searching, specify a parameter of FLEXMATCH.

If you are attempting to create a domain index on a table, see [Create a Domain Index](#). For information on how to recreate an index which was removed by uninstalling Direct, see [Recreate Invalid Domain Indexes](#).

Note: When you rebuild a domain index, you do not need to specify parameters. Direct uses the parameters and parameter values that were specified when you created the index initially. However, to change the parameters, or change their values, use the optional keyword `PARAMETERS` and specify the parameters and values that you want to change.

If you do not specify any parameters to `ALTER INDEX REBUILD` it rebuilds the entire index. The series of steps when it rebuilds the entire index are very similar to those used during index creation, see [Domain Index Creation](#). An Oracle I/O error will leave the index in an invalid state. It will need to be rebuilt again, or dropped and recreated. An error processing chemistry will, in most cases, result in the record not being indexed. The record should be corrected and updated as soon as possible.

Rebuild the Fastsearch Index

To rebuild the Fastsearch index issue either the SQL command `ALTER INDEX ... REBUILD PARAMETERS ('FASTSEARCH')`, or execute the SQL procedure `MDLAUX.RECREATEFASTSEARCH`.

For example:

```
SQL> EXECUTE MDLAUX.RECREATEFASTSEARCH('CORPDB_MOL_IX');
```

Note: BIOVIA does not recommend that you use `ALTER INDEX`. The `ALTER INDEX` command prevents users from concurrently performing substructure searches. You cannot restart the search if a failure occurs.

Rebuild the FLEXMATCH or RXNFLEXMATCH Index

To rebuild the molecule FLEXMATCH search index or the RXNFLEXMATCH index, issue the SQL command `ALTER INDEX ... REBUILD PARAMETERS ('FLEXMATCH')`, for example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('FLEXMATCH');
```

Rebuild the Molecular Formula Index

To rebuild the molecular formula search index, issue the SQL command `ALTER INDEX ... REBUILD PARAMETERS ('FORMULA')`, for example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('FORMULA');
```

Note: Rebuilding the molecular formula index is not the same as rebuilding the formula string. For more information, see [Recompute the Molecular Formula](#).

Rebuild the Substructure Search Keys

To rebuild the molecule or reaction substructure search and similarity keys issue either the SQL command `ALTER INDEX ... REBUILD PARAMETERS ('KEYS')`, or execute the SQL procedure `MDLAUX.RECREATEKEYS`.

For example:

```
SQL> EXECUTE MDLAUX.RECREATEKEYS('CORPDB_MOL_IX');
```

Note: BIOVIA does not recommend that you use `ALTER INDEX`. The `ALTER INDEX` command prevents users from concurrently performing substructure, or similarity searches. You cannot restart the search if a failure occurs.

Truncate a Domain Index

Oracle Database Server permits users to truncate tables whenever they want to remove data. The TRUNCATE operation is similar to dropping and recreating a table. However, it does not utilize redo logs or require a COMMIT statement. This reduces system overhead. For information about the TRUNCATE operation, see your Oracle Database Server documentation.

To permit Direct users to utilize the Oracle Database Server TRUNCATE functionality, Direct truncates any domain indexes that were created on the table that the user truncates.

To truncate a molecule table and associated domain index, issue the SQL statement that follows:

```
TRUNCATE TABLE TABLE;
```

where TABLE is the name of the table that you want to truncate.

Oracle Database Server calls Direct to truncate any domain indexes that are associated with the table. Direct truncates any underlying objects as well.

IMPORTANT! Direct might not prevent a user from truncating a table while another user is performing maintenance on an index on that table.

Analyze a Domain Index

The Oracle optimizer analyzes user tables and the data contained therein to compare the cost, in terms of system overhead, of alternative paths to access data. Direct supports this optimizer functionality by permitting users to analyze the domain indexes that they create. This data is reported to the Oracle optimizer, which uses it to optimize access to data. For more information on the Oracle optimizer, see your Oracle Database Server documentation.

To analyze a domain index, use the DBMS_STATS package functions on the index or table containing the domain index.

To compute table statistics:

```
SQL> EXECUTE DBMS_STATS.GATHER_TABLE_STATS('SCHEMA', 'MOLTABLE',
CASCADE=>TRUE, ESTIMATE_PERCENT=>NULL);
```

To estimate statistics, set a value for ESTIMATE_PERCENT from 0.000001 to 100, the percent of rows to look at. For example, if *n* is 5, then 5% of the table will be looked at:

```
SQL> EXECUTE DBMS_STATS.GATHER_TABLE_STATS('SCHEMA', 'MOLTABLE',
CASCADE=>TRUE, ESTIMATE_PERCENT=>n);
```

where:

- SCHEMA is replaced with the actual schema name. The case must match the actual value, which is generally upper case.
- MOLTABLE is replaced with the actual molecule table name. The case must match the actual table name, which is generally upper case.

Recreate Invalid Domain Indexes

This section explains how to use the function MDLAUX.RECREATEINDEXES. Use this function when you uninstall and reinstall Direct. This function recreates all invalid domain indexes in the current user schema.

Note: Rebuilding a domain index is not the same as recreating an invalid domain index. You recreate an invalid index only when you re-install Direct. Rebuild an index when the index becomes corrupted. For information on how to rebuild a domain index on a table, see [Rebuild a Domain Index](#).

Before you can reinstall Direct, you must drop the existing installation. When you drop the installation, Oracle Database Server marks all existing domain indexes `INVALID` with type `UNKNOWN`. After you have reinstalled Direct, run `MDLAUX.RECREATEINDEXES` to validate the domain indexes.

The function `MDLAUX.RECREATEINDEXES` searches for all domain indexes marked invalid and recreates them without regenerating any indexed data.

If there are no invalid domain indexes, the function returns `NULL`. Otherwise, it returns a string containing the index, table, and column names that have been processed.

Note: Each user who has created a table with a domain index must log in and run this function.

To recreate all domain indexes, issue the SQL statement that follows:

```
SQL> SELECT MDLAUX.RECREATEINDEXES FROM DUAL;
```

This function searches for invalid molecule indexes of type `UNKNOWN`. It then issues the following commands for each molecule index.

```
DROP INDEX INDEX FORCE;
CREATE INDEX INDEX ON TABLE (COLUMN_NAME) INDEXTYPE IS
CARTRIDGESCHEMA.MXIXMDL PARAMETERS ('NOACTION');
```

The example that follows indicates the type of output that you might see:

```
RECREATEINDEXES
```

```
-----
Created mol index:SAMPLE2D_IX Table:SAMPLE2D_MOL Column:CTAB
```

If an error occurs during the `CREATE INDEX` process, processing terminates and Direct appends the error string from `MDLAUX.ERRORS` to the result, as shown:

```
RECREATEINDEXES
```

```
-----
Created mol index:SAMPLE2D_IX Table:SAMPLE2D_MOL Column:CTAB
```

```
*** Creation failed:
```

```
MDL-0333: Domain index creation failed
```

Note: Oracle Database Server still might create an index object which is invalid. Drop this object using the SQL statement that follows.

```
SQL> DROP INDEX SAMPLE2D_IX;
```

You cannot use `MDLAUX.RECREATEDINDEXES` to upgrade a domain index from Direct 6, 7, or 8 to Direct. You must use the `directupgrade` utility to upgrade a previous version of a domain index.

Optional Index Management Tasks

Add Fastsearch to a Domain Index

To add substructure Fastsearch capability to a domain index which was created *without* Fastsearch capability, issue the SQL command `ALTER INDEX`, for example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('ENABLE=FASTSEARCH');
```

Note: Domain indexes are created with Fastsearch enabled by default. Use this command to add Fastsearch only if the domain index was explicitly created without Fastsearch.

Add Substructure Search Keys to a Domain Index

To add substructure search keys to a domain index which was created without keys, issue the SQL command `ALTER INDEX`, for example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('ENABLE=KEYS');
```

Note: Domain indexes are created with keys enabled by default. Issue this command *only* if the domain index was explicitly created without keys.

Add a Modification Date Column to a Table

To add a modification date column to a table, issue the SQL command `ALTER TABLE` to add a `DATE` column, and then add or modify `INSERT` and `UPDATE` triggers on the table to incorporate a PL/SQL statement to set this field. There can be only one trigger at each trigger point on a table. Therefore, the source code for all triggers at the same trigger point must be merged into a single trigger.

The example that follows shows you how to add a modification date column to a table.

```
ALTER TABLE CORPDB_MOL ADD (MOL_MOD_DATE DATE);
CREATE OR REPLACE TRIGGER MOL_INS_TRIG
BEFORE INSERT ON CORPDB_MOL
FOR EACH ROW
BEGIN
    IF :NEW.MOL_MOD_DATE IS NULL THEN
        :NEW.MOL_MOD_DATE := SYSDATE;
    END IF;
END;
/
CREATE OR REPLACE TRIGGER MOL_UPD_TRIG
BEFORE UPDATE ON CORPDB_MOL
FOR EACH ROW
BEGIN
    IF (UPDATING('CTAB')) AND NOT UPDATING('MOL_MOD_DATE') THEN
        :NEW.MOL_MOD_DATE := SYSDATE;
    END IF;
END;
/
```

An alternative approach is to apply a column default value of `SYSDATE` to the date column, but this records only the date of insertion of the row, not the date of updates to the structure.

The `AFTER INSERT` trigger allows the user to supply the value and allows the user to copy rows from one table to another without setting this field to a new value.

The `UPDATE` trigger verifies whether the molecule structure has been updated and verifies whether the molecule date has been supplied in the update. This ensures that the trigger only sets the date when the molecule structure has actually been updated, and allows the date field to be supplied in an update.

Add a molfile or Chime String Column to a Table

To add a column to a table containing a molecule column, issue the SQL command `ALTER TABLE` to add a `CLOB` column, and then add or modify the `INSERT` and `UPDATE` triggers on the table to incorporate a PL/SQL statement to set this field. There can be only one trigger at each trigger point on a table. Therefore, the source code for all triggers at the same trigger point must be merged into a single trigger. Use the function `MDLAUX.MOLFILE` to generate a molfile from a stored molecule, and `MDLAUX.CLOBTOCHIME` to generate a Chime string from a molfile.

To add a molfile column, use a higher level application to manage this, by adding a column and creating a trigger, as shown in the example that follows.

```
ALTER TABLE CORPDB_MOL ADD (MOLFILE CLOB);
CREATE OR REPLACE TRIGGER MOL_INS_TRIG
BEFORE INSERT ON CORPDB_MOL
FOR EACH ROW
BEGIN
    IF :NEW.MOLFILE IS NULL THEN
        :NEW.MOLFILE := MDLAUX.MOLFILE(:NEW.CTAB);
    END IF;
END;
/
CREATE OR REPLACE TRIGGER MOL_UPD_TRIG
BEFORE UPDATE ON CORPDB_MOL
FOR EACH ROW
BEGIN
    IF (UPDATING('CTAB')) AND NOT UPDATING('MOLFILE') THEN
        :NEW.MOLFILE:= MDLAUX.MOLFILE(:NEW.CTAB);
    END IF;
END;
/
```

The step for creating a CHIME_STRING column is to wrap MDLAUX.MOLFILE with MDLAUX.CLOBTOCHIME to derive the Chime string if the value is not supplied in the insert or update:

```
:NEW.CHIME_STRING:= MDLAUX.CLOBTOCHIME(MDLAUX.MOLFILE(:NEW.CTAB));
```

Add a rxnfile or Chime String Column to a Table

To add a column to retain the rxnfile or chime representation of a reaction, use the SQL command ALTER TABLE to add a CLOB column to the table. Then add or modify the INSERT and UPDATE triggers on the table to incorporate a PL/SQL statement to set this field. There can be only one trigger at each trigger point on a table. Therefore, the source code for all triggers at the same trigger point must be merged into a single trigger. Use the function MDLAUX.RXNFILE to generate a rxnfile from a stored reaction, and MDLAUX.CLOBTOCHIME to generate a Chime string from a rxnfile.

For example:

```
ALTER TABLE CORPDB_RXN ADD (RXNFILE CLOB);

CREATE OR REPLACE TRIGGER RXN_INS_TRIG
BEFORE INSERT ON CORPDB_RXN
FOR EACH ROW
BEGIN
    IF :NEW.RXNFILE IS NULL THEN
        :NEW.RXNFILE := MDLAUX.RXNFILE(:NEW.RCTAB);
    END IF;
END;
/
CREATE OR REPLACE TRIGGER RXN_UPD_TRIG
BEFORE UPDATE ON CORPDB_RXN
FOR EACH ROW
BEGIN
    IF (UPDATING('RCTAB')) AND NOT UPDATING('RXNFILE') THEN
        :NEW.RXNFILE:= MDLAUX.RXNFILE(:NEW.RCTAB);
    END IF;
END;
```

```
END IF;  
END;  
/
```

The INSERT trigger verifies whether the RXNFILE field is null so as to allow the RXNFILE value to be supplied in the insert and to allow rows to be copied from one table to another without setting this field to a new value.

The UPDATE trigger verifies whether the reaction structure has been updated and verifies whether the RXNFILE field has been supplied in the update. This ensures that it only sets the RXNFILE field when the molecule structure has actually been updated, and allows the RXNFILE field to be supplied in an update.

The steps for creating a CHIME_STRING column are similar. Wrap MDLAUX.RXNFILE with MDLAUX.CLOBTOCHIME to derive the Chime string if the value is not supplied in the insert or update:

```
:NEW.CHIME_STRING:= MDLAUX.CLOBTOCHIME(MDLAUX.RXNFILE(:NEW.RCTAB));
```

Add a Molecular Formula Column to a Table

To add a molecular formula column to a table containing a molecule column, issue the SQL command ALTER TABLE to add a CLOB column. Then add or modify the INSERT and UPDATE triggers on the table to incorporate a PL/SQL statement to set this field. There can be only one trigger at each trigger point on a table. Therefore, the source code for all triggers at the same trigger point must be merged into a single trigger. Call the function MDLAUX.MOLFMLA to generate a molecular formula from a stored structure, for example:

```
ALTER TABLE CORPDB_MOL ADD (FORMULA CLOB);  
CREATE OR REPLACE TRIGGER MOL_INS_TRIG  
BEFORE INSERT ON CORPDB_MOL  
FOR EACH ROW  
BEGIN  
  
    IF :NEW.FORMULA IS NULL THEN  
        :NEW.FORMULA := MDLAUX.MOLFMLA('CORPDB_MOL_IX', :NEW.CTAB);  
    END IF;  
END;  
/  
CREATE OR REPLACE TRIGGER MOL_UPD_TRIG  
BEFORE UPDATE ON CORPDB_MOL  
FOR EACH ROW  
BEGIN  
    IF (UPDATING('CTAB')) AND NOT UPDATING('FORMULA') THEN  
        :NEW.FORMULA := MDLAUX.MOLFMLA('CORPDB_MOL_IX', :NEW.CTAB);  
    END IF;  
END;  
/
```

The INSERT trigger verifies whether the FORMULA field is null to allow the user to supply the value and to allow rows to be copied from one table to another without setting this field to a new value.

The UPDATE trigger verifies whether the molecule structure has been updated and whether the FORMULA field has been supplied in the update. This ensures that the FORMULA is only updated when the molecule structure has actually been updated, and allows the user to supply the FORMULA field in an update.

Note: Provide the molecule domain index name as an argument to the MDLAUX.MOLFMLA function.

Add a Molecule Name Column to a Table

To add a molecule name column to a table containing a molecule column, issue the SQL command `ALTER TABLE` to add a `VARCHAR2` column, and then add or modify `INSERT` and `UPDATE` triggers on the table to incorporate an SQL statement to set this field. There can be only one trigger at each trigger point on a table. Therefore, the source code for all triggers at the same trigger point must be merged into a single trigger. The function `MDLAUX.MOLNAME` can be used to get the molecule name from a stored structure.

An example of how to add a molecule name column:

```
ALTER TABLE CORPDB_MOL ADD (MOLNAME VARCHAR2(80));
CREATE OR REPLACE TRIGGER MOL_INS_TRIG
BEFORE INSERT ON CORPDB_MOL
FOR EACH ROW
DECLARE
    MOLNAME VARCHAR2(4000);
BEGIN
    MOLNAME := MDLAUX.GETSAVEDMOLNAME;
    IF :NEW.MOLNAME IS NULL THEN
        :NEW.MOLNAME := MOLNAME;
    END IF;
END;
/
CREATE OR REPLACE TRIGGER MOL_UPD_TRIG
BEFORE UPDATE ON CORPDB_MOL
FOR EACH ROW
DECLARE
    MOLNAME VARCHAR2(4000);
BEGIN
    MOLNAME := MDLAUX.GETSAVEDMOLNAME;
    IF (UPDATING('CTAB')) AND NOT UPDATING('MOLNAME') THEN
        :NEW.MOL_MOLNAME := MOLNAME;
    END IF;
END;
/
```

For information about the function `MDLAUX.MOLNAME`, see the *BIOVIA Direct Developers Guide* and *BIOVIA Direct Reference*.

Alternatively, supply the molecule name value as part of the insert and do not attempt to obtain the molecule name from the molfile being registered.

The `INSERT` trigger verifies whether the molname is null to allow the user to supply the value and to allow rows to be copied from one table to another without setting this field to a new value.

The `UPDATE` trigger verifies whether the molecule structure has been updated and whether the molname has been supplied in the update. This ensures that the molname is set only when the molecule structure has actually been updated, and to allow the user to supply the molname field in an update.

Add a Molecular Weight Column to a Table

To add a molecular weight column to a table containing a molecule column, issue the SQL command `ALTER TABLE` to add a `NUMBER` column with sufficient precision and scale, and then add or modify `INSERT` and `UPDATE` triggers on the table to incorporate a PL/SQL statement to set this field. There can

be only one trigger at each trigger point on a table. Therefore, the source code for all triggers at the same trigger point must be merged into a single trigger. Use the function `MDLAUX.MOLWT` to generate a molecular weight from a stored structure.

Note: Provide the molecule domain index name as an argument to the `MDLAUX.MOLWT` function.

An example of how to add a molecular weight column to a table;

```
ALTER TABLE CORPDB_MOL ADD (MOL_MOD_DATE DATE);
CREATE OR REPLACE TRIGGER MOL_INS_TRIG
BEFORE INSERT ON CORPDB_MOL
FOR EACH ROW
BEGIN
    IF :NEW.MOL_MOD_DATE IS NULL THEN
        :NEW.MOL_MOD_DATE := SYSDATE;
    END IF;
END;
/
CREATE OR REPLACE TRIGGER MOL_UPD_TRIG
BEFORE UPDATE ON CORPDB_MOL
FOR EACH ROW
BEGIN
    IF (UPDATING('CTAB')) AND NOT UPDATING('MOL_MOD_DATE') THEN
        :NEW.MOL_MOD_DATE := SYSDATE;
    END IF;
END;
/
```

Alternatively, you can supply the molecular weight value as part of the insert and not attempt to compute the molecular weight from the CTAB.

The INSERT trigger verifies whether the molecular weight is null to allow the user to supply the value and to allow rows to be copied from one table to another without setting this field to a new value.

The UPDATE trigger verifies whether the molecule structure has been updated and verifies whether the molecular weight has been supplied in the update. This ensures that it only sets the molecular weight when the molecule structure has actually been updated, and allows the molecular weight field to be supplied in an update.

Note: A generic molecule column has both a minimum and a maximum molecular weight value. These can be stored in two columns. Use the `MDLAUX.MOLWTMIN` and `MDLAUX.MOLWTMAX` functions to generate the values.

Add a Molecule Text Keys Column to a Table

To add a molecule text keys column to a table, issue the SQL command `ALTER TABLE` to add a `VARCHAR` column with sufficient width, and then add or modify `INSERT` and `UPDATE` triggers on the table to incorporate a PL/SQL statement to set this field. There can be only one trigger at each trigger point on a table. Thus, the source code for all triggers at the same trigger point must be merged into a single trigger. Use the function `MDLAUX.MOLKEYS` to generate text 2D molecule keys from a stored molecule.

An example of how to add a molecule text keys column to a table follows.

Use a higher level application to manage this functionality, by adding a column and creating a trigger, as shown in the following example.

```

ALTER TABLE CORPDB_MOL ADD (MOL2DKEYS VARCHAR2(4000));

CREATE OR REPLACE TRIGGER MOL_INS_TRIG
BEFORE INSERT ON CORPDB_MOL
FOR EACH ROW
BEGIN
    IF :NEW.MOL2DKEYS IS NULL THEN
        :NEW.MOL2DKEYS := MDLAUX.MOLKEYS('CORPDB_MOL_MDLIX', :NEW.CTAB, ' DEC
        DELIM=, LEAD TRAIL');
    END IF;
END;
/
CREATE OR REPLACE TRIGGER MOL_UPD_TRIG
AFTER UPDATE ON CORPDB_MOL
FOR EACH ROW
BEGIN
    IF (UPDATING('CTAB')) AND NOT UPDATING('MOL2DKEYS') THEN
        :NEW.MOL2DKEYS := MDLAUX.MOLKEYS('CORPDB_MOL_IX', :NEW.CTAB, ' DEC
        DELIM=, LEAD TRAIL');
    END IF;
END;
/

```

Note: BIOVIA provides the MDLAUXOP.MOLKEYS function for use in the trigger. For more information about this function, see the *BIOVIA Direct Developers Guide* and *BIOVIA Direct Reference Guide*.

The INSERT trigger verifies whether the MOL2DKEYS field is null to allow rows to be copied from one table to another without setting this field to a new value.

The UPDATE trigger verifies whether the molecule structure has been updated and verifies whether the MOL2DKEYS field has been supplied in the update. This ensures that it only sets the MOL2DKEYS field when the molecule structure has actually been updated, and allows the MOL2DKEYS field to be supplied in an update.

Note: Provide the molecule domain index name as an argument to the MDLAUX.MOLKEYS function.

Add an InChI key or SMILES String Column to a Table

To add a column to a table that contains a molecule column, issue the SQL command ALTER TABLE to add a VARCHAR2 or CTAB column. Then add or modify the INSERT and UPDATE triggers on the table to incorporate a PL/SQL statement to set the field. There can be only one trigger at each trigger point on a table. Therefore, the source code for all triggers at the same trigger point must be merged into a single trigger.

- Use a VARCHAR2(27) column and the function MDLAUX.INCHIKEY to generate an InChI key from the stored molecule.
- Use a CLOB column and the function MDLAUX.SMILES to generate the SMILES string from the stored molecule.

For example, to add a column containing an InChI or SMILES string:

```

ALTER TABLE CORPDB_MOL ADD (SMILES CLOB);
CREATE OR REPLACE TRIGGER MOL_INS_TRIG
BEFORE INSERT ON SAMPLE2D_MOL
FOR EACH ROW

```

```
BEGIN
  IF :NEW.INCHI IS NULL THEN
    :NEW.SMILES := MDLAUX.INCHI(:NEW.CTAB);
  IF :NEW.SMILES IS NULL THEN
    :NEW.SMILES := MDLAUX.SMILES(:NEW.CTAB);
  END IF;
END;
/
SHOW ERRORS;
CREATE OR REPLACE TRIGGER MOL_UPD_TRIG
BEFORE UPDATE ON SAMPLE2D_MOL
FOR EACH ROW
BEGIN
  IF UPDATING('CTAB') THEN
    IF NOT UPDATING('INCHI') THEN
      :NEW.INCHI := MDLAUX.INCHI(:NEW.CTAB);
    IF NOT UPDATING('SMILES') THEN
      :NEW.SMILES := MDLAUX.SMILES(:NEW.CTAB)
    END IF;
  END IF;
END;
/
SHOW ERRORS
```

Alter Duplicate Checking Behavior

To disable molecule duplicate checking during registration, issue the SQL command `ALTER INDEX ... REBUILD PARAMETERS ('UNIQUE=NO')`, for example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('UNIQUE=NO');
```

To enable molecule duplicate checking during registration, or to change the existing duplicate check criteria, issue the SQL command `ALTER INDEX ... REBUILD PARAMETERS ('UNIQUE=options')`, for example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('UNIQUE="match=all"');
```

When enabling duplicate checking, or changing existing duplicate criteria, `ALTER INDEX` performs duplicate checks using the new criteria for the entire database. If any duplicates are found, `ALTER INDEX` terminates with a warning and the existing setting is not changed.

Note: Domain indexes are created with duplicate checking disabled by default.

Alter Inverted Keys Chunk Size

Key-based substructure search and similarity searches generally perform better with larger chunk sizes. As the database grows, it might be beneficial to increase the key chunk size. For more information on chunk size, see [Parameters for CREATE INDEX](#).

To change the key chunk size issue the SQL command `ALTER INDEX ... REBUILD PARAMETERS ('CHUNKSIZE=size')`, for example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REGUILD PARAMETERS ('CHUNKSIZE=128K');
```

Recompute the Molecular Formula

To reset the molecular formula string values, set the molecular formula column to MOLFMLA(CTAB). For example:

```
UPDATE CORPDB_MOLTABLE SET FORMULA = MOLFMLA(CTAB);
COMMIT;
```

For very large tables, you might need to break the UPDATE statement into segments, using a WHERE clause to restrict the range of the records. For example:

```
UPDATE CORPDB_MOLTABLE SET FORMULA = MOLFMLA(CTAB) WHERE CDBREGNO < 1000000;
COMMIT;
UPDATE CORPDB_MOLTABLE SET FORMULA = MOLFMLA(CTAB) WHERE CDBREGNO >=
1000000;
COMMIT;
```

Recompute the Molecular Weight Values

To recalculate molecular weight values, set the molecular weight column to MOLWT(CTAB).

For example:

Syntax: UPDATE TABLE SET MOLWEIGHTCOLUMN=MOLWT(MOL_COLUMN);

Example:

```
UPDATE CORPDB_MOLTABLE SET MOLWEIGHT = MOLWT(CTAB);
COMMIT;
```

For very large tables, you might need to break the UPDATE statement into segments, using a WHERE clause to restrict the range of the records. For example:

```
SQL> UPDATE CORPDB_MOLTABLE SET MOLWEIGHT = MOLWT(CTAB) WHERE CDBREGNO <
1000000;
SQL> COMMIT;
SQL> UPDATE CORPDB_MOLTABLE SET MOLWEIGHT = MOLWT(CTAB) WHERE CDBREGNO >=
1000000;
SQL> COMMIT;
```

Validate the Molecular Formula Index

To validate the molecular formula index:

1. Start SQL*Plus and log in.
2. Call the SQL procedure MDLAUX.SCANINDEX('FORMULA'), for example:

```
SQL> EXECUTE MDLAUX.SCANINDEX('FORMULA');
```

Output from the scan is written into the local index log table after the table has been truncated. For information on selecting rows from this table, see [MDLAUX.LOGTABLE](#). Index errors are written to this table, as are progress messages every *n* rows, where *n* is a variable that is set to 100 or 1000, depending on the size of the table.

If there are no errors, an example of typical output in the log table is:

```
SQL> SELECT SUBSTR(msg,1,60) MSG FROM TABLE(MDLAUX.LOGTABLE('sample2d_
mol','asc'));
MSG
```

```
-----
Scanning molecule index SAMPLE2D_IX on table DAVIDG.SAMPLE2D_MOL
```

```
Scanning ROWID conversion table
ROWID conversion table scan complete, there are 365 rows
Scanning FORMULA index
Scanning row 100
Scanning row 200
Scanning row 300
FORMULA index scan complete
```

For more information about the syntax and usage of this procedure, see [MDLAUX.SCANINDEX](#).

Validate Molecular Weights

To locate records where the molecular weight is different from the value that will be computed using the MOLWT operator, compare each molecular weight to the value of MOLWT(CTAB). For example:

```
SQL> SELECT CDBREGNO,MOLWEIGHT, (MOLWEIGHT-MOLWT(CTAB)) AS DIFF
      2 FROM CORPDB_MOL
      3 WHERE MOLWEIGHT <> MOLWT(CTAB);
```

Remove Fastsearch from a Domain Index

To remove Fastsearch from a domain index, issue the SQL command ALTER INDEX ... REBUILD PARAMETERS ('DISABLE=FASTSEARCH'), for example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('DISABLE=FASTSEARCH');
```

IMPORTANT! Removing Fastsearch could significantly slow down substructure searches.

Remove Substructure Search Keys from a Domain Index

To remove the substructure search and similarity keys from a molecule domain index, issue the SQL command ALTER INDEX ... REBUILD PARAMETERS ('DISABLE=KEYS'), for example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('DISABLE=KEYS');
```

IMPORTANT! Removing keys could significantly slow down substructure searches, and significantly slows down similarity searches.

Rename a Domain Index

To rename a domain index, issue the SQL statement that follows:

```
SQL> ALTER INDEX OLD_INDEX RENAME TO NEW_INDEX;
```

This statement renames the domain index and its underlying objects.

Drop a Domain Index

To drop a domain index, issue the SQL statement that follows:

```
SQL> DROP INDEX INDEX [FORCE];
```

where INDEX is the name of the domain index that you want to drop.

Direct drops the underlying objects as well.

Enable NEMA Keys for Exact-match Searching

To use NEMA keys for exact match searches, issue the SQL command ALTER INDEX, for example:

```
ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('ENABLE=NEMAKEY');
```


After being enabled, NEMA keys are used for exact-match FLEXMATCH searches and for duplicate checking, during registration. For information on NEMA key searches, see *NEMA Key Searching* in the *BIOVIA Chemical Representation Guide*.

NEMA key is not currently used with reactions, but is used internally for symmetry detection and RXNFLEXMATCH timeout resolution.

Due to the slower molecule registration time, NEMA key generation is disabled by default. For a new molecule domain index, you can enable NEMA keys in exact-match searching by adding the parameter `ENABLE=NEMAKEY` to your `CREATE INDEX` statement. For example:

```
SQL> CREATE INDEX MOLDB_MDLIX ON MOLDB (CTAB)
      2 INDEXTYPE IS C$DIRECT2021.MXIXMDL |
      3 PARAMETERS ('ENABLE=NEMAKEY');
```

To enable use of NEMA keys in exact-match searches in an existing molecule domain index, use `ALTER INDEX REBUILD` with the parameter `ENABLE=NEMAKEY`. For example:

```
SQL> ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('ENABLE=NEMAKEY');
```

For information on operators and functions that generate NEMA keys, see the *BIOVIA Direct Developers Guide*.

Disable NEMA Keys for Exact-match Searching

To disable NEMA keys for exact match searches, issue the SQL command `ALTER INDEX`, for example:

```
ALTER INDEX CORPDB_MOL_IX REBUILD PARAMETERS ('DISABLE=NEMAKEY');
```

After being disabled, normal FLEXMATCH is used for MATCH-ALL searches, including duplicate checking, during registration. For information on NEMA key searches, see *NEMA Key Searching* in the *BIOVIA Chemical Representation Guide*.

Running Routine Index Maintenance Procedures Automatically

This section explains how to run routine maintenance procedures automatically in the background. It includes examples of code that you can use to set up maintenance procedures to run automatically.

New and updated records are added to the Direct domain index sequentially. Thus, substructure and similarity searches over newly added or updated records are relatively slow. To reduce the amount of time required for such searches, run these two routine maintenance procedures:

- `MDLAUX.UPDATEPENDINGFASTSEARCH`
- `MDLAUX.UPDATEPENDINGINVERSIONS`

These maintenance procedures update the domain index so that the new records are integrated into the inverted keys and Fastsearch tree, greatly speeding up searches.

For information on how to run these procedures manually, see [MDLAUX.UPDATEPENDINGFASTSEARCH](#) and [MDLAUX.UPDATEPENDINGINVERSIONS](#).

To run these procedures automatically at predetermined intervals, you can:

- Create batch scripts which run SQL*Plus and call these maintenance procedures for tables which are undergoing registration.
Use the tools that are provided with your operating system to set the batch scripts to run at the desired intervals.
- Use the `DBMS_JOB` or `DBMS_SCHEDULER` package provided with your Oracle Database Server to automate the running of these two procedures.

Review the example PL/SQL procedure **Example code** that follows uses the DBMS_JOB package to perform the tasks necessary to update the domain index automatically.

The update procedures are designed to be run concurrently with searching and registration.

Note: If you choose to perform these maintenance procedures using batch scripts, call the functions MDLAUX.PENDINGINVERSIONS and MDLAUX.PENDINGFASTSEARCH to determine if there are any records to be updated before you call the maintenance procedures.

The example procedure, UPDATEMDLIX, processes records with sequential keys as well as records that have pending FASTSEARCH updates. Run it as an Oracle job. UPDATEMDLIX creates a table named INDEX_JOB_LOG and writes it to a log. Errors cause the job to pause and to be marked as *broken*.

Examine the log table periodically. Correct errors as required, then mark such *broken* jobs *not broken* to continue processing.

UPDATEMDLIX takes a single argument, the name of a domain index. Do not include the schema name. The schema that submits the job should own the index being updated.

Note: You must own the domain index on which you run the procedures MDLAUX.PENDINGINVERSIONS and MDLAUX.PENDINGFASTSEARCH.

Example

The example code that follows creates a procedure, UPDATEMDLIX, that calls the maintenance procedures automatically. If you copy and paste this code, verify that you isolate the comments, which are marked by double dashes (– –) from the actual code.

Submit the job with a statement similar to that shown in the example. Specify the name of the index to be processed as the argument to UPDATEMDLIX(). The example assumes that the index is named mdctest_molix. This example starts the job at 1 AM on the Sunday following today and repeats once a week, always at 1 AM on the following Sunday.

```
VARIABLE jobid NUMBER
BEGIN
    DBMS_JOB.SUBMIT(
        :jobid,
-- Job ID
        'UPDATEMDLIX(''mdctest_molix'');',
-- what
        NEXT_DAY(TRUNC(SYSDATE), 'SUNDAY')+1/24,
-- when
        'NEXT_DAY(TRUNC(SYSDATE), ''SUNDAY'')+1/24');
-- Interval

-- Here, you must insert a COMMIT statement, or the job will not run.

    COMMIT;
END;
/
PROMPT Submitted job:
PRINT jobid

-- See the documentation for DBMS_JOB that is supplied by Oracle
-- Corporation for complete information on the arguments to
```

```

-- DBMS_JOB.SUBMIT and job scheduling.

-- To remove a job that has been submitted, supply the job number to remove
-- as the argument, as shown:
BEGIN
DBMS_JOB.REMOVE(&1);
END;
/
-- The example that follows demonstrates how to clear a broken flag. The
-- third argument controls when the job restarts. As in the preceding
-- example, the restart is scheduled for 1AM next Sunday:
BEGIN
DBMS_JOB.BROKEN(
    &1,
-- Job ID
    FALSE,
-- BROKEN=FALSE
    NEXT_DAY(TRUNC(SYSDATE), 'SUNDAY')+1/24);
-- When
END;
/
create or replace procedure updatemdlix (ixname varchar2)
is
    pending number;
    jobid number;
    errmsg varchar2(4000);
    errors varchar2(4000);
    doing_key boolean;
    doing_fs boolean;
    cnt number;
    id number;
    type CurType is ref cursor;
    csr curType;
    dattim date;
    msg varchar2(4000);
begin
    doing_key := false;
    doing_fs := false;

-- To create INDEX_JOB_LOG, if necessary, or to remove previous entries for
-- this index if it already exists:

SELECT COUNT(*) INTO cnt FROM USER_TABLES
    WHERE TABLE_NAME = 'INDEX_JOB_LOG';
    if (cnt = 0) then
        execute immediate
'CREATE TABLE INDEX_JOB_LOG'||
' (INDEXNAME VARCHAR2(30), ID NUMBER,'||
' DATTIM DATE, MSG VARCHAR2(4000))';
    else
        execute immediate
            'DELETE FROM INDEX_JOB_LOG WHERE INDEXNAME = :ix'
            using ixname;
        commit;

```

```
        end if;
        id := 1;

-- To find the job number, assuming that the job submission was a call to
-- UPDATEMDLIX, and that there is just one such job running.
execute immediate
'SELECT JOB'||
' FROM USER_JOBS'||
' WHERE UPPER(WHAT) LIKE '%UPDATEMDLIX%' ||
' AND UPPER(WHAT) LIKE UPPER('' || '%'||ixname||'% ' ||'')'
into jobid;

-- The table INDEX_JOB_LOG receives status information and any errors from
-- the MDLAUX.UPDATEPENDINGINVERSIONS and MDLAUX.UPDATEPENDINGFASTSEARCH
-- procedures. This procedure also writes header and trailer entries to the
-- table INDEX_JOB_LOG.

-- The example that follows shows how to invert keys for records which are
-- pending inversion.

execute immediate
'SELECT MDLAUX.PENDINGINVERSIONS(:ix) INTO :cnt FROM DUAL'
into cnt
using ixname;

if (not (cnt > 0)) then
    msg := 'Background job '||to_char(jobid)||
        ' skipping key inversion,'||
        ' pending count is zero';
    execute immediate
'INSERT INTO INDEX_JOB_LOG VALUES (:ix, :id, sysdate, :msg)'
        using ixname, id, msg;
    id := id + 1;
    COMMIT;
else
    msg := 'Background job '||to_char(jobid)||
        ' starting key inversion';
    execute immediate
'INSERT INTO INDEX_JOB_LOG VALUES (:ix, :id, sysdate, :msg)'
        using ixname, id, msg;
    id := id + 1;
    COMMIT;

    doing_key := true;
    MDLAUX.UPDATEPENDINGINVERSIONS(ixname);
    doing_key := false;

    open csr for
'SELECT DATTIM,MSG FROM'||
' TABLE(MDLAUX.LOGTABLE('' ||ixname||'', 'ASC'))';
    loop
```

```

        fetch csr into dattim,msg;
        exit when csr%notfound;
        execute immediate
'INSERT INTO INDEX_JOB_LOG VALUES (:ix, :id, :dt, :msg)'
        using ixname, id, dattim, msg;
        id := id + 1;
    end loop;
    close csr;
    COMMIT;

    msg := 'Background job '||to_char(jobid)||
        ' finished key inversion';
    execute immediate
'INSERT INTO INDEX_JOB_LOG VALUES (:ix, :id, sysdate, :msg)'
        using ixname, id, msg;
    id := id + 1;
    COMMIT;
end if;

-- The example that follows shows how to update the FASTSEARCH index for
-- records which are pending inversion.

execute immediate
'SELECT MDLAUX.PENDINGFASTSEARCH(:ix) INTO :cnt FROM DUAL'
into cnt
using ixname;

if (not (cnt > 0)) then
    msg := 'Background job '||to_char(jobid)||
        ' skipping FASTSEARCH update,'||
        ' pending count is zero';
    execute immediate
'INSERT INTO INDEX_JOB_LOG VALUES (:ix, :id, sysdate, :msg)'
        using ixname, id, msg;
    id := id + 1;
    COMMIT;
else
    msg := 'Background job '||to_char(jobid)||
        ' starting FASTSEARCH update';
    execute immediate
'INSERT INTO INDEX_JOB_LOG VALUES (:ix, :id, sysdate, :msg)'
        using ixname, id, msg;
    id := id + 1;
    COMMIT;

    doing_fs := true;
    MDLAUX.UPDATEPENDINGFASTSEARCH(ixname);
    doing_fs := false;

    open csr for
'SELECT DATTIM,MSG FROM' ||
'TABLE(MDLAUX.LOGTABLE('' '||ixname||'', 'ASC'))';
    loop
        fetch csr into dattim,msg;

```

```
        exit when csr%notfound;
        execute immediate
'INSERT INTO INDEX_JOB_LOG VALUES (:ix, :id, :dt, :msg)'
        using ixname, id, dattim, msg;
        id := id + 1;
    end loop;
    close csr;
    COMMIT;

msg := 'Background job '||to_char(jobid)||
' finished FASTSEARCH update';
execute immediate
'INSERT INTO INDEX_JOB_LOG VALUES (:ix, :id, sysdate, :msg)'
using ixname, id, msg;
id := id + 1;
COMMIT;
end if;
exception
when others then
Fetch errors and write to INDEX_JOB_LOG
    if (doing_key) then
errmsg := 'Background job '||to_char(jobid)||
' key inversion failed';
    elsif (doing_fs) then
errmsg := 'Background job '||to_char(jobid)||
' FASTSEARCH update failed';
    else
errmsg := 'Background job '||to_char(jobid)||
' failed outside of key inversion'||
' and FASTSEARCH update';
        end if;
        if (sqlcode <> 0) then
            errmsg := errmsg||
: '||sqlerrm;
        end if;
SELECT MDLAUX.ERRORS INTO errors FROM DUAL;
    if (errors is not null) then
        errmsg := errmsg||
MDLAUX.ERRORS: '||errors;
    end if;
    execute immediate
'INSERT INTO INDEX_JOB_LOG VALUES (:ix, :id, sysdate, :msg)'
    using ixname, id, errmsg;
    id := id + 1;
    COMMIT;

    -- Cause job to pause until user looks to see:
    -- what went wrong
    DBMS_JOB.BROKEN(jobid, true);

    -- Could also raise an exception here, for example
    -- raise_application_error(-20100, errmsg);
end;
/
```

```
show errors;
```

Atom-To-Atom Maps

When you search for reactions with RSS, you normally add atom-to-atom maps or bond change marks or both to your query to retrieve only those reactions in which the transformation matches your query.

For example, to find reactions where a carbonyl is transformed to an alcohol, add atom-to-atom maps and a bond change mark.

However, for a query with atom-to-atom maps and bond change marks to be effective, all stored reactions also must contain accurate atom maps and bond marks. Direct includes a procedure you can call to generate atom-maps and bond change marks automatically on a selected set of reactions in a table. This process is referred to as automapping.

The automapping process involves looping over rows from a table, and then performing these steps for each row:

1. Fetches the reaction from the row.
2. Generates atom-to-atom maps and bond change marks for the reaction. This also produces a count of the number of changed and added maps and marks, and a status value which reflects the confidence of the automapper that the new maps and marks are correct.
3. Updates the row with the changed reaction if there are any changed or additional maps or marks.
4. Inserts the row ROWID, and the automapper count and status values into another table.

Tip: The algorithm that the automapper uses can become confused by unbalanced reactions. For this reason, it is important that you receive information from the program about the status of each mapping. The automap procedure handles this by writing information about each mapping to a second table. The section that follows describes this process in greater detail.

Using the Automapper

Before you can use the automapper procedure, create a table to hold the status information that the automapper requires. The section that follows explains the procedure to use.

Create a Status Information Table

Create the status information table before you attempt to use the automapper.

To create a status information table, issue `CREATE TABLE` as shown:

```
CREATE TABLE STATUS_TABLE (RID ROWID, STATUS NUMBER, CHANGES NUMBER);
```

You can specify an existing table to hold status information. However, you must truncate the table before you can use it.

Use the Automapper

To use the automapper, issue `MDLAUX.REGENAAMAPS` as shown:

```
CALL MDLAUX.REGENAAMAPS('RXN_TABLE', 'RXN_COLUMN', 'STATUS_TABLE', ['AUTOMAP_MODE' [, 'ROW_RESTRICTION_QUERY']]);
```

where:

- `RXN_TABLE` specifies the name of the table that contains reactions.
- `RXN_COLUMN` specifies the name of the reaction column.

- **STATUS_TABLE** specifies the name of the status table. This table receives information about the status of each automap operation.

If you only specify these three arguments, the automap procedure processes all of the reactions in the table using an automap mode `RegenAlter`. The fourth and fifth arguments are optional. They are explained in more detail in the sections that follow. For more information on the `RegenAlter` mode, see the table that follows.

- **AUTOMAP_MODE** controls how the automapper interprets any existing atom-to-atom maps or bond marks in a reaction.

This optional argument takes one of the values listed in the following table.

Value	Explanation
NULL	Same as <code>RegenAlter</code> .
<code>RegenAlter</code>	Uses existing atom-to-atom maps and bond marks to guide the mapping process. Modifies existing maps and marks if necessary.
<code>RegenAuto</code>	Disregards any existing atom-to-atom maps and bond marks. Treats each reaction as if it is initially unmapped.
<code>Default</code>	Uses existing atom-to-atom maps and bond marks to guide the mapping process. Does not change existing maps and marks. Thus, if the automapper decides that an existing map or mark should be changed or removed, it stops automapping without modifying the reaction.

- **ROW_RESTRICTION_QUERY** is an optional argument that controls which rows in the reaction table are automapped.

Add a `WHERE` clause to restrict the rows. An example of how to construct a query using this argument with a `WHERE` clause follows:

```
SQL> SELECT ROWID, RXN_COLUMN FROM RXNS WHERE ROW_RESTRICTION_QUERY;
```

For example, if you set `ROW_RESTRICTION_QUERY` to the value `RXNREGNO BETWEEN 1001 AND 2000`, Direct only automaps those reactions where the value of the `RXNREGNO` column is between 1001 and 2000.

Note: When you include the restriction query argument, also specify an automap mode.

Lock and Commit

The automap procedure locks each row as it processes the reaction in that row. It then commits the transaction. The `COMMIT` statement releases the lock on the reaction. Direct uses an autonomous transaction. This ensures that other transactions that the user might have open are not affected.

Examine the Automap Status

Use the status table to determine if you should examine any reactions for errors in the atom-to-atom mapping process. Each reaction that is mapped has an entry in the status table. The `RID` column contains the row ID of the reaction, the `STATUS` column contains the automap status, and the `CHANGES` column contains the number of changes that were made to the reaction. Examine any reaction that has a `STATUS` value of 4 or greater. Any reaction for which the `CHANGES` value is greater than 0 (zero) has had changes made to existing atom-to-atom maps or bond marks. If you think that the original maps

and marks of such a reaction were correct, examine these reactions as well. If the CHANGES value is (-1), then only additions, not modifications, have been made to the existing maps or marks.

For more information about map changes and map status, see *BIOVIA Direct Reference Guide*.

Chapter 5:

Local Partitioned Index

Creating a Partitioned Domain Index

Oracle Enterprise Edition allows for the creation of partitioned tables, that is, tables that have been decomposed into smaller and more manageable pieces called partitions.

BIOVIA Direct has always allowed the creation of a domain index over an entire partitioned table. Direct can also create a partitioned domain index where the domain index itself has been decomposed into partitions that match the table partitions. Some advantages to using a partitioned domain index are:

- Domain index partitions can be built or rebuilt in parallel, speeding up the task of building an index
- Domain index partitions can be searched in parallel, speeding up structure searches
- Oracle only searches those index partitions which it knows can satisfy the query, speeding up structure searches

A Direct domain index may be partitioned along with the base table. To create a local partitioned index, first create a range-partitioned table (Oracle only supports local domain indexes with `PARTITION BY RANGE`):

```
create table mdctest (extreg varchar2(30) primary key, ctab blob)
partition by range (extreg)
(partition tp1 values less than ('M'),
partition tp2 values less than ('maxvalue'));
```

You can partition by date range, for example:

```
create table projects_by_year (
regno number (9),
f_date date not null,
ctab blob,
primary key (regno)
partition by range (f_date)
(partition year2012 VALUES LESS THAN ('01-JAN-2013'),
partition year2013 VALUES LESS THAN ('01-JAN-2014'),
partition year2014 VALUES LESS THAN ('01-JAN-2015'))
```

Then create the domain index with the `LOCAL` clause. You will get a non-partitioned domain index if you do not specify the `LOCAL` clause. No partition arguments are required, the default is to name the index partitions the same as the table partitions and the top-level `PARAMETERS` clause will then be used for all of the partitions:

```
create index mdctest_molix on mdctest (ctab)
indextype is C$DIRECT2021.mxixmdl
parameters ('tablespace=xspace')
local;
```

If you wish, the index partitions may be named and have their own `PARAMETERS` clauses:

```
create index mdctest_molix on mdctest (ctab)
indextype is C$DIRECT2021.mxixmdl
parameters ('tablespace=xspace')
local
```

```
(partition ip1 parameters ('lob_tablespace=lobts1'),
partition ip2 ('lob_tablespace=lobts2'));
```

In this example partition IP1 is created with parameters 'tablespace=xspace lob_tablespace=lobts1' meaning that secondary tables and indexes will use tablespace XSPACE, but that its Fastsearch LOB column will use tablespace LOBTS1. Partition IP2 is created with parameters 'tablespace=xspace lob_tablespace=lobts2', its Fastsearch LOB column will use a different tablespace LOBTS2.

After partitions are created there are no top-level parameters stored in the index's property table, i.e. there are no parameters which apply to all partitions. Each partition has its own complete set of parameters. This means that each partition will have its own UNIQUE setting and local environment. The environment between partitions could be different if the partitions were added to the table at different times and the cartridge global environment was changed between when partitions were added.

Please refer to the following when indexing large partitions: [Creating Indexes on Large Partitions](#)

Creating Indexes on Large Partitions

Note: Because of SQL performance problems when using ORDER BY clauses with partitions, partitioned domain index creation does not use an ordered selection of data from the molecule or reaction table. As a result index creation will not restart if an error occurs. When creating an index on very large partitions you may encounter an "ORA-1555 Snapshot too old" error and indexing will terminate. The index might still exist and should be dropped after this occurs.

There are several ways to work around this problem:

- Increase the Oracle undo_retention parameter and undo tablespace size, undo_retention must be longer than the total index partition creation time.
- Create the index using PARAMETERS ('DISABLE=FASTSEARCH'), then use MDLAUX.RECREATEFASTSEARCH on each index partition after index creation has completed. Most of the time needed to create a domain index is used in fastsearch data creation, this will significantly shorten the index partition creation step.
- Create the index using PARAMETERS ('RETRYFETCH'). This will use an ordered select statement which is a process that can be restarted, but may take much longer to execute.

Adding a New Partition

Add a new table and index partition using ALTER TABLE ADD PARTITION UPDATE INDEXES, for example:

```
alter table mdctest add partition tp3 values less than ('R')
update indexes;
```

or:

```
alter table mdctest add partition tp3 values less than ('R')
update indexes
(mdctest_molix
(partition ip3 parameters ('tablespace=zspace')));
```

The UPDATE INDEXES clause is essential or a new partition in the domain index will not be created! If the argument to UPDATE INDEXES is left out of the SQL command as in the first example, the new index partition will then get the name of the table partition. Adding the argument allows the specification of a partition name and PARAMETERS clause.

Splitting a Partition

Split an existing table and its associated index partition using `ALTER TABLE SPLIT PARTITION UPDATE INDEXES`, for example:

```
alter table mdctest split partition tp1
at ('E')
into (partition tp1a, partition tp1b)
update indexes
(mdctest_molix
(partition ip1a,
partition ip1b parameters ('tablespace=qspace')));
```

Again `UPDATE INDEXES` is essential, and its argument is not required. If a new partition contains data, or if a `PARAMETERS` argument was specified for it, it will automatically be rebuilt.

Splitting a partition is handled internally by dropping the partition to be split and adding two new partitions. The `PARAMETERS` associated with the old partition are used for the two new ones. The `PARAMETERS` argument in the `UPDATE INDEXES` clause, if any, is provided by Oracle during the automatic rebuild which occurs after the split.

Rebuilding a Partition

Rebuild an individual index partition using `ALTER INDEX REBUILD PARTITION`, for example:

```
alter index mdctest_molix rebuild partition ip1a
parameters ('tablespace=gspace');
```

Partitions may be rebuilt in parallel by issuing the `ALTER INDEX` command for each partition in a separate oracle session.

You may not rebuild a partitioned domain index without specifying a partition, i.e. Oracle will reject the following command if `mdctest_molix` is a local partitioned index:

```
alter index mdctest_molix rebuild
parameters ('tablespace=gspace');
```

You may use `ALTER INDEX` without the `REBUILD` option on a partitioned index, however Direct only supports this with the `RENAME` option. Any attempt to modify the Direct parameters will return a warning and do nothing, you must use `ALTER INDEX REBUILD PARTITION` to modify parameters.

Exchanging Table and Partition

Exchange a table partition and its associated domain index partition with a table and its associated domain index using `ALTER TABLE EXCHANGE PARTITION WITH TABLE`. No `UPDATE INDEXES` clause is used. The new exchanged domain index is valid and usable without any further action.

The new exchanged domain index partition is valid, however Oracle will mark it as `UNUSABLE`, and it requires a rebuild before it can be used. Because the `ROWIDs` in the new exchanged table partition are still valid, you may rebuild the domain index partition using the `NOACTION` parameter. This will mark it as `USABLE` without actually rebuilding domain index data such as Fastsearch information.

The following example shows creation of a partitioned table and index, a non-partitioned table and index, and exchanging the two:

```
create table mdctest (extreg varchar2(30) primary key, ctab blob)
partition by range (extreg)
(partition tp1 values less than ('M'),
```

```

partition tp2 values less than ('maxvalue'));
create index mdctest_molix on mdctest (ctab)
indextype is C$DIRECT2021.mxixmdl
local
(partition ip1, partition ip2);
create table mdctest_notpartitioned (extreg varchar2(30) primary key, ctab
blob);
create index mdctest_notpartitioned_ix on mdctest_notpartitioned (ctab)
indextype is C$DIRECT2021.mxixmdl;
alter table mdctest
exchange partition tp2 with table mdctest_notpartitioned;
alter index mdctest_molix rebuild partition ip2 parameters ('noaction');

```

Direct Index Maintenance Procedures

MDLAUX.RECREATEFASTSEARCH, MDLAUX.RECREATEKEYS, MDLAUX.UPDATEPENDINGFASTSEARCH, MDLAUX.UPDATEPENDINGINVERSIONS and several other commands have been modified so that they accept a partition name in the argument. If no partition name is specified these functions will operate on all partitions in sequential order (no parallelization), otherwise they will operate only on the specified partition. For example to update pending Fastsearch records for all partitions use:

```
execute mdlaux.updatependingfastsearch('mdctest');
```

or

```
execute mdlaux.updatependingfastsearch('mdctest_molix');
```

To update records only in table partition TP1, which has corresponding index partition IP1 use:

```
execute mdlaux.updatependingfastsearch('mdctest partition tp1');
```

or

```
execute mdlaux.updatependingfastsearch('mdctest_molix partition ip1');
```

Full syntax for the table or index partition argument is:

```
'[""]schema[""].[""]objname[""] [ PARTITION [""]partname[""] ]'
```

Parallelization of Index Maintenance

ALTER INDEX REBUILD PARTITION and other maintenance operations such as MDLAUX.RECREATEFASTSEARCH can be run in parallel over separate partitions at the same time. To do this, run each command in a separate Oracle session, and if using an MDLAUX function or procedure be sure to specify the partition name along with the table or index name. For example:

```
window1: sqlplus bigdbuser/bigdbuser
```

```
window1: SQL> execute mdlaux.recreatefastsearch('bigdb partition p1');
```

```
window2: sqlplus bigdbuser/bigdbuser
```

```
window2: SQL> execute mdlaux.recreatefastsearch('bigdb partition p2');
```

Parallelizing Index Partition Creation and Searching

You may specify a PARALLEL N clause when creating a partitioned domain index. This tells Oracle to use N processes when creating the index or searching it.

Not all searches can use parallelization, it depends on the search query and how the table is partitioned. Use `EXPLAIN PLAN` to determine if a specific query will be parallelized.

Parallel DDL must be enabled in order to create index partitions in parallel, and parallel query must be enabled in order to search index partitions in parallel. These are both enabled by default, but the DBA could have disabled them. If not enabled, the DBA will need to enable them.

This example sets the level of parallelization of the domain index to use eight processes:

```
create index mdctest_molix on mdctest (ctab)
indextype is C$DIRECT2021.mxixmdl
parameters ('tablespace=xspace')
local
(partition ip1 parameters ('lob_tablespace=lobts1'),
partition ip2 (lob_tablespace=lobts2'))
parallel 8;
```

Notes:

FLEXMATCH and RXNFLEXMATCH searches perform better when parallel searching is disabled. Disable parallel searching for a query by using the `NO_PARALLEL_INDEX(tablespec)` hint. You must not specify the index name in the hint or Oracle will ignore the hint. Specify only the table name in the hint. For example:

```
select /*+ no_parallel_index(mega20mols4part) */ cdbregno
from mega20mols4part
where flexmatch(ctab, 'c:/query.mol', 'match=all') = 1;
```

Structure of Partitioned Domain Index

The secondary tables in the domain index have the same structure columns and attributes, however most of them will be partitioned with system managed partitions which means that Oracle provides the partition name required for DML and DDL on the secondary tables. The `indexname_PROP` and `indexname_LOG` tables are not partitioned.

The Direct sequence cannot be partitioned, thus there will be one sequence created for each partition. The correlation of sequence name and partition is maintained in the `indexname_PROP` secondary table.

Limitations with Partitioned Domain Indexes

- The domain index creation parameter `UNIQUE` only applies within a partition. It is not possible for a partitioned domain index to perform duplicate checking over the entire index, i.e. over all partitions. Duplicate checking is only performed for the partition in which the molecule or reaction is inserted. Thus when a domain index is created with `PARAMETERS ('UNIQUE=MATCH=ALL')` each index partition will not contain duplicate structures, however the same structure may appear in more than one partition. If you wish to restrict duplicates over all partitions you can either create a non-partitioned domain index over the entire table, or your registration application must perform the duplicate check by searching over the table.
- Each domain index partition has its own local chemical environment, or its own periodic table, salt definitions file, etc. Using separate chemical environments greatly simplifies domain index operations, especially `EXCHANGE PARTITION AND TABLE`. This has two consequences:
 - Functions such as `MDLAUX.MOLWT` and `MDLAUX.MOLFMLA` accept an index and a partition name in the first argument. When only an index name is provided the chemical environment from the first index partition is used.

- Direct functions which require a chemical environment take a first argument which currently can be one of:

NULL Use global chemical environment

'*indexname*' Use chemical environment contained in specified index

'*tablename*' Use chemical environment contained in Direct domain index associated with specified table

- The argument syntax is extended as described in [Direct Index Maintenance Procedures](#) to take an optional partition name. Unlike the index maintenance procedures, if the partition name is missing then the chemical environment from the first partition in the domain index is used. For these functions the argument now becomes:

NULL Use global chemical environment

'*indexname* PARTITION *partitionname*' Use chemical environment contained in specified index partition

'*tablename* PARTITION *partitionname*' Use chemical environment contained in Direct domain index partition associated with specified table partition

'*indexname*' Use chemical environment contained in first index partition

'*tablename*' Use chemical environment contained in Direct domain index partition associated with first table partition

- The following functions require a chemical environment and use the new extended syntax:

```
MDLAUX.GETINDEXENVFILE
MDLAUX.SGROUPFIELDS
MDLAUX.MOLTOMOLFILE
MDLAUX.RXNTORXNFILE
MDLAUX.RXNKEYS
MDLAUX.MOLWT
MDLAUX.MOLFMLA
MDLAUX.MOLKEYS
MDLAUX.MOLNEMAKEY
MDLAUX.ISOTOPICFORMULA
MDLAUX.MONOISOTOPICMASS
MDLAUX.ISSEQUENCE
```

- If the global environment is changed in between when partitions are added you may end up with partitions having different environments. If this occurs you may want to rebuild the environment in all partitions so that it matches the current global environment, using ALTER INDEX REBUILD PARTITION PARAMETERS ('ENVIRONMENT'). For example:

```
alter index mdctest_molix partition ip1 rebuild ('environment');
alter index mdctest_molix partition ip2 rebuild ('environment');
alter index mdctest_molix partition ip3 rebuild ('environment');
```

- Oracle does not allow the ALTER INDEX REBUILD command on a partitioned index, you must rebuild each partition separately by adding the PARTITION keyword and partition name to the SQL command. You can take advantage of this and rebuild each partition in a separate process (e.g. telnet session or CMD prompt), see [Parallelization of Index Maintenance](#).
- Oracle does allow ALTER INDEX without the REBUILD option to be used with a partitioned index, however Direct only supports the ALTER INDEX RENAME command, for example:
alter index mdctest_molix rename to largedb_molix;

- Operations that in previous versions of Direct used alter index `mdctest_molix` parameters ('updatependingfastsearch'); must now be done using `ALTER INDEX REBUILD`, for example:
alter index mdctest_molix rebuild partition ip1 parameters ('updatependingfastsearch');
- Data Pump and Classic Import are not able to import the system partitioned secondary tables which are generated when a Direct partitioned domain index is created. `MDLAUX.PREPAREINDEXEXPORT` will raise an exception. Export and Import are still available, but Export will not include any secondary table data and Import will have to rebuild the entire index.
- Dropping a partitioned index when it has one or more partitions in an `INVALID` state will cause the index name to be removed from the Oracle dictionary tables and views, however the partition names will not be removed. If you connect as user `SYS` and look at `OBJ$`, you will see the spurious partition entries, these should have been removed by Oracle:

```
column subname format a10
select o.name,o.subname,o.type#,o.obj# from obj$ o, user$ u
where o.owner#=u.user# and u.name='TESTSCHEMA' and o.name='MDCTESTIX';
```

NAME	SUBNAME	TYPE#	OBJ#
MDCTESTIX	IP1	20	513989
MDCTESTIX	IP2	20	513990

Note: It is not possible for a partitioned domain index to perform duplicate checking over the entire index. Duplicate checking is only performed in the partition in which the molecule or reaction is inserted. If overall duplicate checking is needed, an external application must provide it.

Chapter 6:

Upgrading Indexes

To upgrade a Direct 6, 7 or 8 molecule or reaction table, see [Upgrading a Direct 6, 7, or 8 Molecule or Reaction Table](#). You will update your chemical environment, create a copy of your table, and use the program `directupgrade` to update the chemistry in the table and create a new Direct 9.x domain index. It does not matter whether there is a domain index present on the Direct 6, 7 or 8 table.

If you are upgrading a Direct 9 or later molecule or reaction table that has an existing domain index visible in Oracle's `USER_INDEXES` views, and all of its secondary tables, you must upgrade the indextype using `MDLAUX.UPGRADEINDEXES_PREPARE` and `MDLAUX.UPGRADEINDEXES_UPGRADE` described in the section [Cartridge Management Functions and Procedures](#), followed by updating the chemical environment with a new key definition file (this rebuilds `KEYS`) and rebuilding `FASTSEARCH`. This procedure is described in the following section, [Upgrading a Direct Table with an Existing Domain Index](#).

If you are upgrading a Direct 9 or later molecule or reaction table which does not have an existing domain index visible in Oracle's `USER_INDEXES` views, for example a table created from a dump file on a machine where Direct is not installed, refer to the section

Upgrading a Direct Table with an Existing Domain Index

Before proceeding to upgrade tables to Direct 2021, if you have customized your Direct periodic table (PTABLE) or salt definition file (SALTS) chemical environment files, copy those files into the new Direct chemical environment. See [Setting the Cartridge C\\$DIRECT2021 Environment](#) for more information.

1. Follow the procedure for [MDLAUX.UPGRADEINDEXES_PREPARE/UPGRADE](#) to upgrade your domain index's indextype.
2. Direct 9.1 added additional keys to support substructure search and similarity search of structures containing haptic (pi-system) bonding. If you are upgrading from Direct 9.0, add these keys to your domain index with `ALTER INDEX index-name REBUILD [PARTITION partition-name] PARAMETERS ('ENVIRONMENT')`. For example:

```
SQL> alter index moltable_ix rebuild parameters ('environment');
```

This step loads the new key definitions into the domain index and rebuilds the substructure/similarity search keys in the index. For more information, see [Parameters for Alter Index Rebuild](#).

3. If you are upgrading from Direct 2018 or earlier, enable biopolymer sequence text searching in the domain index with `ALTER INDEX index-name REBUILD [PARTITION partition-name] PARAMETERS ('ENABLE=SEQUENCETEXT')`. For example:

```
SQL> alter index moltable_ix rebuild parameters ('enable=sequencetext');
```

If you are upgrading a partitioned domain index you must rebuild each partition in the index.

4. Rebuild inverted key substructure search, indexing data in your domain index with `MDLAUX.RECREATEKEYS`. For example:

```
SQL> execute mdlaux.recreatekeys('moltable_ix');
```

5. Rebuild substructure search (Fastsearch) indexing data in your domain index with `MDLAUX.RECREATEFASTSEARCH`. For example:

```
SQL> execute mdlaux.recreatefastsearch('moltable_ix');
```

This step is the longest part of the upgrade process and generally takes about an hour for each million structures in the table.

- If you are upgrading a molecule table from Direct 2019 or earlier and you have RNA or DNA structures in the table, the structures must be re-registered to compute additional indexing information. Starting with the Direct 2020 release, the `ISRNA(molecule-column)` operator returns 1 if the structure contains RNA/DNA and 0 otherwise, and is used to determine which structures must be re-registered. The general syntax is:

```
UPDATE tablename SET ctabname=ctabname WHERE ISRNA(ctabname)=1
```

For example:

```
SQL> update moltable set ctab=ctab where isrna(ctab)=1;
```

```
SQL> commit;
```

If any rows were updated by Oracle, you must update the pending inversions and Fastsearch data with `MDLAUX.UPDATEPENDINGINVERSIONS('tablename')` and `UPDATEPENDINGFASTSEARCH('tablename')`. For example:

```
SQL> execute mdlaux.updatependinginversions('moltable');
```

```
SQL> execute mdlaux.updatependingfastsearch('moltable');
```

- If you are upgrading a molecule or reaction table from Direct 2020 or earlier, you will need to re-register molecules or reactions that contain symmetrically tetra-substituted double bonds. These bonds cannot be cis or trans, but in Direct 2020 and earlier they were stored in the table with an arbitrary cis or trans designation. In Direct 2021 and later they are perceived as being either and this will not match the previous cis or trans value in a FLEXMATCH search. The purpose of this step is to locate these molecules and re-register them with the correct designation so that FLEXMATCH will find them in the database.

Locate these molecules or reactions using `mdlaux.scanindex` with the 'CTAB' argument and update structures that do not find themselves, but which do find themselves when ignoring stereochemistry.

For example:

```
SQL> execute mdlaux.scanindex('mdctestix', 'ctab');

BEGIN mdlaux.scanindex('mdctestix', 'ctab'); END;
*
ERROR at line 1:
MDL-1152: Molecule CTAB scan has errors, use MDLAUX.LOGTABLE to view errors
MDL-1112: MDLAUX.SCANINDEX has warnings or errors, use MDLAUX.LOGTABLE to view them
ORA-06512: at "C$DIRECTS2021.MDLAUXOP", line 4895
ORA-06512: at "C$DIRECTS2021.MDLAUXOP", line 4885
ORA-06512: at line 1
```

```
SQL> select msg from table(mdlaux.logtable('mdctestix', 'asc'));
```

```
MSG
```

```
-----
Scanning molecule index MDCTESTIX on table MDCTEST.MDCTEST
Scanning ROWID conversion table
ROWID conversion table scan complete, there are 2 rows
```

```

Scanning molecule CTABs
Molecule at ROWID=AANqImAAPAAAADmAAA cannot find itself with FLEXMATCH
Molecule at ROWID=AANqImAAPAAAADmAAA does find itself with FLEXMATCH
when ignoring stereochemistry and data Sgroups (IGNORE=STE,DAT)
Previous message occurred at ROWID=AANqImAAPAAAADmAAB
Molecule CTAB scan has errors

SQL> update mdctest set ctab=mol(molfile(ctab)) where rowid =
'AANqImAAPAAAADmAAB';

1 row updated.

SQL> commit;

Commit complete.

```

Upgrading a Direct Table without an Existing Domain Index

If you import a dump file containing a Direct 9.0 or 9.1 table and domain index on a machine where Direct 9.0 or 9.1 has never been installed or has been completely uninstalled, the Direct domain index will not be created by the import. If you import the dump file on a machine where Direct was installed, its binary files on disk have been removed, but the cartridge schema was not dropped then import will create an index but it will be invalid. In this case you must drop the index that was created using the **FORCE** option before proceeding with the upgrade:

```
SQL> drop index indexname force;
```

The method for upgrading this table to Direct 2021 depends on whether or not the domain index secondary tables are present. To determine this use the following SQL statement, replacing *indexname* with the name of the domain index:

```
SQL> create index indexname on tablename (ctabcolumn)
2 indextype is C$DIRECT2021.mxixmdl parameters ('noaction');
```

To upgrade a table of reactions use the following statement:

```
SQL> create index indexname on tablename (ctabcolumn)
2 indextype is C$DIRECT2021.rxixmdl parameters ('noaction');
```

If the index creation is successful, you will have a Direct 2021 domain index on the table. You must still follow steps 2 and 3 in the [Upgrading an Table with an Existing Domain Index](#) section to complete the domain index upgrade.

Chapter 7:

Exporting and Importing

This chapter explains how to export and import with your Direct tables.

Export and Import

Use the Oracle Database Server Export and Import utilities to transfer or back up tables that contain molecules or reactions. For information on how to use Export and Import, see your Oracle Database Server documentation.

When you use the Oracle Database Server Export utility to export a table that contains a molecule domain index, Export normally only writes the SQL command to create the index into the exported file. Export does not normally write any of the secondary objects, such as the inverted keys or Fastsearch tables, to the exported file.

When you use the Oracle Database Server Import utility to import the exported file, Import recreates the molecule domain index. For a molecule table containing hundreds of thousands of rows, creating the domain index can be a lengthy operation. The amount of time depends on the quantity of data as well as the hardware at your site.

To reduce the amount of time that is required to export data, call `MDLAUX.PREPAREINDEXEXPORT` before you use the Oracle Database Server Export utility. This function creates a special table that greatly speeds up the process of recreating the domain index, which occurs during Import.

`MDLAUX.PREPAREINDEXEXPORT`:

- Causes the Oracle Database Server Export utility to include domain index secondary tables in the exported data file
- Prepares a special table, `index_name_CRCV`, which the Import utility uses to recalculate ROWIDs stored in a domain index secondary table.

For more information, see [MDLAUX.PREPAREINDEXEXPORT](#).

When you use the Oracle Import utility (`imp`) to import an exported file that contains the domain index secondary tables and `indexname_CRCV` table, the molecule domain index does not need to be fully recreated. Instead, only the stored ROWIDs need to be recomputed, and some tables must be copied to ensure that they are properly marked as secondary in the Oracle Database Server data dictionaries.

IMPORTANT!

If you are exporting and importing using Oracle Original Export and Original Import utilities (`exp` and `imp`), the import might fail to create secondary tables that contain LOB columns. If a tablespace used for a LOB column in a secondary table does not exist in the Oracle instance where the dump file is imported, Original Import will fail to create the table and the import will terminate.

Oracle Data Pump Import does not have this problem because it allows the user to remap tablespace names during import, and is the recommended method for exporting and importing data.

Original Import can still be used. Secondary tables containing LOB columns must be created manually before running Original Import. To work around this problem, list the contents only of the dump file as explained in your Oracle Database Server documentation for Original Import. Use the information that is displayed to create empty tables with LOB columns. Then reimport the tables from the dump file.

Secondary tables with LOB columns are:

- `indexname_PROP`
- `indexname_CONV`
- `indexname_CTAB`
- `indexname_IKY2`
- `indexname_IRKY`
- `indexname_FSIX`
- `indexname_FSUP`
- `indexname_FSDL`

Thus, importing and exporting data using the Oracle utilities has these steps:

1. Run the function `MDLAUX.PREPAREINDEXEXPORT` to prepare the data for export.
2. If using Original Import, create tables with LOB columns in the schema in which to import the tables.
3. Run the Oracle Import utility (`impd` or `imp`) to import the data from the dump file.

If Import Fails to Create the Domain Index

If the Import utility fails to create the molecule or reaction domain index:

1. Drop the index and all domain index secondary objects which were created.
2. Drop the table and any other tables that were successfully imported.
3. Rerun the import.

Alternatively, you can issue the SQL command `CREATE INDEX` to create the domain index manually.

IMPORTANT! If you issue `CREATE INDEX`, do *not* use the parameters that were printed by the Import utility. Instead issue `CREATE INDEX` using one of the following options.

- Either issue `CREATE INDEX` with *no* parameters, or
- Use only those parameters that you know were used to create or alter the index on the molecule table which was exported.

Domain index secondary objects might comprise:

```
SEQUENCE indexname_SQNC
TABLE indexname_PROP
TABLE indexname_LOG
TABLE indexname_CTAB
TABLE indexname_CONV
TABLE indexname_CCLK
TABLE indexname_FLEX
TABLE indexname_RFLX
TABLE indexname_SGRP
TABLE indexname_FMLA
TABLE indexname_SKY2
TABLE indexname_IKY2
TABLE indexname_FSUP
TABLE indexname_FSDL
TABLE indexname_FSIX
TABLE indexname_GENX
```

where `indexname` is the name of the molecule domain index, truncated to 24 characters.

Importing Data and the Domain Index

Note: The following SQL statement sets up the domain index so that expdp will export the domain index data.

```
MDLAUX.PREPAREINDEXEXPORT('SAMPLE2D_IX')
```

```
-----
Index import information written to: "DCSAMPLES"."SAMPLE2D_IX_CRCV"
Include this table in your export. After export drop this table.
```

```
F:\work>expdp dcsamples/ dumpfile=sample2d.dmp directory=dpump_dir1 tables=
(sample2d,sample2d_ix_crcv)
```

```
Export: Release 11.2.0.1.0 - Production on Mon June 17 11:23:42 2013
```

```
Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights
reserved.
```

```
Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 -
64bit Production
```

```
With the Partitioning, OLAP, Data Mining and Real Application Testing
options
```

```
Starting "DCSAMPLES"."SYS_EXPORT_TABLE_01": dcsamples/*****
dumpfile=sample2d.dmp directory=dpump_dir1 tables=(sample2d,sample2d_ix_
crcv)
```

```
Estimate in progress using BLOCKS method...
```

```
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
```

```
Total estimation using BLOCKS method: 704 KB
```

```
Processing object type TABLE_EXPORT/TABLE/TABLE
```

```
Processing object type TABLE_EXPORT/TABLE/GRANT/OWNER_GRANT/OBJECT_GRANT
```

```
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
```

```
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
```

```
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
```

```
Processing object type TABLE_EXPORT/TABLE/INDEX/TABLE
```

```
Processing object type TABLE_EXPORT/TABLE/INDEX/TABLE_DATA
```

```
Processing object type TABLE_EXPORT/TABLE/INDEX/DOMAIN_INDEX/INDEX
```

```
. . exported "DCSAMPLES"."SAMPLE2D"                357.8 KB      382
rows
```

```
. . exported "DCSAMPLES"."SAMPLE2D_IX_CRCV"          15.39 KB      382
rows
```

```
. . exported "DCSAMPLES"."SAMPLE2D_IX_LOG"            6.882 KB       14
rows
```

```
. . exported "DCSAMPLES"."SAMPLE2D_IX_PROP"          111.7 KB       25
rows
```

```
. . exported "DCSAMPLES"."SAMPLE2D_IX_CONV"          11.70 KB      382
rows
```

```
. . exported "DCSAMPLES"."SAMPLE2D_IX_CTAB"           0 KB           0
rows
```

```
. . exported "DCSAMPLES"."SAMPLE2D_IX_SKY2"           0 KB           0
rows
```

```
. . exported "DCSAMPLES"."SAMPLE2D_IX_IKY2"           52 KB          1
rows
```

```
. . exported "DCSAMPLES"."SAMPLE2D_IX_FLEX"          11.49 KB      442
```

```

rows
. . exported "DCSAMPLES"."SAMPLE2D_IX_SGRP"          0 KB          0
rows
. . exported "DCSAMPLES"."SAMPLE2D_IX_NEC"          33.07 KB        382
rows
. . exported "DCSAMPLES"."SAMPLE2D_IX_FMLA"         16.40 KB        382
rows
. . exported "DCSAMPLES"."SAMPLE2D_IX_FSUP"          0 KB          0
rows
. . exported "DCSAMPLES"."SAMPLE2D_IX_FSDL"          0 KB          0
rows
. . exported "DCSAMPLES"."SAMPLE2D_IX_FSIX"          0 KB          0
rows
. . exported "DCSAMPLES"."SAMPLE2D_IX_CCLK"          0 KB          0
rows
Master table "DCSAMPLES"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded

```

```

*****
**

```

Dump file set for DCSAMPLES.SYS_EXPORT_TABLE_01 is:

F:\WORK\TEMP\SAMPLE2D.DMP

Job "DCSAMPLES"."SYS_EXPORT_TABLE_01" successfully completed at 11:24:13

Note: The follow SQL command drops the CRVC table.

```
SQL> drop table sample2d_ix_crcv purge;
```

Table dropped.

Note: Import the dump file from above into a different schema.

```
F:\work>impdp test/test dumpfile=sample2d.dmp directory=dpump_dir1 full=y
remap_schema=dcsamples:test
```

Import: Release 11.2.0.1.0 - Production on Mon Jun 17 11:30:20 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit

Production

With the Partitioning, OLAP, Data Mining and Real Application Testing options

Master table "TEST"."SYS_IMPORT_FULL_01" successfully loaded/unloaded

Starting "TEST"."SYS_IMPORT_FULL_01": test/***** dumpfile=sample2d.dmp directory=dpump_dir1 full=y remap_schema=dcsamples:test

Processing object type TABLE_EXPORT/TABLE/TABLE

Processing object type TABLE_EXPORT/TABLE/TABLE_DATA

```
. . imported "TEST"."SAMPLE2D"          357.8 KB        382
```

rows

```
. . imported "TEST"."SAMPLE2D_IX_CRCV"    15.39 KB        382
```

rows

Processing object type TABLE_EXPORT/TABLE/GRANT/OWNER_GRANT/OBJECT_GRANT

```

Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/INDEX/TABLE
Processing object type TABLE_EXPORT/TABLE/INDEX/TABLE_DATA
. . imported "TEST"."SAMPLE2D_IX_LOG"                6.882 KB      14
rows
. . imported "TEST"."SAMPLE2D_IX_PROP"                111.7 KB      25
rows
. . imported "TEST"."SAMPLE2D_IX_CONV"                11.70 KB      382
rows
. . imported "TEST"."SAMPLE2D_IX_CTAB"                 0 KB          0
rows
. . imported "TEST"."SAMPLE2D_IX_SKY2"                 0 KB          0
rows
. . imported "TEST"."SAMPLE2D_IX_IKY2"                 52 KB         1
rows
. . imported "TEST"."SAMPLE2D_IX_FLEX"                11.49 KB      442
rows
. . imported "TEST"."SAMPLE2D_IX_SGRP"                 0 KB          0
rows
. . imported "TEST"."SAMPLE2D_IX_NEC"                 33.07 KB      382
rows
. . imported "TEST"."SAMPLE2D_IX_FMLA"                16.40 KB      382
rows
. . imported "TEST"."SAMPLE2D_IX_FSUP"                 0 KB          0
rows
. . imported "TEST"."SAMPLE2D_IX_FSDL"                 0 KB          0
rows
. . imported "TEST"."SAMPLE2D_IX_FSIX"                 0 KB          0
rows
. . imported "TEST"."SAMPLE2D_IX_CCLK"                 0 KB          0
rows
Processing object type TABLE_EXPORT/TABLE/INDEX/DOMAIN_INDEX/INDEX
Job "TEST"."SYS_IMPORT_FULL_01" successfully completed at 11:30:30

```

Note: The following SQL file is a check to see that the index is correct.

```

SQL> column table_name format a20
SQL> column index_name format a20
SQL> column ityp_owner format a15
SQL> select table_name, index_name, ityp_owner, domidx_status as status,
domidx_opstatus as opstatus from user_indexes where ityp_owner is not null;

```

TABLE_NAME	INDEX_NAME	ITYP_OWNER	STATUS
OPSTAT			
-----	-----	-----	-----
-			
SAMPLE2D	SAMPLE2D_IX	C\$DIRECT2021	VALID
VALID			

Chapter 8:

Logging Information

Logging Index Operations

A Direct molecule domain index contains a table that logs status information and error messages when a user performs the following operations:

- `CREATE INDEX`
- `ALTER INDEX`
- `MDLAUX.SCANINDEX`
- `MDLAUX.UPDATEPENDINGINVERSIONS`
- `MDLAUX.UPDATEPENDINGFASTSEARCH`
- `MDLAUX.RECREATEKEYS`
- `MDLAUX.RECREATEFASTSEARCH`

To view this status information, call the table function `MDLAUX.LOGTABLE` as explained in [MDLAUX.LOGTABLE](#). You can view the status from a separate Oracle Database Server session during maintenance operations. Alternatively, view it in the same session after the index operation has completed.

Information that is stored in this log table is removed just before starting a new `CREATE INDEX` or index maintenance operation. Be sure to view or save the information immediately after performing any of the preceding listed operations.

In addition to logging information to the domain index internal table, Direct also allows you to log status information to an external table. The format of this table is identical to that of the internal table, the table will be created if necessary. To use external table logging add the parameter `LOGTABLE=[user.]tablename` to the `PARAMETERS` clause of `CREATE INDEX` or `ALTER INDEX`, or to the appropriate argument of `MDLAUX.SCANINDEX`, `MDLAUX.UPDATEPENDINGVERSIONS`, etc. If the table already exists, the new status entries will be appended to it unless you also include the parameter `TRUNCATELOGTABLE`.

See the description of [MDLAUX.SCANINDEX](#), for more information on placement of the parameters.

The log table will be created with the following structure:

```
CREATE TABLE tablename (  
  ID          NUMBER,  
  DATTIM,     DATE  
  PARTITION , VARCHAR2(30)  
  MSG,        VARCHAR2(4000))
```

- `ID` is a sequence number that orders entries within a given operation. If you append to the table you will have duplicate `ID` numbers, however the timestamps will differentiate them.
- `DATTIME` is the date and time that the entry was inserted into the table.
- `PARTITION` is the name of the index partition, or `NULL` if the domain index is not partitioned.
- `MSG` is the text of the status information.

You would typically fetch rows ordered by `ID,DATTIM`.

Logging to a File

Direct provides a mechanism to log error messages and other informational messages to a file. Use this mechanism:

- To keep a permanent log of all errors that occur within Direct for a given user account
- To keep a permanent log of most index operations that use Direct for a given user account
- At the request of BIOVIA Customer Support, to log information about Direct operations which might help to resolve a problem

Note: The folder or directory to which the log file is written must allow `WRITE` access to `WORLD` or `Everyone` who might write log files. The log file is owned by the operating system account that owns the Oracle instance.

This logging mechanism operates on a per-Oracle-user-account basis. To enable logging, connect to SQL*Plus as an Oracle user, and execute the procedure [MDLAUX.CREATELOGFILE](#).

When the Oracle user calls `MDLAUX.CREATELOGFILE`, it creates a table named `MDL_DC_LOGGING` in that user account. The table contains one row which specifies the full path name of the log file and a single numeric value that specifies the amount of information to be written to the log file. This numeric value is referred to as the logging level.

Specify a logging level of zero (0) or one (1) only. Any other logging level should be used only at the request of BIOVIA Customer Support. Higher levels negatively affect Direct performance because of the amount of information that is written to the log file.

A logging level of zero (0) logs only errors. A logging level of (1) log errors and status information for index operations such as `CREATE INDEX` and `ALTER INDEX`, among others.

If you do not specify a logging level when you create the table `MDL_DC_LOGGING`, Direct assumes a logging level of 0.

To disable logging:

1. Drop the table `MDL_DC_LOGGING` from the user account, *OR*
2. Execute the procedure `MDLAUX.CREATELOGFILE` with a `NULL file_name` argument.

Generally, Direct displays only the first 4000 bytes of any error messages that it saves in memory. To display the most recent error message, call the function `MDLAUX.ERRORS`. For information on how to call this function, see [Direct Error Messages](#).

Note: The Direct installer writes a log that contains informational messages about its progress. This log is not the same as the Direct log file that is explained in this section.

The function `MDLAUX.CREATELOGFILE` updates the table row, and re-initializes logging. This means that if you disabled logging before calling this function, logging is enabled afterward.

Log levels

Tip: The higher the logging level, the more information Direct writes to the log file.

Log Level	Purpose
0	Log error messages only
1	Log calls to <code>CREATE INDEX</code> , <code>ALTER INDEX</code> , <code>DROP INDEX</code> , <code>RENAME INDEX</code> , and <code>TRUNCATE</code>

Log Level	Purpose
	INDEX, SCAN, and UPDATE. It also logs error messages. Includes status report calls during index creation or rebuilding, either every 100 rows or every 1000 rows, depending on the size of the table.

Create the MDL_DC_LOGGING Table

The MDL_DC_LOGGING table logs information about Direct operations. To use the logging feature of Direct, first create the table MDL_DC_LOGGING. You create the table in the user schema.

To create MDL_DC_LOGGING, call the PL/SQL procedure that follows:

```
MDLAUX.CREATELOGFILE (FILE_NAME [, NUMBER])
```

where:

- FILE_NAME is a variable that represents the name of a file to which logging information can be written. If you specify this argument as NULL, logging is disabled.

Tip: To write Direct information to an existing file, specify the full path name of the file in the first argument.

Verify that the Oracle Database Server extproc process has the required permissions to create and write to the file that you specify.

- NUMBER is a variable that represents the logging level that you want. If you do not specify the second argument, Direct treats it as if you specified the logging level of (0).

Each entry that Direct writes to the table contains the date, a string containing the Oracle Database Server extproc process ID and sequence number of the WRITE operation, and a status message. The ID string takes the form pppppppppp.nnnnnnnnn, where p represents the process ID and n represents the sequence number, for example 000001101.000000002. Sort table output by the ID column.

Change the Logging Level

To change the logging level or the logfile name, the user must:

1. Disconnect from Oracle Database Server.
2. Log in to Oracle Database Server as the same Oracle Database Server user.

Note: You must disconnect from, and then reconnect to Oracle Database Server in order for the changes that you make to be visible to Oracle Database Server. Oracle Database Server only reads the table once per user session.

3. Modify the table as explained in [Modify the Logging Table](#).
4. Access Direct.

Modify the Logging Table

To modify the logging table, issue the SQL statements that follow:

```
SQL> UPDATE MDL_DC_LOGGING SET LOGLEVEL=LOGLEVEL,
2 LOGFILE='/TMP/FILE_NAME.TXT';
SQL> COMMIT;
```

where:

- LOGLEVEL is a variable that represents the level of logging that you set.
- FILE_NAME is a variable that represents the full pathname of the text file to which Direct logs information.

The Oracle Database Server extproc process must be able to write to the file. There are two special keywords which you can include in the file name:

- <HOST>

The value of the COMPUTERNAME environment variable. On Linux, replace this variable with that portion of the hostname, for example, from `uname -n` up to, but not including, the first dot character.

- <PID>

The integer process ID number.

For example, if you specify LOGFILE as `c:\logs\md1log_<host>_<pid>.txt`, and the hostname is ALTAIR and the PID has a value of 1234, the full LOGFILE path name is `c:\logs\md1log_ALTAIR_1234.txt`.

The keywords are not case-sensitive.

Logging and Performance

Logging allows a system administrator or Oracle Database Server database administrator to monitor the progress of any process that Direct uses to accomplish its tasks. It is especially important because the error stack in Direct does not hold more than 4KB of error message data. This is not sufficient for a system administrator to determine which processes might have failed or how to address any problems that Direct might encounter.

However, logging does have an impact on searching speed. The more messages you write to the log file, the greater the impact on performance.

Tip: Administrators who do not want to slow query speed should either turn off logging or set the LOGLEVEL to 0 until more detailed logging is needed.

Direct Error Messages

This section explains how to view the errors that are generated by Direct.

Direct does not display error messages as they are generated. Instead, it saves the error messages in memory. It then calls the Oracle Database Server, which notifies the user that an error has occurred. As a result, when Direct experiences an error, the user sees only the most recent Direct error messages. No more than approximately 900 characters can be displayed by Oracle Database Server.

To see all of the Direct error messages, call the function MDLAUX.ERRORS, as shown:

```
SELECT MDLAUX.ERRORS FROM DUAL;
```

This function retrieves the last 4000 characters of the errors that Direct has saved in memory. This function also clears the error stack.

Call MDLAUX.ERRORS after you run an operation. This allows you to verify that the procedure completed successfully without errors.

Direct also warns the user of potentially problematic situations. For example, if you issue ALTER INDEX with an invalid PARAMETERS value, the statement does not fail. Such failure would leave the index in an

unusable state. Instead, Direct informs Oracle Database Server that the statement concluded successfully and generates an error message showing the invalid parameter.

For a list of the Direct error messages, see the *BIOVIA Direct Error Messages Guide*.

Chapter 9:

Replication

This chapter explains how to implement replication using Direct.

Direct and Replication

Tip: If you plan to use Oracle Database Server Replication at your site, verify that the Oracle Database Server parameter `GLOBAL_NAMES` is set to `TRUE`. Also verify that the Oracle Database Server edition that you are using supports replication functionality.

Note: BIOVIA recommends that you use Oracle Database Server Streams replication technology with Direct.

Direct Replication Scripts

Direct has supplied scripts to manage replication of ISIS/Host and Direct molecule databases. With Direct 2021, you do not need these scripts. Direct 2021 uses native Oracle Database Server functionality to manage replication. After you have indexed the tables in your molecule databases using Direct, you can replicate such tables using Oracle Database Server functionality.

IMPORTANT! To use Oracle Database Server Replication optimally, you need a thorough understanding of how Oracle Database Server performs and manages replication. Consult Oracle Corporation for more information on replication.

Configure Chemical Tables for Replication

When configuring chemical tables for replication:

- The table should have an Oracle Database Server primary key to support replication.
If the column that serves as a key in the chemical table, such as, for example, `CDBREGNO`, is indexed but is not declared as an Oracle Database Server primary key, convert it to a primary key before starting replication. If the table does not contain a unique column or columns that can function as the primary key, add a primary key column. For example, you can add a column with a `DEFAULT SYS_GUID()` value.
- Replicate only the chemical table that contains the structure BLOB column.
Do not attempt to replicate the additional tables that are created by Direct when you create the domain index on that column. Only tables holding user data should be replicated. These include the table containing the actual chemical-structure BLOB column and any tables joined to it in user applications. Do not replicate the index tables that are created by the `CREATE INDEX` command on the BLOB column. For example, in the sample databases that BIOVIA provides, you replicate `SAMPLE2D` and `SAMPLE2D_BOILING_POINT` tables. You do not replicate any table that has the `SAMPLE2D_IX_` prefix.
The domain indexes are maintained separately in each replicated instance.
- Each instance that contains chemical tables requires an installation of Direct.
After creating a replica table or snapshot at each site, issue the appropriate `CREATE INDEX` command at each site or verify that the Oracle Database Server Import utility has properly copied the indexes. You might need to repeat subsequent index-maintenance commands at each site.

Streams Replication

Oracle Database Server Streams technology enables replication as well other Oracle Database Server functionality. BIOVIA recommends using Oracle Database Server Streams technology for replication.

Oracle Database Server Streams mines the database redo logs to capture DML and DDL changes in objects on the source database. The changes are then formatted as LCRs (logical change records) and propagated through the network to target sites. At the target sites, the LCRs are applied to the appropriate objects on the destination database. Filtering rules and propagation rules determine the granularity of the operation and the configuration of network topology respectively.

Oracle Database Server Streams allows you to reconfigure replicated database objects dynamically. You can perform replication and reconfiguration of replicated database objects simultaneously.

Streams Configuration and Administration

The following list of configuration and initialization parameters for Oracle Database Server Streams Replication is not exhaustive. Verify with your Oracle DBA that these parameters are set to at least the values that are shown:

- Any database where changes are captured must run in ARCHIVELOG mode.
Verify that you have allocated sufficient disk space for the log files, particularly if the database is very active.
- Set GLOBAL_NAMES=TRUE.
- Set JOB_QUEUE_PROCESSES to the maximum number of jobs that can run simultaneously plus one.
Each streams propagator runs a job. This value should be no less than 2.
- Set LOG_PARALLELISM=1 for any database that captures events.
- Set OPEN_LINKS to a minimum value of 4.
- Add 10 MB to SHARED_POOL_SIZE for each capture process.
- Set STREAMS_POOL_SIZE to a minimum of 10 MB for each parallel capture or apply process, as recommended by Oracle Corporation.
This parameter allows the SGA memory that is required by Oracle Streams Replication to be allocated from a separate pool instead of being limited to 10 percent of the shared pool.
- When performing Oracle Database Server Streams Replication instantiation, run the Oracle Database Server Export utility with the option OBJECT_CONSISTENT=Y.
Do not permit DDL changes to objects being exported during the Export process.
- Run Import with the option STREAMS_INSTANTIATION=Y.
For example, you might want to specify the location of a Ptable or SALTS file in the ALTER INDEX or CREATE INDEX command.

You can configure Oracle Database Server Streams Replication in either of the following ways:

- Manually by executing functions in the packages such as DBMS_STREAMS_ADM, DBMS_RULES_ADM as explained in your Oracle Database Server documentation, or
- Using the Wizard available in the Oracle Database Server Enterprise Manager Console.

BIOVIA recommends that you perform instantiation of replicated tables manually using the Oracle Database Server Export and Import utilities. This ensures that the domain index is handled correctly during instantiation. Execute MDLAUX.PREPAREINDEXEXPORT to prepare a Direct domain index for efficient transport as explained in [MDLAUX.PREPAREINDEXEXPORT](#).

Note: Snapshot replication requires full index recreation in at least one snapshot copy, possibly at each instantiation site. A full recreation of the domain index is a time-consuming procedure.

If you have configured replication to propagate DDL, you need to issue the `ALTER INDEX` maintenance commands only on the master table. The maintenance commands are propagated to all other sites, where they run asynchronously.

Index maintenance commands that are executed via procedure calls, such as `MDLAUX.UPDATEPENDINGFASTSEARCH`, are not automatically propagated. Issue such commands separately at each site.

Establish conflict resolution procedures if you plan to allow updates to propagate in both directions between replicated tables. BIOVIA recommends that you permit inserts and updates at only one instance and make all other instances read-only. Oracle Corporation refers to this configuration as the *Primary Database Ownership* model. For more information on replication, see your Oracle Database Server documentation.

Limitations on Replication

Enable supplemental logging on primary key columns or designated substitute keys and on any other columns that are needed for replication dependencies such as, for example, in conflict resolution or for rule-based transforms. For more information about these terms, see your Oracle Database Server documentation.

Oracle Database Server cannot capture DML changes in:

- BFILE
- ROWID
- UROWID
- user-defined types such as:
 - object types, including the `NUMRANGE` type defined by Direct
 - REFs
 - Varrays
 - nested tables

Advanced Replication can handle most of these types with the exception of:

- LONG
- LONG RAW
- BFILE
- UROWID

If you are using Advanced Streams Replication, you might encounter problems handling associated overflow segments for the LOB columns in which Direct stores structure data for index-organized tables. Sequence values are local and could vary for individual rows between source and replicas.

Chapter 10:

Command Reference

This chapter explains the functions and procedures that you use administer Direct.

IMPORTANT! When you issue SQL commands, statements that vary only by constants and parameter values are reparsed. By contrast, in PL/SQL, such statements are not reparsed. To prevent SQL statements that vary only by constants and parameter values from being reparsed, set the parameter `CURSOR_SHARING` to `'FORCE'`.

Cartridge Management Functions and Procedures

This section lists the functions and procedures that you use to manage the data cartridge. It also explains the syntax and contains examples of how to use these functions and procedures.

MDLAUX.CARTRIDGESCHEMA

This function returns the name of the schema, or user account, in which Direct was created. This user is the owner of Direct.

Syntax

```
mdlaux.cartridgeschema
```

Usage

```
select mdlaux.cartridgeschema from dual;
```

Return value

A VARCHAR2 string that contains the schema name.

MDLAUX.CREATEBOOSTFILE

This procedure creates a boost file. Boost files can be created for a molecule or reaction domain index at any time. When creating a FASTSEARCH boost file, the index must have Fastsearch enabled. When creating a KEYS boost file, the index must have Keys enabled.

Note: If Fastsearch or Keys is not enabled, MDLAUX.CREATEBOOSTFILE will terminate with an appropriate error message.

Syntax

```
MDLAUX.CREATEBOOSTFILE('index-or-table-name', 'what-to-boost'  
[BOOSTDIR=directory-path]')
```

Parameter	Description
index-or-table-name	Specifies the name of a Direct molecule or reaction domain index, or the name of a table containing exactly one Direct molecule or reaction domain index. This is the index for which boost files are to be created.
what-to-boost	A blank-separated string of one or more of the following keywords: <ul style="list-style-type: none">■ FASTSEARCH – creates one boost file for each row in the indexname_FSIX table, the files are named username_indexname_

Parameter	Description
	<p>fastsearchNNNNN. This enables slightly faster substructure searches.</p> <ul style="list-style-type: none"> ■ KEYS – creates one boost file for each row in the <code>indexname_IKY2</code> or <code>indexname_IRKY</code> table, the files are named <code>username_indexname_keysNNNNN</code>. This enables much faster similarity searches and slightly faster substructure searches when the SSS query contains polymer or data Sgroups. These searches do not use Fastsearch. ■ CTAB – creates two boost files, one named <code>username_indexname.ctab.dat</code> and the other named <code>username_indexname.ctab.idx</code>. This enables faster substructure searches when the SSS query contains polymer or data Sgroups. These searches do not use Fastsearch.
BOOSTDIR=directory-path	Optional. Names the directory in which the boost files will be created. If the BOOSTDIR=directory-path option is not specified, a BOOSTDIR parameter must have been previously set, either as a parameter associated with the index by <code>CREATE INDEX</code> or <code>ALTER INDEX</code> , or as a global cartridge property by <code>MDLAUX.SETPROPERTY</code> .

Usage

```
EXECUTE MDLAUX.CREATEBOOSTFILE('ACD2D_IX', 'FASTSEARCH');
```

In this example, a file named `user_acd2d_ix_fastsearch00001` will be created in the directory that was specified in the BOOSTDIR parameter to `CREATE INDEX` or to the value of the BOOSTDIR property specified in `MDLAUX.SETPROPERTY`.

IMPORTANT! If there is an existing boost file with the same name as a new boost file, it will be deleted first.

MDLAUX.CREATELOGFILE

This procedure creates a logfile in the Oracle server's file system. Additionally it creates the MDL_DC_LOGGING table in the Oracle schema from which it is executed if it does not already exist. If the table exists, this procedure updates the table and reinitializes logging.

Syntax

```
MDLAUX.CREATELOGFILE (FILE_NAME [, LOGLEVEL])
```

Parameter	Description
file_name	<p>A variable that represents the full path name of a file to which logging information can be written. The file name can contain these placeholders:</p> <p><HOST> On Windows, this keyword is replaced by the value of the COMPUTERNAME environment variable. On Linux this variable is replaced by that portion of the hostname up to, but not including, the first dot character.</p> <p><PID> This keyword is replaced with the process ID number.</p>
loglevel	A variable that represents the logging level that you want. If you specify this argument as NULL, or do not specify it, the logging level is set to 0, and only errors are logged. This

Parameter	Description
	<p>is the default in Direct. Permitted values are:</p> <p>0 - Logs error messages</p> <p>1 - Logs calls to CREATE INDEX, ALTER INDEX, SCAN, UPDATE, and other index maintenance operations. Includes status reporting.</p> <p>>1 - Logs informational messages. Use only at the request of BIOVIA Customer Support.</p>

- **Note:** The folder or directory to which the log file is written must allow WRITE access to WORLD or Everyone who might write log files. The log file is owned by the operating system account that owns the Oracle instance.

Usage

```
EXECUTE MDLAUX.CREATELOGFILE ('c:\logs\mdllog_ALTAIR_1234.txt',[,1])
```

Comments

This procedure does not validate the file name argument. If you specify an invalid file name, logging cannot be enabled.

BIOVIA provides the function MDLAUX.CREATELOGFILE as a simple way to create the log file table.

For information on updating the log file, see [Modify the Logging Table](#).

MDLAUX.ERRORS

This function returns a string that contains all informational messages such as errors, warnings, and so forth, that were issued by Direct or its component libraries since the last call to the ERRORS function.

Syntax

```
mdlaux.errors
```

Usage

```
SELECT MDLAUX.ERRORS FROM DUAL;
```

Return Value

A VARCHAR2 string with the first 4000 characters of the informational messages.

Comments

Oracle Database Server presents the user with only as many error messages as can be contained in approximately 900 characters. This function provides users with a way to view the entire error stack. Calling the function clears the stack. Oracle Database Server limits VARCHAR2 to 4000 characters, so Direct only saves the first 4000 characters from any informational messages.

The error stack is a session-specific object. MDLAUX.ERRORS returns the errors for the current session. If the session in which the error occurred has been lost, a subsequent call to MDLAUX.ERRORS starts a new session and returns NULL.

Call this function after any operation when an error occurs. Use it to view or issue warnings and informational messages. For example, a user who issues ALTER INDEX with an invalid PARAMETERS value sees an error message showing the invalid parameter. However, the SQL statement does not fail.

To write error messages permanently to a file, call [MDLAUX.CREATELOGFILE](#).

MDLAUX.GETENVFILE

This procedure fetches the current global environment item and writes the data to a text file. It also returns a value of datatype NUMBER.

Syntax

MDLAUX.GETENVFILE ('ENV_NAME' , 'FILE_NAME')

Parameter	Description
ENV_NAME	<p>A variable that represents the global environment information which you want written to a text file.</p> <p>The environment item can be one of:</p> <ul style="list-style-type: none"> ■ PTABLE = Periodic table ■ SALTS = Salt definitions ■ MOL2DKEYDEFS = Molecule 2D key definitions ■ SGROUPFIELDS = Data Sgroup field name definitions ■ MOLSKEYDEFS = Molecule 2D subset key definitions ■ ISOTOPEDATA = Atomic weights of isotopes ■ HELMMONOMERS = HELM monomer definitions ■ HELMMONOMERMAP = HELM monomer definitions, mapping to SCSR template atoms ■ HELMSCSRTEMPLATES = HELM monomer definitions (SCSR template atoms format)
FILE_NAME	<p>The full path name of the text file that Direct creates, and to which it writes the global environment information. The Oracle Database Server extproc process must have the permissions required to write to this file.</p> <p>If the file already exists, Direct overwrites its contents if the file is writable, and returns (1). For other values that Direct returns, see the table that follows.</p> <ul style="list-style-type: none"> ■ 0 - MDLAUX.GETENVFILE failed. For information about the reason for the failure, call the function MDLAUX.ERRORS ■ 1 - MDLAUX.GETENVFILE was successful and wrote the data to the text file that it created. ■ 2 - MDLAUX.GETENVFILE created an empty text file. If you have never previously set the environment, this function creates an empty file. If you previously set the environment, the text file exists, but its contents are overwritten, if the user that called this function has write permissions on the text file. When this function creates an empty file, it returns (2), not (1).

Example

```
SELECT MDLAUX.GETENVFILE('PTABLE', '/tmp/ptable.dat') FROM DUAL;
```

MDLAUX.GETEXTPROCID

This procedure returns a VARCHAR2 string containing the process ID of the current Oracle Database Server extproc process.

For an example of how to call this function, see MDLAUX.KILLEXTPROC.

MDLAUX.GETPROPERTY

This procedure gets the current value of a global cartridge-level property. This procedure can be run by any Oracle user to see the current global Direct cartridge properties in the current Oracle instance.

Syntax

```
MDLAUX.GETPROPERTY ('property-name') RETURN VARCHAR2
```

Parameter	Description
<i>property_name</i>	<p>Specifies the global property name to one of the following options:</p> <ul style="list-style-type: none"> ■ BOOSTDIR ■ TEMPDIR ■ NTHREADS ■ STEREORESOLUTION ■ FORCEV3000 ■ FORCEV2000 ■ USEDANDT ■ IgnoreChargesInPiSystems ■ IgnoreHigherOrderStereo ■ RingHomologyQueriesOnlyMapTerminalRings ■ InterpretQueryAtomsLiterally ■ InterpretRAtomsLiterally ■ InterpretZAtomsLiterally ■ InterpretXAtomsLiterally ■ IgnoreTerminalPhosphates ■ ForceFingerprintSearch <p>Note: The <i>property-name</i> value is not case sensitive and can be entered in uppercase, lowercase, or mixed case.</p>

Return Value

A text string that contains the value associated with the global property. If there is no value set for this property, the function returns a NULL value.

Comments

For information about the values appropriate for each property name, see [MDLAUX.SETPROPERTY](#).

MDLAUX.INDEXPARAMETERS

This procedure fetches all parameters that are applicable to this index. It returns a VARCHAR2 string.

Syntax

```
SELECT MDLAUX.INDEXPARAMETERS ('TABLE OR INDEX') FROM DUAL;
```

Parameter	Description
TABLE	a variable that represents the name of the molecule table on which you want Direct to obtain information.
INDEX	a variable that represents the name of the index on the molecule table on which you want Direct to obtain information.

Usage

An example of the output for an index which was created with 9.0.2.4 and upgraded to 9.1, then rebuilt so that the internal version was also updated is as follows:

```
SQL> select mdlaux.indexparameters('dir2233') from dual;
MDLAUX.INDEXPARAMETERS('DIR2233')
-----
Molecule index DIR2233IX version 9.1.0.2 (created with 9.0.2.4)
PREALLOCATE=NOSTORAGECHECK
SQL> alter index dir2233ix rebuild;
Index altered.
SQL> select mdlaux.indexparameters('dir2233') from dual;
MDLAUX.INDEXPARAMETERS('DIR2233')
-----
Molecule index DIR2233IX version 9.1.0.2
PREALLOCATE=NOSTORAGECHECK
```

The string that is returned when you call this function contains all the parameters that are applicable to the index that you specified.

MDLAUX.KILLEXTPROC

This procedure kills an Oracle Database Server extproc process. It takes as its sole argument the integer process id of the Oracle Database Server extproc process to kill.

Syntax

```
SELECT MDLAUX.KILLEXTPROC (pid) FROM DUAL;
```

Parameter	Description
pid	The identifier of the process you want to kill.

Before you call either GETEXTPROCID or KILLEXTPROC:

- Start the second session in which you plan to call KILLEXTPROC *before* you start the session in which you run GETEXTPROCID.
When a runaway session needs to be killed, your chances of starting a session on the same node to kill it, especially in a load-balanced environment, are smaller. Thus, you need to start the second process before you start the search.
- Start the second session in which you plan to call KILLEXTPROC on the same node as the session you want to kill.
Otherwise, the KILLEXTPROC call might fail, or might kill a completely unrelated session.
- KILLEXTPROC is not guaranteed to work on an Oracle Database Server RAC system with load balancing enabled, because the sessions might be dispatched to different nodes.

- KILLEXTPROC depends on session affinity to the underlying process at the operating system level, and might not be effective if you have enabled session pooling.

To derive the (pid), run MDLAUX.GETEXTPROCID

Example: `SELECT MDLAUX.GETEXTPROCPID FROM DUAL;`

Note the value that this statement returns. This is the process ID that you supply as an argument when you call the KILLEXTPROC function.

Terminate the query that you want to kill by using a separate thread of the application, or a separate application, to create a new connection to Oracle Database Server and execute the function MDLAUX.KILLEXTPROC. Provide as an argument the process id that you obtained by calling MDLAUX.GETEXTPROCPID in the first session. For example:

Usage

`SELECT MDLAUX.KILLEXTPROC (12345) FROM DUAL;`

Return Value

This procedure returns a VARCHAR2 value such as:

Value	Description
1	The process was successfully killed.
0	The process could not be killed. Verify that you specified the correct value for the argument.

Comments

If the normal Oracle Database Server query terminate process fails to terminate the query, use this procedure to terminate a long-running query. Killing the Oracle Database Server extproc process that is running the query terminates the query.

MDLAUX.LOGTABLE

This procedure allows you to view information from the index log table *indexname*_LOG that logs status information about these commands or procedures:

- CREATE INDEX
- ALTER INDEX
- MDLAUX.SCANINDEX
- MDLAUX.UPDATEPENDINGINVERSIONS
- MDLAUX.RECREATEKEYS
- MDLAUX.UPDATEPENDINGFASTSEARCH
- MDLAUX.RECREATEFASTSEARCH

This table function effectively defines two columns as (DATTIM DATE, MSG VARCHAR2(4000)) using the type LOGENTRY_T.

Syntax

`MDLAUX.LOGTABLE('TABLE OR INDEX', 'FLAG')`

Parameter	Description
TABLE	A variable that represents the name of the molecule table.

Parameter	Description
INDEX	A variable that represents the name of the index on the molecule table.
FLAG	<p>Lets you specify the order in which information should be returned. To specify this argument, use one of the following options:</p> <ul style="list-style-type: none"> ■ ASC Organize information in ascending order, that is, the first row that was entered is the first row returned. ■ DESC Organize information in descending order, that is, the most recently entered row is the first row returned.

Usage

Enclose the table function in the keyword `TABLE()` and call it in the `FROM` clause of a select statement. You can select either or both of `DATTIM` and `MSG`. `DATTIM` specifies the date and time that the message was inserted into the log table, `MSG` is the message text.

```
SELECT TO_CHAR(DATTIM, 'MM/DD/YYYY HH24:MI:SS') "DATE", SUBSTR(MSG,1,40)
"MSG" FROM TABLE(MDLAUX.LOGTABLE('SAMPLE2D_MOL', 'ASC'));
```

This example views the last 10 records that were inserted into the log table:

```
SELECT TO_CHAR(DATTIM, 'MM/DD/YYYY HH24:MI:SS') "DATE", SUBSTR(MSG,1,40) "MSG"
FROM TABLE(MDLAUX.LOGTABLE('SAMPLE2D_MOL', 'DESC')) WHERE ROWNUM <= 10;
```

Status information that results from building or rebuilding of an index is written to the local index log table. To view such status information, open a separate connection to Oracle Database Server and call `MDLAUX.LOGTABLE`. This function queries the local index log table to determine the status of the operation. Entries are written to the local index log table and committed periodically.

MDLAUX.NODROP

Use this function in conjunction with `DROP INDEX [FORCE]` and `CREATE INDEX PARAMETERS ('NOACTION')`. Use *only* when requested by BIOVIA Customer Support.

Syntax

```
SELECT MDLAUX.NODROP FROM DUAL;
DROP INDEX INDEX [FORCE];
CREATE INDEX INDEX ON table (column) INDEXTYPE IS C$DIRECT2021.MXIXMDL
PARAMETERS ('noaction');
```

Return Value

A `VARCHAR2` string informing the user that the function was called. It also signals a subsequent `DROP INDEX` statement to cause Oracle Database Server to drop the index without dropping the underlying molecule structure search data.

MDLAUX.NOSTRUCT

This function is used to register a nostruct into a table.

Syntax

```
insert into mytable (id, ctab) values ('MyID-1', mol(mdlaux.nostruct));
```


Return Value

This function returns a VARCHAR2 string containing the text of a nostructure molfile.

MDLAUX.PENDINGFASTSEARCH

Use this function to determine when to update the Fastsearch index. BIOVIA recommends that you update the Fastsearch index whenever there are 2000 or more rows pending update. Use to determine the number of records pending fastsearch.

Syntax

```
SELECT MDLAUX.PENDINGFASTSEARCH('TABLE OR INDEX') FROM DUAL;
```

Parameter	Description
TABLE	A variable that represents the name of the molecule table.
INDEX	A variable that represents the name of the index on the molecule table.

Return Value

This procedure returns as a NUMBER data type the total number of molecule or reaction records that are pending in the Fastsearch update area. The function returns (0) if there are no records pending Fastsearch update.

Comments

If the value is 2000 or more, update the Fastsearch index as explained in MDLAUX.UPDATEPENDINGFASTSEARCH.

MDLAUX.DELETEDFASTSEARCH

This procedure determines the number of molecules or reactions with pending sequential keys that require inversion.

Syntax

```
SELECT MDLAUX.DELETEDFASTSEARCH('TABLE OR INDEX') FROM DUAL;
```

Parameter	Description
TABLE	A variable that represents the name of the molecule table.
INDEX	A variable that represents the name of the index on the molecule table.

If the value is 100000 or more, recreate the Fastsearch index as explained in MDLAUX.RECREATEFASTSEARCH.

MDLAUX.PREPAREINDEXEXPORT

This procedure is used to prevent the Oracle Database Server Import utility from recreating the entire domain index.

If you are exporting a large table, recreating the index during import is time-consuming. This function creates a table, *indexname_CRCV*, and imports the information in that table to update ROWIDS shared in the domain index. This is much faster than recreating the index.

- Execute this function (select it from DUAL) before you export your molecule or reaction table.

- When exporting the table containing the domain index, be sure to include the new table in the list of tables that you are exporting. The function return value provides the name of this table that must be included. This is shown in the *Usage* example below.

Syntax

```
MDLAUX.PREPAREINDEXEXPORT('TABLE OR INDEX') FROM DUAL;
```

Parameter	Description
TABLE	A variable that represents the name of the molecule table.
INDEX	A variable that represents the name of the index on the molecule table.

The MDLAUX.PREPAREINDEXEXPORT function:

- Verifies that the table containing the domain index contains exactly one primary key column. If the table does not contain exactly one primary key column, the function raises an exception and terminates.
- Creates a table `indexname_CRCV` with two columns, `RID` and `pkcolumnname`. These columns are declared as `VARCHAR2(4000)`. No storage parameters are specified. Default values are used. The variable `indexname` is the domain index name, truncated to 24 characters.
- Populates the table with the `ROWID` and primary key column values from the table that has the molecule domain index on it.

Usage

```
C:\>sqlplus dcsamples/dcsamples
SQL*Plus: Release 11.2.0.1.0 Production on Fri May 31 16:11:38 2013
Copyright (c) 1982, 2010, Oracle. All rights reserved.
```

```
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing
options
```

```
SQL> describe sample2d_ix_crcv;
ERROR:
ORA-04043: object sample2d_ix_crcv does not exist
```

```
SQL> select mdlaux.prepareindexexport('sample2d') from dual;
```

```
MDLAUX.PREPAREINDEXEXPORT('SAMPLE2D')
```

```
-----
Index import information written to: "DCSAMPLES"."SAMPLE2D_IX_CRCV"
Include this table in your export. After export drop this table.
```

```
SQL> describe sample2d_ix_crcv;
Name                                         Null?    Type
-----
RID                                         VARCHA2(4000)
CDBREGNO                                   VARCHA2(4000)
```

```
C:\>expdp dcsamples/dcsamples dumpfile=DCSAMPLES.DMP directory=dpump_dir1
tables=(sample2d,sample2d_ix_crcv)
```

Export: Release 11.2.0.1.0 - Production on Fri May 31 16:12:49 2013
 Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit

Production

With the Partitioning, OLAP, Data Mining and Real Application Testing options

Starting "DCSAMPLES"."SYS_EXPORT_TABLE_01": dcsamples/*****

dumpfile=DCSAMPLES.DMP directory=dpump_dir1 tables=(sample2d,sample2d_ix_crcv)

Estimate in progress using BLOCKS method...

Processing object type TABLE_EXPORT/TABLE/TABLE_DATA

Total estimation using BLOCKS method: 704 KB

Processing object type TABLE_EXPORT/TABLE/TABLE

Processing object type TABLE_EXPORT/TABLE/GRANT/OWNER_GRANT/OBJECT_GRANT

Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/

Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS

Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS

Processing object type TABLE_EXPORT/TABLE/INDEX/TABLE

Processing object type TABLE_EXPORT/TABLE/INDEX/TABLE_DATA

Processing object type TABLE_EXPORT/TABLE/INDEX/DOMAIN_INDEX/INDEX

. . exported "DCSAMPLES"."SAMPLE2D"	192.1 KB	382
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_CRCV"	15.40 KB	382
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_LOG"	6.718 KB	9
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_PROP"	111.9 KB	26
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_CONV"	11.71 KB	382
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_CTAB"	0 KB	0
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_SKY2"	0 KB	0
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_IKY2"	52.00 KB	1
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_FLEX"	11.49 KB	442
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_SGRP"	0 KB	0
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_NEC"	0 KB	0
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_FMLA"	16.41 KB	382
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_FSUP"	0 KB	0
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_FSDL"	0 KB	0
rows		

. . exported "DCSAMPLES"."SAMPLE2D_IX_FSIX"	294.7 KB	1
---	----------	---

```

rows
. . exported "DCSAMPLES"."SAMPLE2D_IX_CCLK"          0 KB          0
rows
Master table "DCSAMPLES"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
*****
Dump file set for DCSAMPLES.SYS_EXPORT_TABLE_01 is:
  C:\DCSAMPLES.DMP
Job "DCSAMPLES"."SYS_EXPORT_TABLE_01" successfully completed at 16:13:22

```

```
C:\>sqlplus dcsamples/
```

```

SQL*Plus: Release 11.2.0.1.0 Production on Fri May 31 16:14:41 2013
Copyright (c) 1982, 2010, Oracle. All rights reserved.

```

```

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing
options

```

```
SQL> drop table sample2d_ix_crcv purge;
```

```
Table dropped.
```

When importing a .dmp file that contains domain index secondary tables and a table named `indexname_CRCV`, Direct uses the `indexname_CRCV` table to update the ROWIDS contained in the domain index. Direct also creates the sequence `indexname_SQNC` if it does not already exist. If the secondary tables do not exist, but the `indexname_CRCV` table does exist, it is ignored and the index is created as with a normal import.

IMPORTANT! If you imported using the Oracle Original Import utility (`imp`), some of the secondary tables contain LOB columns. If the tablespace name that you specify when you export a table is not the same as the tablespace name in which you want to create your molecule table, the Oracle Database Server Import utility might not be able to import these secondary tables. Index creation fails. To work around this problem, list the *contents only* of the .dmp file as explained in your Oracle Database Server documentation. Use the information that is displayed to create empty tables with LOB columns. Then re-import the molecule tables from the .dmp file. This is not a problem with Oracle Data Pump Import because you can remap tablespace names during import.

Secondary tables with LOB columns are:

- `indexname_PROP`
- `indexname_CONV`
- `indexname_CTAB`
- `indexname_IKY2`
- `indexname_IRKY`
- `indexname_FSIX`
- `indexname_FSUP`
- `indexname_FSDL`

Domain index secondary objects might comprise:

- SEQUENCE *indexname*_SQNC
- TABLE *indexname*_PROP
- TABLE *indexname*_LOG
- TABLE *indexname*_CCLK
- TABLE *indexname*_CTAB
- TABLE *indexname*_IKY2
- TABLE *indexname*_IRKY
- TABLE *indexname*_FLEX
- TABLE *indexname*_RFLX
- TABLE *indexname*_SKY2
- TABLE *indexname*_SQNC
- TABLE *indexname*_SRKY
- TABLE *indexname*_FSUP
- TABLE *indexname*_FSDL
- TABLE *indexname*_FSIX

Comments

- The inserts are performed in an autonomous transaction and are committed every 100 records.
- Execute this function before you export the table that contains the domain index. Notify users that they must not perform any operation that might modify the table from the time you call MDLAUX.PREPAREINDEXEXPORT and the time that exp or expdp completes the export process. Set the parameter CONSISTENT=Y to ensure that users can continue to perform any operation that does not modify the table.
- Upon successful execution, the function returns a string containing the name of the table to include in the export:
 Example: MDLAUX.PREPAREINDEXEXPORT('MOLTABLE-----
 -----Index import information written to: "schema"."indexname_
 CRCV"
- Include the table *indexname*_CRCV in your export. After completing the export procedure, drop this table. If you do not, a subsequent export could erroneously export this table, which does not contain any intervening changes to the index.
- Perform the export operation using exp or expdp after running this function.
- The domain index extensions to the Oracle Database Server Export utility verify that the *indexname*_CRCV table exists, and if it does, they trigger export of secondary tables that are associated with the domain index. The secondary tables are listed in the output from the exp command, even though you did not specify that they be exported. This is normal. The secondary tables are all named *indexname*_XXXX where XXXX is a variable.
- An example of an exp session follows, with user input shown in boldface. Note that the secondary tables, such as SAMPLE2D_IX_FSIX for example, are exported after the user pressed RETURN to the final table prompt.

MDLAUX.RECREATEFASTSEARCH

This procedure recreates the molecule or reaction Fastsearch data in a way that is compatible with concurrent insert, update, delete, and search operations. During processing, checkpoints are created at intervals specified by the COMMITCOUNT parameter. During the checkpoint operation, all data in

temporary files are merged together and written in the Fastsearch data table, the identifier for the next record to be added to the Fastsearch data is written to the property table, and all table data is committed.

You may resume this process. If you encounter an error during the recreate procedure, correct the error, and resume the procedure, which continues from the last checkpoint rather than the beginning.

Syntax

```
EXECUTE MDLAUX.RECREATEFASTSEARCH('TABLE OR INDEX' [, 'FLAGS']);
```

Parameter	Description
<i>TABLE</i>	A variable that represents the name of the molecule or reaction table.
<i>INDEX</i>	A variable that represents the name of the index on the molecule or reaction table.
FLAGS	Optional second argument. Allows you to specify one or more of the following strings separated by white space: <ul style="list-style-type: none"> ■ COMMITCOUNT=<i>checkpoint-interval</i> Specifies the checkpoint interval, that is the number of records to be added to the Fastsearch data before a checkpoint occurs. Direct creates a checkpoint after each segment of approximately 3500000 molecule structures or 1000000 reaction structures has been processed. Direct performs a COMMIT at that checkpoint. If an error occurs while Direct is processing the next group of structures, the temporary index is still correct up to the last checkpoint. To continue adding structures from the last checkpoint, call RECREATEFASTSEARCH again. ■ TEMPDIR=<i>directory_name</i> Specifies the directory in which temporary files will be created. This directory should have at least 20 GB of free space. If a TEMPDIR is not specified, the operating system default is used. For example, on Windows, the files are placed in C:\windows\temp, or on Linux, the files are placed in /var/tmp. ■ NORESUME Ignores the previous checkpoint information and starts the Fastsearch creation from the beginning of the molecule table. If NORESUME is not specified, the default is to a NORESUME previous interrupted MDLAUXRECREATEFASTSEARCH operation from the last good checkpoint. ■ LOGTABLE=[user.]table_name Name of an external log table. Information about the index operation will be inserted into this table as well as into the internal log table. The table will be created if it does not exist. Specify either a table name in the current schema or a schema-qualified table name, for example: LOGTABLE=LOGT2017 LOGTABLE=LOGUSER.LOGT2017 ■ TRUNCATELOGTABLE Use with the LOGTABLE parameter. If this parameter is present the external log table will be truncated if it already exists.

Tip: You must be logged in to Oracle Database Server as the owner of the domain index to execute this procedure.

Example

```
EXECUTE MDLAUX.RECREATEFASTSEARCH('MOLTABLE');
```

Comments

- RECREATEFASTSEARCH also truncates the local index log table and writes status information that is generated during processing to the log table. To view such status information, open a separate session and call the function MDLAUX.LOGTABLE while the operation is proceeding. For more information, see MDLAUX.LOGTABLE”.
- During processing, the index is locked. Locking the index prevents other users from performing a concurrent call to UPDATEPENDINGFASTSEARCH or RECREATEFASTSEARCH.
- Users who attempt to call RECREATEFASTSEARCH while another user is running the procedure receive this error:
Example: MDL-0827: Unable to recreate FASTSEARCH while another user is performing maintenance
- This operation is performed within an autonomous transaction. This means that it cannot see any uncommitted changes within your SQL session. If you have modified the table whose index is being updated, be sure to commit any changes prior to running this procedure.
- If MDLAUX.RECREATEFASTSEARCH fails, a subsequent call to MDLAUX.RECREATEFASTSEARCH resumes the operation just past the last checkpoint that was reached before the failure.
- If the operation fails due to a reason such as insufficient tablespace, correct the cause of the failure and resume the operation by calling MDLAUX.RECREATEFASTSEARCH again.
- If the operation fails and there is a checkpoint stored in the molecule domain index, you cannot run either MDLAUX.UPDATEPENDINGFASTSEARCH or ALTER INDEX REBUILD PARAMETERS ('UPDATEPENDINGFASTSEARCH'). You must first finish the RECREATEFASTSEARCH operation by calling MDLAUX.RECREATEFASTSEARCH or issuing ALTER INDEX REBUILD PARAMETERS ('FASTSEARCH').

MDLAUX.RECREATEINDEXES

IMPORTANT! This function cannot be used to upgrade a Direct 6, 7, or 8 table to Direct 2021. You must use the Direct Database Update Utility, `directupgrade`, described in [Upgrading Direct](#).

Note: Starting with Direct 8, MDLAUX.RECREATEINDEXES will not upgrade domain indexes for previous versions of Direct if the previous Direct version is installed and the domain indexes have a status as VALID. To upgrade valid domain indexes belonging to a previous version of Direct, use MDLAUX.UPGRADEINDEXES_PREPARE and MDLAUX.UPGRADEINDEXES_UPGRADE.

This function recreates all invalid domain indexes in the current user schema and returns a value of datatype VARCHAR2. Call this function when you uninstall Direct and then either reinstall the same version of Direct, or install a newer version.

Uninstalling Direct leaves any existing molecule domain indexes in an invalid state with an unknown index type. After a successful reinstall or new installation of Direct, run this function to search out the invalid indexes and quickly recreate them, reusing the underlying searching data.

Syntax

```
SELECT MDLAUX.RECREATEINDEXES FROM DUAL;
```

Note: Each user who has created a table with a domain index must run this function. It only processes indexes that can be upgraded to the current program version. For example, it does not recreate Direct 5.2 domain indexes. The user must enable the schema for the new version of Direct by issuing the following command:
EXECUTE C\$DIRECT2021.MDLAUXOP.SETUP;

This function issues the following commands for each index:

```
DROP INDEX INDEX FORCE;
CREATE INDEX INDEX ON TABLE (COLUMN_NAME
INDEXTYPE IS C$DIRECT2021.MXIXMDL PARAMETERS ('NOACTION');
```

Parameter	Description
<i>TABLE</i>	A variable that represents the name of the table on which the index is being recreated.
<i>INDEX</i>	A variable that represents the name of the index that is being recreated.
<i>COLUMN_NAME</i>	A variable that represents the name of the column on which the index is being recreated.

Return Value

If there are no invalid indexes and no indexes that belong to a previous version of Direct, the function returns NULL. Otherwise, it returns a string containing the index, table and column names which have been processed, for example:

```
SELECT MDLAUX.RECREATEINDEXES FROM DUAL;
RECREATEINDEXES
```

```
-----
Created mol index:SAMPLE2D_IX Table:SAMPLE2D_MOL Column:CTAB
```

If an error occurs during the CREATE INDEX portion, processing terminates and Direct appends the error string from MDLAUX.ERRORS to the resulting output:

```
SQL> select mdlaux.recreateindexes from dual;
RECREATEINDEXES
```

```
-----
Created mol index:SAMPLE2D_MOLIX Table:SAMPLE2D Column:MOL
```

```
*** Creation failed:
```

```
MDL-0333: Domain index creation failed, however Oracle may still create an
index object which is invalid. You should drop this object now:
```

```
        DROP INDEX DAVIDG.SAMPLE2D_MOLIX;
```

```
MDL-0153: Molecule search index creation failed
```

Comments

As part of CREATE INDEX ... PARAMETERS ('NOACTION'), this command ensures that all of the tables which comprise the search aids are marked SECONDARY within Oracle Database Server. Oracle Database Server does not provide a direct mechanism for setting this flag. Thus, the command might need to:

- Rename the old tables with temporary names.
- Create new empty tables using the old names.
- Copy the contents of the old tables to the new, empty tables.
- Drop the old tables.

MDLAUX.RECREATEKEYS

This procedure recreates substructure search and similarity search keys.

Syntax

```
MDLAUX.RECREATEKEYS('TABLE OR INDEX'[, 'FLAGS']);
```


Parameter	Description
<i>TABLE</i>	A variable that represents the name of the table on which the index is being recreated.
<i>INDEX</i>	A variable that represents the name of the index that is being recreated.
FLAGS	<p>Optional. Set to NULL or to one or more of the following strings separated by white space:</p> <ul style="list-style-type: none"> ■ COMMITCOUNT=<i>checkpoint_interval</i> Specifies the frequency of checkpointing when users are adding molecule or reaction structures to the index. Direct creates a checkpoint after each chunk has been processed. If you set this argument to a value larger than the chunk size, Direct defaults to the value that you specified as the chunk size. A large value improves inversion performance, but can cause other processes which are updating, inserting or deleting records to have to wait while all COMMITCOUNT sequential key records are deleted. For more information about the chunk size, see the CREATE INDEX parameters table. If an error occurs as you are processing the next group of structures, the temporary index is still correct up to the most recent checkpoint. Issue another call to RECREATEKEYS to continue adding structures from the most recent checkpoint. ■ NORESUME Forces Direct to fully recreate the index. Previously interrupted rebuild operations are not resumed. Direct resumes previously interrupted operations at the checkpoint prior to the failure of such operations. ■ LOGTABLE=[user .] tab]ename Name of an external log table. Information about the index operation will be inserted into this table as well as into the internal log table. The table will be created if it does not exist. Specify either a table name in the current schema or a schema-qualified table name, for example: LOGTABLE=LOGT2017 LOGTABLE=LOGUSER . LOGT2017 ■ TRUNCATELOGTABLE Use with the LOGTABLE parameter. If this parameter is present the external log table will be truncated if it already exists.

Comments

Tip: You must be logged in to Oracle Database Server as the owner of the domain index to execute this procedure.

This function recreates the keys in a way that is compatible with concurrent searching, insertion, update and deletion. It is resumable. If an error occurs, determine the cause of the error, correct it, and reissue the RECREATEKEYS procedure as explained in MDLAUX . RECREATEKEYS.

This operation truncates the local index log table and writes status information to the log table during processing. Use MDLAUX . LOGTABLE to view the status information in a separate session while the operation is proceeding. See MDLAUX . LOGTABLE for more information.

During this operation, the index is locked against other users performing a concurrent call to RECREATEKEYS or UPDATEPENDINGINVERSIONS.

Tip: This index lock requires EXECUTE permission on the Oracle Database Server SYS . DBMS_LOCK package.

Note: If you have previously installed an older version of Direct, users, or the BIOVIA administrator, might already have these permissions.

Users who try to execute RECREATEKEYS or UPDATEPENDINGINVERSIONS while another user is running either procedure receive an error message like the following example:

Example: MDL-1202: Unable to recreate molecule keys while another user is performing maintenance

Note: The RECREATEKEYS function is performed within an autonomous transaction. Thus, users who are accessing the normal index do not see any changes while RECREATEKEYS is running. If you have modified the table whose index is being recreated, commit any changes prior to running this procedure.

RECREATEKEYS creates new temporary inverted key records.

If MDLAUX.RECREATEKEYS fails, a subsequent call to MDLAUX.RECREATEKEYS resumes the operation just past the last checkpoint that was reached before the failure.

If the operation fails due to a reason such as insufficient tablespace, correct the cause of the failure and resume the operation by calling MDLAUX.RECREATEKEYS again.

If the operation fails and there is a checkpoint stored in the molecule domain index, you cannot run either MDLAUX.UPDATEPENDINGINVERSIONS or ALTER INDEX REBUILD PARAMETERS ('UPDATEPENDINGINVERSIONS'). You must first finish the RECREATEKEYS operation, by calling MDLAUX.RECREATEKEYS or issuing ALTER INDEX REBUILD PARAMETERS ('UPDATEPENDINGINVERSIONS').

MDLAUX.REGENAAMAPS

This procedure runs MDLAUX.AUTOMAP on all or a selected set of reactions in an indexed reaction column. Those reactions which are modified by the automapper are updated with the modified version.

Syntax

```
MDLAUX.REGENAAMAPS('RXN_TABLE OR INDEX'[, 'RXN_COLUMN'[, 'STATUS_TABLE'[, 'FLAGS'[, 'WHERECLAUSE']]]);
```

Parameter	Description
<i>RXN_TABLE</i>	A variable that represents the name of the table that contains the reaction column. It might include the schema as, for example, 'schema.table_name'
<i>INDEX</i>	A variable that represents the name of the is the name of the index on the table that contains the reaction column. You can use the index name instead of the table name when calling this function.
<i>RXN_COLUMN</i>	A variable that represents the name of the reaction column.
<i>STATUS_TABLE</i>	A variable that represents the name of the table that contains information about the status of the operation. It might contain the schema as, for example, 'schema.table_name'. This table must be created with the following column names: <ul style="list-style-type: none"> ■ RID ■ ROWID ■ STATUS NUMBER

Parameter	Description
	<ul style="list-style-type: none"> ■ CHANGES NUMBER CREATE TABLE STATUS_TABLE (RID ROWID, STATUS NUMBER, CHANGES NUMBER); Verify that the table is empty prior to each call to REGENAAMAPS. One row is inserted for each reaction which is processed, so verify that you size the table appropriately. ■ FLAGS specifies the mode to MDLAUX.AUTOMAP. If this argument is set to NULL, the mode 'RegenAlter' is used. For more information about the mode, see MDLAUX.AUTOMAP. ■ WHERECLAUSE can be used to include a WHERE clause to restrict the set of reactions processed. Do not include the word 'WHERE' in the restriction query. If you specify this argument, specify the mode as well. Set the value to NULL if you want the reactions processed in the default 'RegenAlter' mode.

Usage

```
CREATE TABLE STAB (RID ROWID, STATUS NUMBER, CHANGES NUMBER);
EXECUTE MDLAUX.REGENAAMAPS('rxntable', 'rxncol', 'stab');
```

Alternatively, you can issue:

```
Example: EXECUTE MDLAUX.REGENAAMAPS('rxntable', 'rxncol', 'stab', 'RegenAuto',
'idnumber like "XYZ-%"');
```

To verify the reactions that need to be manually examined, issue:

```
SQL> SELECT rxntable.idnumber,stab.status,stab.changes
2> FROM rxntable,stab WHERE rxntable.rowid=stab.rid
3> AND (stab.status <> 1 OR stab.changes > 0);
```

To monitor progress, open another connection to Oracle Database Server and issue:

```
SQL> SELECT COUNT(*) FROM rxntable;
SQL> SELECT COUNT(*) FROM stab;
```

Comments

This procedure loops over all rows, if no query, or a subset of the rows, if a query, in the specified table and fetches the ROWID. For each ROWID, it then fetches the reaction, automaps it with mode 'RegenAlter', updates the reaction, and finally inserts a row into the status table. This is performed in an autonomous transaction. Direct performs a COMMIT after processing each row. Query the status table at any time to determine the number of reactions that have been processed.

If automap fails for any given reaction, or if the automapper made no changes to a reaction, that reaction is not updated. The status table contains a CHANGES value of 0 for any reaction which was not updated, and a STATUS value of 0 or 3 if the automapper failed.

The cursor loop means that only reactions which have been inserted up to the point in time when the procedure was started are processed. Direct does not see any reactions that were inserted while REGENAAMAPS is running. The cursor loop does not lock the table, so other users can delete, insert, and update reactions. Only a row that is currently being automapped is locked for the duration of the automap operation, the subsequent update, and the insertion into the status table. The transaction is committed, and the row lock removed, after the status table insert.

A user might receive "SNAPSHOT TOO OLD" error if too many rows are selected for processing. If this occurs, use the optional query to restrict processing to a subset of the table.

Examine any reaction with a MAP_STATUS value of 4 or greater, or a MARKS_CHANGED value of 1 or greater to determine whether the automapping was correct. You might need to manually map and update these reactions.

For more information about MAP_STATUS and MARKS_CHANGED, see MDLAUX.AUTOMAP in the *BIOVIA Direct Developers Guide* and the *BIOVIA Direct Reference*.

MDLAUX.REMOVEBOOSTFILE

This procedure removes a boost file.

Note: If `mdl aux . removeboostfile` is called but `mdl aux . createboostfile` was not, an error occurs because there is no boost file to remove.

Syntax

MDLAUX.REMOVEBOOSTFILE(' *index-or-table-name* ', ' *what-to-boost* ')

Parameter	Description
<i>index-or-table-name</i>	Specifies the name of a Direct molecule or reaction domain index, or the name of a table containing exactly one Direct molecule or reaction domain index. This is the index from which boost files are to be removed.
<i>what-to-boost</i>	A blank separated string of one or more of the following keywords. These are shown here in upper case, but may be entered in the command in any case: <ul style="list-style-type: none"> ■ FASTSEARCH – deletes all boost files for Fastsearch ■ KEYS – deletes all boost files for inverted molecule or reaction keys ■ CTAB – deletes the two connection table boost files

Usage

```
EXECUTE MDLAUX.REMOVEBOOSTFILE('ACD2D_IX', 'FASTSEARCH');
```

In this example, the file `user_acd2d_ix_fastsearch00001` is deleted.

MDLAUX.SCANINDEX

This procedure validates a molecule or reaction domain index. This procedure takes one mandatory argument and an optional secondary argument.

Syntax

MDLAUX.SCANINDEX(' *TABLE* OR *INDEX* ' [, ' *FLAGS* ']);

Parameter	Description
<i>TABLE</i>	A variable that represents the name of the molecule or reaction table.
<i>INDEX</i>	A variable that represents the name of the index on the molecule or reaction table.

Parameter	Description
FLAGS	<p>Optional. An argument that specifies what to validate. If you set this parameter to NULL, or do not specify it, everything is validated except CTAB. Alternatively, you can set it to one or more of the following strings separated by white space:</p> <ul style="list-style-type: none"> ■ CONV Validates ROWID conversion table only. ■ KEYS Validates conversion table first, then validates sequential and inverted molecule or reaction keys. ■ FLEXMATCH Validates the conversion table first, then validates the molecule or reaction FLEXMATCH hash values. This only validates the index. It does not attempt to map structures to themselves. Use CTAB to map structures against themselves to determine if they match themselves. ■ FASTSEARCH Validates the conversion table first, then validates the Fastsearch index. ■ FORMULA Validates the conversion table first, then validates the formula index. ■ CTAB Validates the conversion table first, then verifies that each molecule or reaction will match itself with an exact match using "MATCH=ALL" FLEXMATCH switches. If a structure fails to match itself, this procedure also tries to match the structure to itself, ignoring stereochemistry and Sgroup data. ■ ROWS=<i>min-max</i> Specifies the rows to scan. A <i>row</i> is actually the value of the internal regno in the domain index for the row, regnos start at 1. If there are no deletions in a table, the maximum regno will be maximum row number. Because regnos are not reused by delete operations, if rows have been deleted, the maximum regno value will be higher than the number of rows in a table. To determine the maximum internal regno value in the domain index, enter: <pre>SELECT MAX(REGNO) FROM indexname_CONV;</pre> Some examples for this parameter are: <pre>ROWS=1-9999999999</pre> <pre>ROWS=100-200</pre> Use this parameter to resume a scan that has failed due to a very bad structure which causes a network disconnect during the scan. Another use is to break up a very long scan (for example, a scan of a very large database) into more manageable pieces. ■ REPORTFIELD=<i>column-name</i> The REPORTFIELD parameter must specify a single column from the molecule or reaction table. If the column does not exist there is no error message, it outputs the rowid value. Prior to version 9, only the rowid appears in output messages, for example: <pre>execute mdlaux.scanindex('mdctest');</pre> Inverted molecule keys do not match at ROWID=AAH8dFAACAAAMH1AAB With the new argument ROWID=value is replaced with column-name='value', for example: <pre>execute mdlaux.scanindex('mdctest', 'reportfield=extreg');</pre> Inverted molecule keys do not match at EXTREG='phcl'

Parameter	Description
	<p>The conversion table is used to convert integer molecule identifiers to molecule table ROWIDs. The conversion table is used to drive most scans, and is automatically validated prior to running a scan.</p> <p>Output from the scan is placed into the local index log table after first truncating the table. For information on selecting rows from this table, see MDLAUX.LOGTABLE. Index errors are written to this table, as are progress messages every <i>n</i> rows, where <i>n</i> is a variable that is set to 100 or 1000, depending on the size of the table. If there are no errors, an example of typical output in the log table is:</p> <pre>SQL> SELECT SUBSTR(MSG,1,60) MSG FROM 2 TABLE(MDLAUX.LOGTABLE('SAMPLE2D_MOL','ASC')); MSG ----- Scanning molecule index SAMPLE2D_IX on table DAVIDG.SAMPLE2D_ MOL Scanning ROWID conversion table ROWID conversion table scan complete, there are 365 rows Validating FASTSEARCH internal structure Scanning node 1000 Scanning node 2000 Scanning node 3000 Scanning node 4000 Scanning node 5000 Scanning node 6000 Scanning node 7000 Scanning node 8000 Scanning node 9000 Scanning node 10000 Scanning node 11000 Scanning node 12000 Scanning node 13000 Scanning node 14000 Scanning node 15000 Scanning node 16000 Scanning node 17000 Scanning node 18000 Scanning node 19000 Scanning node 20000 Scanning node 21000 FASTSEARCH validation complete Scanning SGROUPDATA index Scanning molecule 2D KEYS index Scanning molecule FLEXMATCH hash index Scanning FORMULA index Scanning row 100 Scanning row 200 Scanning row 300 SGROUPDATA scan complete Molecule 2D KEYS scan complete Molecule FLEXMATCH hash index scan complete</pre>

Parameter	Description
	<p>FORMULA index scan complete</p> <p>Error lines include the ROWID, if possible.</p> <ul style="list-style-type: none"> ■ LOGTABLE=[user .] tablename Name of an external log table. Information about the index operation will be inserted into this table as well as into the internal log table. The table will be created if it does not exist. Specify either a table name in the current schema or a schema-qualified table name, for example: LOGTABLE=LOGT2017 LOGTABLE=LOGUSER.LOGT2017 ■ TRUNCATELOGTABLE Use with the LOGTABLE parameter. If this parameter is present the external log table will be truncated if it already exists.

MDLAUX.SETENVFILE

This procedure sets a global chemical environment item to the contents of a file. The environment should contain your site-specific standard periodic table and salt definition files.

Syntax

MDLAUX.SETENVFILE ('ENV_NAME', 'FILE_NAME')

Parameter	Description
ENV_ NAME	<p>One of:</p> <ul style="list-style-type: none"> ■ PTABLE = Periodic table ■ SALTS = Salt definitions ■ MOL2DKEYDEFS = Molecule 2D key definitions Do not change these key definitions. ■ SGROUPFIELDS = Data Sgroup field name definitions ■ RXNCTRKEYDEFS = Reacting center key definitions Do not change these key definitions. ■ MOLSKEYDEFS = Molecule 2D subset key definitions ■ ISOTOPEDATA = Atomic weights of isotopes ■ HELMMONOMERS = HELM monomer definitions ■ HELMMONOMERMAP = HELM monomer definitions, mapping to SCSR template atoms ■ HELMSCSRTEMPLATES = HELM monomer definitions (SCSR template atoms format)
FILE_ NAME	<ul style="list-style-type: none"> ■ NULL = Sets the environment to a standard default. ■ FILE_NAME = The full path name of a text file that contains the values to which you want to set your chemical environment. The Oracle Database Server extproc process must have read permissions on the file. <p>Note: An empty file is treated as a NULL file name argument.</p>

Usage

```
SQL> SELECT MDLAUX.SETENVFILE('PTABLE', '/direct/mfiles/ptable.dat') FROM  
DUAL;  
SQL> SELECT MDLAUX.ERRORS FROM DUAL;
```

Return Value

The function returns 1 if the environment was changed successfully.

Comments

Only the schema that owns Direct, C\$DIRECT2021, has the permissions that are required to run MDLAUX.SETENVFILE.

The MDLAUX.SETENVFILE function copies the name and contents of the file that is specified by the second argument to the global properties table C\$DIRECT2021.MDL_INDEX_PROPERTIES. You do not need to issue a COMMIT statement after this function is called. MDLAUX.SETENVFILE runs as an autonomous transaction and commits the change automatically.

When you load a new environment file, such as the ptable, into the Direct cartridge schema it will only be used when creating new domain indexes. It will not be used with existing domain indexes. The existing domain indexes contain private copies of the environment files that were in the cartridge when they were created or previously rebuilt with the 'environment' option. If you wish to use the updated environment with an existing domain index you must rebuild that domain index using the parameter 'environment'. For example:

```
alter index moltable_molidx rebuild parameters ('environment');
```

ALTER INDEX will rebuild those parts of the domain index that are affected by any changes to the environment files since the time the indexes were created or previously rebuilt with the 'environment' option. If no changes to the index data are required the ALTER INDEX statement will return immediately, it will not rebuild the index.

You must execute this command for each molecule or reaction domain index that you want to use the new environment files.

See [Parameters for ALTER INDEX](#) for more information on rebuilding the domain index.

Note: You can copy CTAB records from one table to another, or create temporary tables dynamically, even when such tables use differing Ptables in their environments.

MDLAUX.SETFLAGS('NODATE')

Molfiles and rxnfiles written by Direct using the MOLFILE and RXNFILE operators contain the current day date. Chime strings, which are compressed molfiles or rxnfiles, also contain the date. The presence of the date can make comparing these files difficult. Direct provides a way to disable the inclusion of the date.

Execute the procedure MDLAUX.SETFLAGS('NODATE') to disable including the date in molfiles and rxnfiles. This is a session setting. It is active for as long as the Oracle session is running. To re-enable the including the date in the same session, execute the following procedure MDLAUX.SETFLAGS('REMOVE=NODATE').

Syntax

From SQL*Plus use the EXECUTE command to execute the procedure:

```
SQL> execute mdlaux.setflags('nodate');
```

From JDBC, ODBC, or OCI, you need to use an anonymous PL/SQL block:


```
begin mdlaux.setflags('nodate'); end;
```

MDLAUX.SETFLAGS('ForceFingerprintSearch')

Execute the procedure MDLAUX.SETFLAGS('ForceFingerprintSearch') to automatically add the 'FINGERPRINT' flag to all SIMILAR and MOLSIM searches in the current user session. It is not possible for the user to execute a traditional similarity search when this flag is set. This is a session setting. It is active for as long as the Oracle session is running. For more information see "Fingerprint Searching" and the "SIMILAR" and "MOLSIM" sections in the *BIOVIA Direct Reference Guide*.

Syntax

From SQL*Plus use the EXECUTE command to execute the procedure:

```
SQL> execute mdlaux.setflags('ForceFingerprintSearch');
```

From JDBC, ODBC, or OCI, you need to use an anonymous PL/SQL block:

```
begin mdlaux.setflags('ForceFingerprintSearch'); end;
```

MDLAUX.SETFLAGS('FORCEV3000')

Execute the procedure MDLAUX.SETFLAGS('FORCEV3000') to force V3000 output in molfiles and rxnfiles. This is a session setting. It is active for as long as the Oracle session is running. To automatically determine whether to output V2000 or V3000 files, execute the following procedure MDLAUX.SETFLAGS('REMOVE=FORCEV3000').

Syntax

From SQL*Plus use the EXECUTE command to execute the procedure:

```
SQL> execute mdlaux.setflags('forcev3000');
```

From JDBC, ODBC, or OCI, you need to use an anonymous PL/SQL block:

```
begin mdlaux.setflags('forcev3000'); end;
```

MDLAUX.SETFLAGS('FORCEV2000')

Execute the procedure MDLAUX.SETFLAGS('FORCEV2000') to force V2000 output in molfiles and rxnfiles.

Note: Structures which require V3000 format will not be written and the output from MOLFILE, MOLCHIME, RXNFILE and RXNCHIME will be NULL. A warning message will be put onto the Direct error stack and can be retrieved with MDLAUX.ERRORS.

This is a session setting. It is active for as long as the Oracle session is running. To automatically determine whether to output V2000 or V2000 files, execute the following procedure MDLAUX.SETFLAGS('REMOVE=FORCEV2000'):

Syntax

From SQL*Plus use the EXECUTE command to execute the procedure:

```
SQL> execute mdlaux.setflags('forcev2000');
```

From JDBC, ODBC, or OCI, you need to use an anonymous PL/SQL block:

```
begin mdlaux.setflags('forcev2000'); end;
```

MDLAUX.SETFLAGS('USEDANDT')

Execute the procedure `MDLAUX.SETFLAGS('USEDANDT')` to force deuterium and tritium hydrogen isotopes to be written as the D and T pseudoatoms in molfiles and rxnfiles. It also causes these to appear as D and T, not H, in image output from the `MOLIMAGE` and `RXNIMAGE` operators. This is a session setting. It is active for as long as the Oracle session is running. To write deuterium and tritium atoms as hydrogen with an isotope property, execute the following procedure `MDLAUX.SETFLAGS('REMOVE=USEDANDT')`.

Syntax

From SQL*Plus use the `EXECUTE` command to execute the procedure:

```
SQL> execute mdlaux.setflags('USEDANDT');
```

From JDBC, ODBC, or OCI, you need to use an anonymous PL/SQL block:

```
begin mdlaux.setflags('USEDANDT'); end;
```

MDLAUX.SETPROPERTY

This procedure sets global, cartridge-level properties.

Note: Only the `C$DIRECT2021` user can execute the `MDLAUX.SETPROPERTY` procedure.

Syntax

`MDLAUX.SETPROPERTY ('property_name', 'property-value-or-NULL-to-remove')`

Parameter	Description
<i>property_name</i>	<p>Specifies the global property name to one of the following options:</p> <ul style="list-style-type: none"> ■ BOOSTDIR ■ TMPDIR ■ NTHREADS ■ STEREORESOLUTION ■ FORCEV3000 ■ FORCEV2000 ■ USEDANDT ■ IgnoreChargesInPiSystems ■ IgnoreHigherOrderStereo ■ RingHomologyQueriesOnlyMapTerminalRings ■ InterpretQueryAtomsLiterally ■ InterpretRAtomsLiterally ■ InterpretZAtomsLiterally ■ InterpretXAtomsLiterally ■ IgnoreTerminalPhosphates ■ ForceFingerprintSearch <p>Note: The <i>property-name</i> value is not case sensitive and can be entered in uppercase, lowercase, or mixed case.</p>

Parameter	Description
<i>property-value</i>	<p>Specifies a text-string value to be associated with the global property. The following values are dependent on the global property names. Values are:</p> <ul style="list-style-type: none"> ■ BOOSTDIR property value must contain the full pathname of a default directory of the boost files. For more information on boost files, see the Direct Boost Files. The Oracle extproc process must be able to create files in this directory. ■ TMPDIR property value must contain the full pathname of a directory that is the default directory that will contain the temporary files used during Fastsearch processing by the MDLAUX.UPDATEPENDINGFASTSEARCH, MDLAUX.RECREATEFASTSEARCH, CREATE INDEX, and ALTER INDEX operations. The Oracle extproc process must be able to create files in this directory. ■ NTHREADS value must contain a string containing an integer from zero to one less than the number of processors available in the computer. This parameter controls the number of threads used during substructure (SSS or RSS) searching. ■ STEREORESOLUTION property value may be either 'NEMA' or 'PPCHEM'. The default and recommended value is 'NEMA'. This property controls what on symmetry within the molecule. The NEMA algorithm is more accurate. The algorithm is used to determine whether stereocenters are valid or not based generates additional information about meso stereocenters used during FLEXMATCH searching. However, the NEMA algorithm can take a long time with certain types of very large structures. The alternate algorithm, PPCHEM, is much faster but is unable to determine the symmetry for some complex molecules, in addition it does not flag meso stereocenters for special matching during FLEXMATCH searches. ■ FORCEV3000 is used to force the MOLFILE and RXNFILE operators to create molfiles and rxnfiles in V3000 format. If the property value is any non-empty string then files will always be written in V3000 format. If the property is not present then a molfile or rxnfile will only written in V3000 format if the structure contains features which are not compatible with the older V2000 file format. ■ FORCEV2000 is used to force the MOLFILE, MOLCHIME, RXNFILE, and RXNCHIME operators to create molfiles and rxnfiles in V2000 format. If the property value is any non-empty string then files will be written in V2000 format. If that is not possible, they will be written as NULL. If the property is not present (for example, if the argument is NULL) then a molfile or rxnfile will be written in V2000 format. However, if the property contains features which are not compatible with the older V2000 file format, a molfile or rxnfile will be written in V3000 format. V2000 file format can not be used for molecules or reactions which contain features that require the newer V3000 format. Writing files for these structures with the FORCEV2000 option will result in a NULL output value from the MOLFILE, MOLCHIME, RXNFILE, and RXNCHIME operators and a descriptive warning will be placed on the error stack. The error stack can be retrieved with MDLAUX.ERRORS. ■ USEDANDT property is used to force deuterium and tritium hydrogen isotopes to be written as the D and T pseudoatoms in molfiles and rxnfiles. It also causes these to appear as D and T, not H, in image output from the MOLIMAGE and RXNIMAGE operators. If the property value is any non-empty string then files and images will contain D and

Parameter	Description
	<p>T pseudoatoms for hydrogen isotopes. If the property is not present then a molfile or rxnfile will contain hydrogen atoms with additional isotope information, and images will show the isotopes as H.</p> <ul style="list-style-type: none"> ■ IgnoreChargesInPiSystems allows the administrator to automatically add this flag to all SSS, FLEXMATCH, RSS and RXNFLEXMATCH searches. If the property value is any non-empty string the IgnoreChargesInPiSystems flag is added to every SSS, FLEXMATCH, RSS and RXNFLEXMATCH search executed by users of the cartridge. For more information on the IgnoreChargesInPiSystems flag, see the FLEXMATCH Switches section in the <i>BIOVIA Chemical Representation Guide</i> and the RSS and SSS sections in the <i>BIOVIA Direct Reference Guide</i>. While any non-NULL value causes the flag to be added, BIOVIA recommends that you use the value <i>true</i>. ■ IgnoreHigherOrderStereo allows the administrator to automatically add this flag to all SSS, FLEXMATCH, RSS and RXNFLEXMATCH searches. If the property value is any non-empty string the IgnoreHigherOrderStereo flag will be added to every SSS, FLEXMATCH, RSS and RXNFLEXMATCH search executed by users of the cartridge. For more information on the IgnoreHigherOrderStereo flag see the Stereochemistry section in the <i>BIOVIA Chemical Representation Guide</i> and the RSS and SSS sections in the <i>BIOVIA Direct Reference Guide</i>. While any non-NULL value causes the flag to be added, BIOVIA recommends that you use the value <i>true</i>. ■ RingHomologyQueriesOnlyMapTerminalRings allows the administrator to automatically add this flag to all SSS and RSS searches. If the property value is any non-empty string the RingHomologyQueriesOnlyMapTerminalRings flag will be added to every SSS and RSS search executed by users of the cartridge. For more information on the RingHomologyQueriesOnlyMapTerminalRings flag see the RSS and SSS sections in the <i>BIOVIA Direct Reference Guide</i>. While any non-NULL value causes the flag to be added, BIOVIA recommends that you use the value <i>true</i>. ■ InterpretQueryAtomsLiterally allows the administrator to automatically add this flag to all SSS and RSS searches. If the property value is any non-empty string the InterpretQueryAtomsLiterally flag will be added to every SSS and RSS search executed by users of the cartridge. For more information on the InterpretQueryAtomsLiterally flag see the RSS and SSS sections in the <i>BIOVIA Direct Reference Guide</i>. While any non-NULL value causes the flag to be added, BIOVIA recommends that you use the value <i>true</i>. ■ InterpretRAtomsLiterally allows the administrator to automatically add this flag to all SSS and RSS searches. If the property value is any non-empty string the InterpretRAtomsLiterally flag will be added to every SSS and RSS search executed by users of the cartridge. For more information on the InterpretRAtomsLiterally flag see the RSS and SSS sections in the <i>BIOVIA Direct Reference Guide</i>. While any non-NULL value causes the flag to be added, BIOVIA recommends that you use the value <i>true</i>. ■ InterpretZAtomsLiterally allows the administrator to automatically add this flag to all SSS and RSS searches. If the property value is any non-empty string the InterpretZAtomsLiterally flag will be added to every SSS and RSS search executed by users of the cartridge. For more information on the InterpretZAtomsLiterally flag see the RSS and SSS sections in the <i>BIOVIA</i>

Parameter	Description
	<p><i>Direct Reference Guide</i>. While any non-NULL value causes the flag to be added, BIOVIA recommends that you use the value <i>true</i>.</p> <ul style="list-style-type: none"> ■ <code>InterpretXAtomsLiterally</code> allows the administrator to automatically add this flag to all SSS and RSS searches. If the property value is any non-empty string the <code>InterpretXAtomsLiterally</code> flag will be added to every SSS and RSS search executed by users of the cartridge. For more information on the <code>InterpretXAtomsLiterally</code> flag see the RSS and SSS sections in the <i>BIOVIA Direct Reference Guide</i>. While any non-NULL value causes the flag to be added, BIOVIA recommends that you use the value <i>true</i>. ■ <code>IgnoreTerminalPhosphates</code> allows the administrator to automatically add this flag to all SSS and FLEXMATCH searches. If the property value is any non-empty string, the <code>IgnoreTerminalPhosphates</code> flag is added to every SSS and FLEXMATCH search executed by users of the cartridge. For more information on the <code>IgnoreTerminalPhosphates</code> flag, see the FLEXMATCH Switches section in the <i>BIOVIA Chemical Representation Guide</i> and the SSS section in the <i>BIOVIA Direct Reference Guide</i>. While any non-NULL value causes the flag to be added, BIOVIA recommends that you use the value <i>true</i>. <p><code>ForceFingerprintSearch</code> allows the administrator to automatically add the 'FINGERPRINT' flag to all SIMILAR and MOLSIM searches. If the property value is any non-empty string, the FINGERPRINT flag is added to every SIMILAR or MOLSIM search executed by users of the cartridge. It is not possible for users to execute a traditional similarity search in this case. While any non-NULL value causes the flag to be added, Dassault Systèmes recommends that you use the value <i>true</i>. For more information see Fingerprint Searching and the "SIMILAR" and "MOLSIM" sections in the <i>BIOVIA Direct Reference Guide</i>.</p> <ul style="list-style-type: none"> ■ <code>NULL</code> removes the global property.

Usage

```
SQL> EXECUTE MDLAUX.SETPROPERTY('BOOSTDIR', '/opt/direct/boostfiles');
SQL> EXECUTE MDLAUX.SETPROPERTY('TMPDIR', '/opt/direct/temp');
SQL> EXECUTE MDLAUX.SETPROPERTY('NTHREADS', '3');
SQL> EXECUTE MDLAUX.SETPROPERTY('BOOSTDIR', NULL);
SQL> EXECUTE MDLAUX.SETPROPERTY('FORCEV3000', 'x')
SQL> EXECUTE MDLAUX.SETPROPERTY('FORCEV2000', 'x')
SQL> EXECUTE MDLAUX.SETPROPERTY('IgnoreHigherOrderStereo', 'true');
```

MDLAUX.SETUP

This procedure sets up the synonyms that are required to use Direct.

Before an Oracle Database Server user can use Direct, the user must execute this procedure. This procedure defines synonyms for the package names and operators that are required to access Direct functionality. For example:

```
EXECUTE C$DIRECT2021.MDLAUXOP.SETUP;
SELECT MDLAUX.VERSION FROM DUAL;
```

IMPORTANT! Before a user can call this procedure, the Oracle database administrator must grant the `CREATE SYNONYM` privilege directly to the user.

If the required CREATE SYNONYM privilege has been granted to a user role rather than granted directly to the user, the user sees this error message:

Example: ORA-20100: MDLI: CREATE SYNONYM privilege required (granted directly, not via role)

Another possible error is:

Example: ORA-20100: MDLI: Object name in use: *<name>*

The name that the user specified to create the synonym is being used to name some other object, for example, a table. The user must rename the *other* object before trying to access Direct.

MDLAUX.SHOWBOOSTFILES

This procedure shows the current boost file status for a domain index. This function returns a string containing the status of each of the three different types of boost file – FASTSEARCH, KEYS, and CTAB.

Syntax

```
MDLAUX.SHOWBOOSTFILES('index-or-table-name')
RETURN VARCHAR2
```

Parameter	Description
<i>index-or-table-name</i>	Specifies the name of a Direct molecule or reaction domain index, or the name of a table containing exactly one Direct molecule or reaction domain index.

Return Value

A VARCHAR2 containing four lines of text.

The first line names the index, the remaining three lines indicate whether or not there is a boost file for FASTSEARCH, KEYS and CTAB. If there is a boost file, its directory is included in the text line.

Usage

```
SELECT MDLAUX.SHOWBOOSTFILES('ACD2D_IX') FROM DUAL;
MDLAUX.SHOWBOOSTFILES('ACD2D_IX')
```

```
-----
Boost files for index "ACD"."ACD2D_IX":
FASTSEARCH: Boost files in directory: c:\boostfiles
KEYS: Boost files in directory: c:\boostfiles
CTAB: No boost files
```

MDLAUX.UPDATEPENDINGFASTSEARCH

This procedure updates molecules or reactions that have pending Fastsearch data.

Syntax

```
EXECUTE MDLAUX.UPDATEPENDINGFASTSEARCH('TABLE OR INDEX' [, 'FLAGS']);
```

Parameter	Description
<i>TABLE</i>	A variable that represents the name of the molecule or reaction table.
<i>INDEX</i>	A variable that represents the name of the index on the molecule or reaction table.

Parameter	Description
FLAGS	<p>Optional. A second argument that lets you specify one or more of the following strings separated by white space:</p> <ul style="list-style-type: none"> ■ COMMITCOUNT=<i>commit/merge-interval</i>: Specifies the commit/merge interval, that is, the number of records pending update that are to be added to the Fastsearch data before the records are merged into the index. The default value is 100000. After each group of updates as specified in COMMITCOUNT is processed, Direct commits the transaction, synchronizing the index with the updates and deletes. ■ TEMPDIR=><i>directory-name</i> Specifies the directory in which temporary files will be created. This directory should have at least 20 GB of free space. If TEMPDIR is not specified, the operating system default is used. For example, on Windows, the files are placed in C:\windows\temp, or on Linux, the files are placed in /var/tmp. ■ LOGTABLE=[user .] tablenam Name of an external log table. Information about the index operation will be inserted into this table as well as into the internal log table. The table will be created if it does not exist. Specify either a table name in the current schema or a schema-qualified table name, for example: LOGTABLE=LOGT2017 LOGTABLE=LOGUSER.LOGT2017 ■ TRUNCATELOGTABLE Use with the LOGTABLE parameter. If this parameter is present the external log table will be truncated if it already exists.

Comments

Tips:

- You must be logged in to Oracle Database Server as the owner of the domain index to execute this procedure.
UPDATEPENDINGFASTSEARCH updates any records that are pending insertion or deletion into the Fastsearch index. Other users can continue to perform searching, insertion, update, and deletion while UPDATEPENDINGFASTSEARCH is running.
- This index lock requires EXECUTE permission on the Oracle Database Server package SYS.DBMS_LOCK.

Note: If you have previously installed an older version of Direct, users or the Direct administrator might already have these permissions.

Users who attempt to call UPDATEPENDINGFASTSEARCH while another user is running the procedure, receive this error:

Example: MDL-0815: Unable to update pending FASTSEARCH while another user is performing maintenance

UPDATEPENDINGFASTSEARCH also returns an error if RECREATEFASTSEARCH was interrupted. RECREATEFASTSEARCH must have completed before you call UPDATEPENDINGFASTSEARCH.

The process continues as long as there are records pending update.

UPDATEPENDINGFASTSEARCH also truncates the local index log table and writes status information that is generated during processing to the log table. To view such status information, open a separate session and call the function MDLAUX.LOGTABLE while the operation is proceeding. For more information, see MDLAUX.LOGTABLE.

This operation is performed within an autonomous transaction. This means that it cannot see any uncommitted changes within your SQL session. If you have modified the table whose index is being updated, be sure to commit any changes prior to running this procedure.

Example: EXECUTE MDLAUX.UPDATEPENDINGFASTSEARCH('moltable', 'commitcount=4000');

For information on how to run this procedure automatically in the background, see Running Routine Index Maintenance Procedures Automatically.

MDLAUX.UPDATEPENDINGINVERSIONS

This procedure updates molecules or reactions with pending sequential keys and fingerprints that require inversion. To call this procedure, issue:

Syntax

```
EXECUTE MDLAUX.UPDATEPENDINGINVERSIONS('TABLE OR INDEX' [, 'FLAGS']);
```

Parameter	Description
<i>TABLE</i>	A variable that represents the name of the molecule or reaction table.
<i>INDEX</i>	A variable that represents the name of the index on the molecule or reaction table.
<i>FLAGS</i>	<ul style="list-style-type: none"> LOGTABLE=[user.] tablename Name of an external log table. Information about the index operation will be inserted into this table as well as into the internal log table. The table will be created if it does not exist. Specify either a table name in the current schema or a schema-qualified table name, for example: LOGTABLE=LOGT2017 LOGTABLE=LOGUSER.LOGT2017 TRUNCATELOGTABLE Use with the LOGTABLE parameter. If this parameter is present the external log table will be truncated if it already exists.

Tip: You must be logged in to Oracle Database Server as the owner of the domain index to execute this procedure.

UPDATEPENDINGINVERSIONS inverts any keys and fingerprints that are pending inversion while still permitting other users to perform concurrent searching, insertion, update and deletion. However, it prevents other users from concurrently running UPDATEPENDINGINVERSIONS or RECREATEKEYS. See the information that follows for an example of error messages that you might encounter.

UPDATEPENDINGINVERSIONS also truncates the local index log table and writes status information that is generated during processing to the log table. To view such status information, open a separate session and call the function MDLAUX.LOGTABLE while the operation is proceeding. for more information, see MDLAUX.LOGTABLE.

Alternatively, you can issue:

Example: ALTER INDEX MOLTABLE_IX PARAMETERS ('updatependinginversions');

However, ALTER INDEX locks the index and prevents other users from registering records or from searching using the index. This might result in very slow search times. ALTER INDEX is not resumable.

When you issue `ALTER INDEX`, Direct begins the process of updating pending inversions with the first record that required updating when you began inverting keys.

If `ALTER INDEX` fails, the domain index is marked `FAILED`. To recover from a `FAILED` update of pending inversions, issue `ALTER INDEX REBUILD PARAMETERS ('KEYS')`. Alternatively, drop and recreate the index.

After each group of insertions or updates is processed, Direct commits the transaction, synchronizing the inverted keys and the sequential keys. During the final processing for each group, the sequential key table is locked. This prevents other users from inserting, updating, or deleting during this process. The lock is held only as long as it takes to look up and delete the group of sequential key records.

A process performing a search must fetch the inverted keys and the sequential keys in two steps. Thus, it is possible, but unlikely that concurrent searchers could see inconsistent data if the inversion procedure commits data in between the two fetches, because the length of time between the two fetches is very small. However, if such a situation occurs, it could cause missed substructure or similarity search hits and incorrect similarity computations.

The process continues as long as there are new records pending inversion. This command does not go back and attempt to process updates that occur while it is processing records. Issue `MDLAUX.PENDINGINVERSIONS` to determine if there are still records pending inversion after the process ends. Rerun `MDLAUX.UPDATEPENDINGINVERSIONS` as required to update such records.

During this operation, the index is locked against other users performing a concurrent call to `UPDATEPENDINGINVERSIONS` or `RECREATEKEYS`.

Tip: This index lock requires `EXECUTE` permission on the Oracle Database Server `SYS.DBMS_LOCK` package.

Note: If you have previously installed an older version of Direct, users, or the BIOVIA administrator, might already have these permissions.

`UPDATEPENDINGINVERSIONS` returns an error when a `RECREATEKEYS` operation is interrupted. `RECREATEKEYS` must be completed before any user can run `UPDATEPENDINGINVERSIONS`.

Users who attempt to execute `UPDATEPENDINGINVERSIONS` while another user is running the procedure receive the error message that follows:

MDL-0409: Unable to invert keys while another user is performing maintenance
Neither `UPDATEPENDINGINVERSIONS` nor `ALTER INDEX [REBUILD] PARAMETERS ('UPDATEPENDINGINVERSIONS')` operates if the temporary index exists. Run one of the commands described in [Recreate Invalid Domain Indexes](#) to drop the temporary index, then reissue either command.

For information on how to run this procedure automatically in the background, see [Running Routine Index Maintenance Procedures Automatically](#).

MDLAUX.UPGRADEINDEXES_PREPARE and MDLAUX.UPGRADEINDEXES_UPGRADE

These two functions are used together to upgrade domain indexes in the current user schema that were created in a previous version of Direct. Call these functions when you install a new version of Direct, and want to upgrade existing molecule and reaction domain indexes to a new version.

Note: You can run `mdlaux.upgradeindexes_prepare` even if the previous cartridge is not installed. For example, if you uninstall Direct 9.5 and then install Direct 2021, you can still upgrade the Direct 9.5 domain indexes.

When a previous domain index is compatible but has the Oracle `indextype` of the previous domain index, `mdlaux.prepareindexes_prepare` and `mdlaux.prepareindexes_upgrade` will upgrade the `indextype` to the current `indextype` and also ensure that secondary tables remain marked as secondary in the Oracle dictionary tables.

The process is very similar to `mdlaux.recreateindexes` described in the `MDLAUX.RECREATEINDEXES` section, but it is broken up into two functions to accommodate different Direct versions that cannot be run in the same process (for example, 32-bit and 64-bit versions of Direct).

Syntax

```
SELECT MDLAUX.UPGRADEINDEXES_PREPARE FROM DUAL;
SELECT MDLAUX.UPGRADEINDEXES_UPGRADE FROM DUAL;
```

Return Value

Both functions return a value of datatype `VARCHAR2`.

IMPORTANT!

Each Oracle user that has created a table with a domain index must run these functions. They only process indexes that can be upgraded to the current program version. For example, the functions will not upgrade Direct 8 domain indexes.

Before running these functions, the user account containing the domain index must first be enabled for the new version of Direct by issuing the following command:

```
SQL> EXECUTE C$DIRECT2021.MDLAUXOP.SETUP
```

`mdlaux.upgradeindexes_prepare` returns a string containing:

- Names of indexes which it has processed and which will be upgraded by `mdlaux.upgradeindexes_upgrade`
- Names of indexes which have already been upgraded to Direct 2021
- Names of indexes which have already been processed by `mdlaux.upgradeindexes_prepare`, but which have not yet been processed by `mdlaux.upgradeindexes_upgrade`
- Names of indexes which cannot be upgraded

`mdlaux.upgradeindexes_upgrade` returns a string containing the names of indexes that is has processed, that is which are now upgraded.

For example:

```
SQL> execute c$direct2021.mdlauxop.setup
PL/SQL procedure successfully completed.
```

```
SQL> select mdlaux.upgradeindexes_prepare from dual;
```

```
UPGRADEINDEXES_PREPARE
```

```
-----
Prepared index for upgrade/recreate: TEST_IX Table: TEST
Index is already upgraded: MOLPROD_IX
Prepared index for upgrade/recreate: RXNTEST_IX Table: RXNTEST
```

```
SQL> select mdlaux.upgradeindexes_upgrade from dual;
```

```
UPGRADEINDEXES_UPGRADE
```

```
-----
Created index: TEST_IX
Created index: RXNTEST_IX
```

MDLAUX.VERSION

This procedure returns a VARCHAR2 string containing the Direct version numbers.

Syntax

```
SELECT MDLAUX.VERSION FROM DUAL;
SELECT MDLAUX.VERSION ('SUBSYS') FROM DUAL;
```

Parameter	Description
SUBSYS	Optional. An argument that prints out the internal version numbers for subsystems within Direct.

Usage

```
SQL> SELECT MDLAUX.VERSION FROM DUAL;
VERSION
```

```
-----
                        BIOVIA Direct
Revision 2021 (Microsoft Windows Oracle12) (21.1.0.x.x)
(c) Copyright Dassault Systèmes, Inc. 1999–2020
```

Comments

- These functions cannot be used to upgrade a Direct 8 table or domain index to Direct 9. You must use the Direct 9 Update Utility, `directupgrade`, described in [Upgrading Direct](#).
- To upgrade a Direct 6, 7, or 8 database, see [Upgrading a Direct 6, 7, or 8 Molecule or Reaction Table](#).
- These functions will only upgrade a domain index when the old index is present in the `USER_INDEXES` view, and all of the secondary tables which are part of the domain index are present. If you have imported a dump file which has a domain index for a version of Direct which is not installed on your system, there will be no index present in the `USER_INDEXES` view. Refer to [Upgrading Indexes](#) for information on how to upgrade this index.
- A Direct domain index that is no more than one major version behind the current release is usually compatible with the current release. Direct 9 is an exception. A Direct 8 domain index cannot be upgraded to Direct 9.

Parameters for Index Creation and Maintenance

This section lists the parameters that you can use when creating and maintaining domain indexes in Direct.

Several INDEX operations take a PARAMETERS string. This string is passed directly to the appropriate routine. The string should contain blank separated KEYWORDS or KEYWORD=VALUE pairs, such as:

Syntax: PARAMETERS ('TABLESPACE USER1 MAXROWS=2000')

or

Syntax: PARAMETERS ('INVERT')

The keyword match is case-independent.

Rules:

- Do not use blanks in keywords and values. If a value must contain a blank, enclose the entire value in double quotes.
- Do not use spaces around the equals sign (=) that separates a keyword from its value.
- Do not use any symbol except the digits 0-9 to represent integral values.

Parameters for CREATE INDEX

This section lists the parameters that you can pass to the CREATE INDEX command. The following table shows the name of each parameter, whether it takes a value, whether it is persistent, and the function of each parameter.

The syntax of a molecule CREATE INDEX command is as follows:

Molecule Syntax: CREATE INDEX INDEX on TABLE (COLUMN_NAME) INDEXTYPE IS C\$DIRECT2021.MXIXMDL [PARAMETERS ('PARAMETERS')];

Molecule example:

```
SQL> CREATE INDEX CORPDB_IX ON CORPDB_MOLTABLE(CTAB)
2> INDEXTYPE IS C$DIRECT2021.MXIXMDL
3> PARAMETERS('TABLESPACE=bigspace');
```

Reaction Syntax: CREATE INDEX INDEX on TABLE (COLUMN_NAME) INDEXTYPE IS C\$DIRECT2021.RXIXMDL [PARAMETERS ('PARAMETERS')];

Reaction example:

```
SQL> CREATE INDEX REACTIONS_IX ON REACTION(RCTAB);
2> INDEXTYPE IS C$DIRECT2021.RXIXMDL
3> PARAMETERS ('TABLESPACE=bigspace');
```

Parameters which take a value allow you to specify information that a procedure uses to perform an action. In this example, the user has specified the TABLESPACE to use when creating a domain index. The value that is passed to the parameter TABLESPACE is the full name of the Oracle tablespace to use when creating secondary objects.

Parameters that are persistent are stored in the index and used by subsequent ALTER INDEX commands. Parameters that are not persistent are only used by the current CREATE INDEX or ALTER INDEX command.

Parameter	Value	Persist	Function
MOLECULES or MOLECULE	No	No	Creates a 2D molecule index. Does not permit the registration of generic (Markush) molecules. If MOLECULES or GENERICS is not specified, the default is MOLECULES. The MOLECULES and GENERICS parameters only apply molecule indexes of the type MXIXMDL. To create a reaction index, use the indextype RXIXMDL.
GENERICS or GENERIC	No		Creates a generic (Markush) molecule index, and allows generic SSS, OVERLAP, and exact-match searches. If MOLECULES or GENERICS is not specified, the default is MOLECULES. The MOLECULES and GENERICS parameters only apply molecule indexes of the type MXIXMDL. To create a reaction index, use the indextype RXIXMDL.

Parameter	Value	Persist	Function
PREALLOCATE	Yes	Yes	<p>Set to NONE, NOSTORAGECHECK, or ALL. If not specified, the default is NOSTORAGECHECK. When set to NONE, no segment allocation is provided to Oracle when creating secondary objects. However, the total amount to space needed by the objects is estimated and compared with the amount of free space available. Direct signals an error if the amount of free space is not sufficient. The value of MAXROWS is used to estimate the required space. When set to NOSTORAGECHECK, no segment allocation is provided to Oracle, and there is no check for sufficient free space. Use this option when there is sufficient space, or when tablespace data files are set to auto-extend. When set to ALL, Direct will provide segment allocation sizes to Oracle when creating secondary objects. The total amount of space needed by the objects is estimated and compared with the amount of free space available. Direct signals an error if the amount of free space is not sufficient. The value of MAXROWS is used to estimate the required space.</p> <div> <p>Note: BIOVIA does not recommend using the value ALL. It is provided only for backward compatibility. Oracle provides efficient space management without using the value ALL.</p> </div>
MAXROWS	Yes	Yes	<p>The maximum number of rows that the molecule table is expected to contain. This parameter is used in conjunction with the PREALLOCATE parameter, and the MAXROWS parameter is ignored if PREALLOCATE is set to NOSTORAGECHECK. See PREALLOCATE for more information. If the MAXROWS parameter is not supplied and PREALLOCATE is either not supplied or set to ALL, or set to NONE then a default maximum is computed as:</p> <ul style="list-style-type: none"> ■ If the table is empty or contains 1000 or fewer rows, Direct uses a value of 1000. ■ If the table is not empty, Direct sets MAXROWS to $(NROWS + NROWS/5)$ where NROWS is the number of rows in the table. <p>Specify the value as a positive integer. A suffix of K multiplies this value by 1024. A suffix of M multiplies this value by (1024^*).</p>

Parameter	Value	Persist	Function
CHUNKSIZE	Yes	Yes	<p>Size of an inverted key chunk in regnos. The maximum is 262144, the minimum is 32. If you do not specify this option, the chunk size is set to:</p> <ul style="list-style-type: none"> ■ If you specified a value for MAXROWS or Direct computed a value for MAXROWS based on a number of rows in the table when the index was created, the CHUNKSIZE is set to the smaller of that value or to 131072. ■ If you did not specify a value for MAXROWS and the table was empty when the index was created, the default CHUNKSIZE is 128000 for a molecule table or 64000 for a reaction table. <p>Otherwise, the value specified or computed when the index was created is used. This value is rounded up to a multiple of 32.</p> <p>Specify the value as a positive integer. A suffix of K multiplies this value by 1024. A suffix of M multiplies this value by (1024*).</p>
TABLESPACE	Yes	Yes	Name of tablespace to use when creating secondary objects that are used in indexing molecules. If you do not specify this value, Direct uses the default tablespace.
TABLE_TABLESPACE	Yes	Yes	Name of tablespace to use when creating secondary tables. If you do not specify this value, Direct uses the TABLESPACE value, and if this value is not set, the default tablespace is used.
INDEX_TABLESPACE	Yes	Yes	Name of tablespace to use when creating indexes on secondary tables. If you do not specify this value, Direct uses the TABLESPACE value, and if this value is not set, the default tablespace is used.
LOB_TABLESPACE	Yes	Yes	Name of tablespace to use when creating inverted key and Fastsearch LOBs. If you do not specify this value, Direct uses the TABLESPACE value, and if this value is not set, the default tablespace is used.
FS_BUFFER_POOL	Yes	Yes	Specify DEFAULT or KEEP or RECYCLE. Specifies the buffer pool to use when caching the Fastsearch index LOB. If you do not specify a value, Direct uses the DEFAULT buffer pool.

Parameter	Value	Persist	Function
UNIQUE	Opt	Yes	<p>Specifies that molecules or reactions must be unique. Specify the argument as the FLEXMATCH or RXNFLEXMATCH switches to be used in the duplicate check, for example 'UNIQUE=match=all'. If this option is not specified, duplicates will be registered.</p> <p>Note: You might need to use a definition of duplicate structure that differs from 'UNIQUE=match=all'. For example, you might want tautomers of a structure to be perceived as duplicates. For information on using FLEXMATCH parameters to define which structures are equivalent, see <i>Registration to Chemical Databases</i> in the <i>BIOVIA Chemical Representation Guide</i>.</p> <p>If you dropped an index that previously had UNIQUE checking turned on and then recreate the index, you can use the NOCHECKUNIQUE parameter to bypass duplicate checking so that index creation will be faster. For example:</p> <pre>CREATE INDEX ON columnname PARAMETERS ('UNIQUE="MATCH=ALL" NOCHECKUNIQUE');</pre> <p>Direct has been enhanced so that NEMA key matching is used if a FLEXMATCH duplicate check search times out. This effectively eliminates cases where a time out occurs.</p> <p>In the unlikely case that NEMA generation times out, the normal behavior during registration is for the structures to be considered duplicates and disallowed during registration.</p> <p>To allow such timeouts to be registered, include the /ALLOW_TIMEOUT flag in the argument to unique. Separate this flag by a space from the FLEXMATCH or RXNFLEXMATCH switches. For example 'UNIQUE="/allow_timeout match=all''. Since the value contains spaces, enclose it in double quotes.</p> <p>This option causes the binary ctabs to be stored in the index. This increases the index table storage requirements by about 1000 to 1500 bytes per molecule or 4000-6000 bytes per reaction. If you have licensed Oracle Advanced Compression, use the COMPRESS parameter to reduce the space required.</p> <p>If you are creating an index on a table which already contains structures, index creation fails if duplicates exist.</p> <div> <p>Note: The UNIQUE parameter cannot be used with a generic molecule index.</p> </div>
COMPRESS			<p>Enables compression of duplicate check data in an index. Compressing reduces the amount of space required to store the duplicate check binary connection tables, and the space reduction is significant.</p> <p>Direct does not automatically apply the Oracle COMPRESS MEDIUM option to the BLOB column in the table that holds duplicate check CTABs.</p>

Parameter	Value	Persist	Function
			<p>Note: The COMPRESS parameter can only be used when duplicate checking is enable within the index, that is, when UNIQUE=YES or UNIQUE="flexmatch switches" parameter is used.</p> <p>BIOVIA recommends that Oracle 11gR2 users who have licensed Oracle Advanced Compression use the COMPRESS option when duplicate checking is enabled.</p>
COMMITCOUNT	Yes	No	<p>Specifies the commit interval, that is, the number of records to be processed before a COMMIT occurs.</p> <p>Note: This parameter only controls how often a COMMIT is done, and is intended to reduce the amount of rollback/undo tablespace required during the operation.</p> <p>This parameter does not affect Fastsearch temporary files. If a value is not specified or specified as zero, the default value is 50000.</p>
MERGECOUNT	Yes	No	<p>Specifies the merge interval, that is, the number of records to be added to the Fastsearch data before the contents of the temporary files are merged together and written to the Fastsearch LOB.</p> <p>The default value is 3500000 for molecules or 1000000 for reactions. If you specify a smaller value, it can reduce the amount of temporary file space required, but will take longer and use more space in the Fastsearch data table.</p>
TEMPDIR	Yes	Yes	<p>Specifies the directory in which temporary files will be created. This directory should have at least 20 GB of free space. If a directory name is not specified, the operating system default is used. For example, on Windows, the files are placed in C:\windows\temp, or on Linux, the files are placed in /var/tmp.</p>
BOOSTDIR	Yes	Yes	<p>Specifies the directory where the boost files will be stored. For more information on where Direct stores boost files, see Storing in Boost Files.</p> <p>Note: Note: There are several ways to specify the directory. For more information, see Location of Boost Files.</p>

Parameter	Value	Persist	Function
DISABLE	Yes	Yes	<p>Specifies that one or more searching aids be disabled. Disabling a search aid, such as the FASTSEARCH index, speeds up registration, but slows searching. Specify one or more of the words that follow, separating them with commas:</p> <ul style="list-style-type: none"> ■ FLEXMATCH ■ KEYS ■ FASTSEARCH ■ FORMULA <p>For example, 'DISABLE=FLEXMATCH,KEYS'.</p> <p>If no option is specified, all search aids are enabled.</p>
ENABLE=NEMAKEY	Yes	Yes	<p>Enabling NEMAKEY allows the use of NEMA keys during FLEXMATCH exact-match searching and duplicate checking during registration.</p> <p>For more information, see Enable NEMA Keys for Exact-match Searching.</p>
LOGTABLE	Yes		<p>Name of an external log table. Information about the index operation will be inserted into this table as well as into the internal log table. The table will be created if it does not exist. Specify either a table name in the current schema or a schema-qualified table name, for example:</p> <p>LOGTABLE=LOGT2017 LOGTABLE=LOGUSER.LOGT2017</p> <p>For more information see the section Logging Information.</p>
SEQUENCE_EXPAND_LIMIT	Yes	Yes	<p>Specifies the size cutoff for SCSR template expansion during registration and searching of biopolymer sequences. Sequences with a size smaller than or the same as the cutoff will be expanded into full chemistry during registration and searching. The cutoff can either be a number of monomer units such as amino acids, or a molecular weight. Specify the number of monomers as a numeric value optionally with the suffix 'monomers', for example:</p> <p>'SEQUENCE_EXPAND_LIMIT=60'</p> <p>'SEQUENCE_EXPAND_LIMIT=60monomers'</p> <p>Specify the molecular weight as a numeric value with the suffix 'amu', for example:</p> <p>'SEQUENCE_EXPAND_LIMIT=6000amu'.</p> <p>Do not include a space between the number and 'monomers' or 'amu'. If this option is not present SCSR template expansion will not occur, the domain index will act the same as it did in Direct 2017 R2 and earlier. For more information see Biopolymer Searching in Direct.</p>
TRUNCATELOGTABLE	No		<p>Use with the LOGTABLE parameter. If this parameter is present the external log table will be truncated if it already exists.</p>

Parameter	Value	Persist	Function
FINGERPRINT	Yes	Yes	Specifies the type and optionally size (number of bits) of a user-defined fingerprint for similarity searching. The type can be 'accord' or one of the Scitegic fingerprint types such as 'fcfp_4' or 'ecfp_6'. Accord fingerprint size is always 384 bits. The number of bits for Scitegic fingerprints can be specified following a comma, the default is 512 bits. For example: FINGERPRINT=accord FINGERPRINT=ecfp_6,512 For more information, see Fingerprint Searching .

Parameters for ALTER INDEX REBUILD

This section explains the parameters that you can pass to the command ALTER INDEX REBUILD. For information on the parameters that you can pass to the ALTER INDEX command when you do not want to rebuild the index or indexes, see [Parameters for ALTER INDEX](#).

Oracle Database Server marks the index as unavailable during the time that the ALTER INDEX commands are running. Users cannot insert, update, or delete rows during this period. Users can continue to perform searches. However, because the index is not used, searches might be very slow.

The syntax of the ALTER INDEX REBUILD command is as follows:

Syntax: ALTER INDEX INDEXNAME REBUILD PARAMETERS ('PARAMETERS');

For more information on parameters that take values, and persistence of parameters, see [Parameters for Index Creation](#)

Parameter	Value	Persist	Function
UPDATEPENDING INVERSIONS or INVERT	No		Inverts pending sequential keys. Run every 1000 - 2000 insertions or updates to the molecule or reaction structure column. If it fails, the index is marked FAILED. Rebuild again, or drop and recreate the index. Call MDLAUX.PENDINGINVERSIONS to determine number of molecules or reactions with pending sequential keys. The keyword INVERT is provided for compatibility with ISIS/Direct Reactions. BIOVIA recommends that you use UPDATEPENDINGINVERSIONS instead. The index is not rebuilt unless you specify a rebuild option or an environment change. BIOVIA recommends that you use MDLAUX.UPDATEPENDINGINVERSIONS instead of ALTER INDEX to invert sequential keys. MDLAUX.UPDATEPENDINGINVERSIONS can be used at the same time that other users are searching and registering.
UPDATEPENDING FASTSEARCH	No		Updates pending Fastsearch records. Run every 1000 - 2000 insertions or updates to the molecule or reaction structure column. If it fails, the index is marked FAILED. Rebuild again, or drop and recreate the index. Returns an error if

Parameter	Value	Persist	Function
			<p>RECREATEFASTSEARCH was interrupted. RECREATEFASTSEARCH must have completed before you call UPDATEPENDINGFASTSEARCH.</p> <p>The index is not rebuilt unless you specify a rebuild option or an environment change.</p> <p>BIOVIA recommends that you use MDLAUX . UPDATEPENDINGFASTSEARCH instead of ALTER INDEX to update pending Fastsearch records. You can use MDLAUX . UPDATEPENDINGFASTSEARCH while other users are searching and registering.</p>
KEYS	No		<p>Rebuilds the molecule or reaction inverted key index, used for substructure and similarity searching.</p> <div> <p>Note: Unlike MDLAUX . RECREATEKEYS, you cannot resume this operation once stopped..</p> </div> <p>BIOVIA recommends that you use MDLAUX . RECREATEKEYS instead of ALTER INDEX to rebuild inverted keys. You can use MDLAUX . RECREATEKEYS while other users are searching and registering and is also resumable in the event of failure.</p>
FLEXMATCH	No		<p>Rebuilds molecule or reaction FLEXMATCH hash values and molecule NEMA keys, used for FLEXMATCH searching.</p>
FASTSEARCH	No		<p>Rebuilds the molecule or reaction Fastsearch index, used for substructure searching.</p> <div> <p>Note: Unlike MDLAUX . RECREATEFASTSEARCH, this operation is not resumable.</p> </div> <p>BIOVIA recommends that you use MDLAUX . RECREATEFASTSEARCH instead of ALTER INDEX to rebuild inverted keys. You can use MDLAUX . RECREATEFASTSEARCH while other users are searching and registering and is also resumable in the event of failure.</p>
FORMULA	No		<p>Rebuild the molecule formula index, used for FMLAMATCH and FMLALIKE searching.</p>

Parameter	Value	Persist	Function
MAXROWS	Yes	Yes	<p>The maximum number of rows that the molecule table is expected to contain. This parameter is used in conjunction with the PREALLOCATE parameter, which was used when the index was created. The MAXROWS parameter is ignored if PREALLOCATE is set to NOSTORAGECHECK. See PREALLOCATE for more information. If the MAXROWS parameter is not supplied and PREALLOCATE is either not supplied or set to ALL or set to NONE then a default maximum is computed as:</p> <ul style="list-style-type: none"> ■ If the table is empty or contains 1000 or fewer rows, Direct uses a value of 1000. ■ If the table is not empty, Direct sets MAXROWS to $(NROWS + NROWS/5)$ where NROWS is the number of rows in the table. <p>Specify the value as a positive integer. A suffix of K will multiply it by 1024, a suffix of M by $(1024*1024)$.</p> <div> <p>Note: Only index items that are being rebuilt have their storage altered. Thus, if only the inverted keys are being rebuilt, only their storage increases (or decreases).</p> </div>
CHUNKSIZE	Yes	Yes	<p>Size of an inverted key chunk in regnos. The maximum is 262144, the minimum is 32. If you do not specify this option, the chunk size is set to:</p> <ul style="list-style-type: none"> ■ If you specified a value for MAXROWS or Direct computed a value for MAXROWS based on a number of rows in the table when the index was created, the CHUNKSIZE is set to the smaller of that value or to 131072. ■ If you did not specify a value for MAXROWS and the table was empty when the index was created, the default CHUNKSIZE is 128000 for a molecule table or 64000 for a reaction table. <p>Otherwise, the value specified or computed when the index was created is used. This value is rounded up to a multiple of 32.</p> <p>Specify the value as a positive integer. A suffix of K multiplies this value by 1024. A suffix of M multiplies this value by $(1024*1024)$. Presence of this parameter triggers a rebuild of keys.</p>

Parameter	Value	Persist	Function
UNIQUE	Yes	Yes	<p>Specifies that molecules or reactions must be unique. If set to NO, specifies that duplicate molecules or reactions are permitted. To turn off duplicate checking, set 'UNIQUE=no'. Specify the argument as the FLEXMATCH or RXNFLEXMATCH switches to be used in the duplicate check, for example 'UNIQUE=match=all'.</p> <div> <p>Note: You might need to use a definition of duplicate structure that differs from 'UNIQUE=match=all'. For example, you might want tautomers of a structure to be perceived as duplicates. For information on using FLEXMATCH parameters to define which structures are equivalent, see <i>Registration to Chemical Databases</i> in the <i>BIOVIA Chemical Representation Guide</i>.</p> </div> <p>If you have dropped an index that previously had UNIQUE checking turned on and then recreate the index, you can use the NOCHECKUNIQUE parameter to bypass duplicate checking so that index creation will be faster. For example: CREATE INDEX indexname ON tablename (columnname) PARAMETERS ('UNIQUE="MATCH=ALL" NOCHECKUNIQUE');</p> <p>Direct has been enhanced so that NEMA key matching is used if a FLEXMATCH duplicate check search times out. In the unlikely case that NEMA generation times out, the normal behavior during registration is for the structures to be considered duplicates and disallowed during registration. To allow such timeouts to be registered, include the /ALLOW_TIMEOUT flag in the argument to unique. Separate this flag by a space from the FLEXMATCH or RXNFLEXMATCH switches. Enclose the value in double quotes. For example 'UNIQUE="/allow_timeout match=all''. Since the value contains a space, enclose it with double quotes. This option causes the binary ctabs to be stored in the index. This increases the index table storage requirements by about 1000 to 1500 bytes per molecule or 4000-6000 bytes per reaction. If you have licensed Oracle Advanced Compression, use the COMPRESS parameter to reduce the space required. If you are specifying a unique constraint, index rebuilding fails if there are any duplicate structures present.</p> <div> <p>Note: The UNIQUE parameter cannot be used with a generic molecule index.</p> </div>
COMPRESS			<p>Enables compression of duplicate check data in an index. Compressing reduces the amount of space required to store the duplicate check binary connection tables, and the space reduction is significant.</p>

Parameter	Value	Persist	Function
			<p>Direct does not automatically applies the Oracle COMPRESS MEDIUM option to the BLOB column in the table that holds duplicate check CTABs.</p> <p>Note: The COMPRESS parameter can only be used when duplicate checking is enable within the index, that is, when UNIQUE=YES or UNIQUE="flexmatch switches" parameter is used.</p> <p>If an index was created without compression, you can use ALTER INDEX REBUILD to enable it, for example: alter index moltable_ix rebuild parameters ('compress');</p> <p>If an index was created with compression and you want to remove the compression feature, use the parameter COMPRESS=NO with ALTER INDEX REBUILD, for example: alter index moltable_ix rebuild parameters ('compress=no');</p> <p>BIOVIA recommends that Oracle 11gR2 users who have licensed Oracle Advanced Compression use the COMPRESS option when duplicate checking is enabled.</p>
COMMITCOUNT	Yes	No	<p>Specifies the commit interval, that is, the number of records to be processed before a COMMIT occurs.</p> <p>Note: This parameter only controls how often a COMMIT is done, and is intended to reduce the amount of rollback/undo tablespace required during the operation.</p> <p>This parameter does not affect Fastsearch temporary files. If a value is not specified (or specified as zero), the default value is 50000.</p>
FS_BUFFER_POOL	Yes	No	<p>Value is one of DEFAULT, KEEP, or RECYCLE. Specifies which buffer pool to use when caching the Fastsearch index LOB. See the description of storage management Parameters for CREATE INDEX.</p>
MERGECOUNT	Yes	No	<p>Specifies the merge interval, that is, the number of records to be added to the Fastsearch data before the contents of the temporary files are merged together and written to the Fastsearch LOB. The default value is 3500000 for molecules or 1000000 for reactions. If you specify a smaller value, it can reduce the amount of temporary file space required, but it takes longer and uses more space in the Fastsearch data table.</p>
TEMPDIR	Yes	No	<p>Specifies the directory in which temporary files are created. This directory should have at least 20 GB of free space. If a directory name is not specified, the operating system default</p>

Parameter	Value	Persist	Function
			<p>is used. For example, on Windows, the files are placed in C:\windows\temp, or on Linux the files are placed in /var/tmp.</p> <p>Notes:</p> <ul style="list-style-type: none"> ■ To remove the current TEMPDIR directory stored in the index and use the default directory, specify TEMPDIR without any argument ('tempdir') or with an empty string as the argument ('tempdir=""). ■ If the only argument is TEMPDIR=directory-name then no rebuild will be done. The command will only set the property value in the index.
BOOSTDIR	Yes	No	<p>Specifies the directory where the boost files will be stored. For more information on where Direct stores boost files, see What can be stored in boost files.</p> <p>Note: There are several ways to specify the directory. For more information, see Location of boost files.</p>
DISABLE	Yes	Yes	<p>Specifies that one or more searching aids be disabled. Disabling a search aid such as the Fastsearch index speeds up registration, but slows searching. Valid values for this keyword are one or more comma-separated words:</p> <ul style="list-style-type: none"> ■ FLEXMATCH ■ NEMAKEY (molecules indexes only) ■ KEYS ■ FASTSEARCH ■ FORMULA <p>For example, 'DISABLE=flexmatch,keys'.</p> <p>Note: Disabling NEMAKEY prevents FLEXMATCH searches from using NEMA keys to exact-match searches. For more information, see Disable NEMA keys for exact-match searching.</p>
ENABLE	Yes	Yes	<p>Specifies that one or more previously disabled searching aids be enabled. Valid values for this keyword are the same as those for DISABLE. Specifying this parameter triggers a rebuild of the search aid or aids being enabled. For example: ENABLE=SSS rebuilds FASTSEARCH and MOL2DKEYS. Enabling NEMAKEY allows the use of NEMA keys during FLEXMATCH exact-match searching and duplicate checking. For more information on enabling NEMA keys, see Enable NEMA keys for exact-match searching.</p>
ENVIRONMENT	No	No	Verifies that the current global chemical environment is

Parameter	Value	Persist	Function
			<p>compatible with the structures stored in the index. If the both are compatible, the local chemical environment is updated so it matches the current global chemical environment.</p> <p>Use this parameter to <code>ALTER INDEX REBUILD</code> when you have made a change to the global chemical environment and want to update the Direct index with that change.</p> <p>The current global chemical environment will not be compatible with the indexed local chemical environment if:</p> <ul style="list-style-type: none"> ■ A pseudoatom symbol present in the local chemical environment was removed from the global chemical environment. ■ A data Sgroup field present in the local chemical environment was removed from the global chemical environment. <p>If the current global chemical environment is compatible, it is copied into the local chemical environment and the index is rebuilt. No checks are made for differences in salt definition or atomic weights.</p> <p>If salt definitions have changed, and if the index was created with the <code>UNIQUE</code> parameter and the <code>SAL FLEXMATCH</code> switch was not specified, verify that your table does not contain any duplicate structures.</p> <p>If atomic weights have changed, and If your table contains a molecular weight column computed with the Direct <code>MOLWT</code> function, recalculate values in the column.</p>
LOGTABLE	Yes		<p>Name of an external log table. Information about the index operation will be inserted into this table as well as into the internal log table. The table will be created if it does not exist. Specify either a table name in the current schema or a schema-qualified table name, for example:</p> <p><code>LOGTABLE=LOGT2017</code> <code>LOGTABLE=LOGUSER.LOGT2017</code></p> <p>For more information see the section Logging Information.</p>
TRUNCATELOGTABLE	No		<p>Use with the LOGTABLE parameter. If this parameter is present the external log table will be truncated if it already exists.</p>

Parameter	Value	Persist	Function
FINGERPRINT	Yes	Yes	<p>Adds fingerprint similarity searching to an existing index without it, or modifies the type of fingerprint similarity search currently in use. Specify the type and optionally size (number of bits) of a user-defined fingerprint for similarity searching. The type can be 'accord' or one of the Scitegic fingerprint types such as 'fcfp_4' or 'ecfp_6'.</p> <p>Accord fingerprint size is always 384 bits. The number of bits for Scitegic fingerprints can be specified following a comma, the default is 512 bits. For example:</p> <p>FINGERPRINT=accord FINGERPRINT=ecfp_6, 512</p> <p>For more information, see Fingerprint Searching.</p>

Parameters for ALTER INDEX

This section explains the parameters that you can pass to the command ALTER INDEX when you do not want to rebuild the index or indexes.

The syntax of the ALTER INDEX command is as follows:

Syntax: ALTER INDEX INDEXNAME_IX PARAMETERS ('PARAMETERS');

For information on the parameters that you can pass to the ALTER INDEX command when you want the index or indexes rebuilt, see [Parameters for ALTER INDEX REBUILD](#). For information about parameters that take values and persistence of parameters, see [Parameters for CREATE INDEX](#).

Parameter	Value	Persist	Function
UPDATEPENDING INVERSIONS or INVERT	No		<p>Inverts pending sequential keys and fingerprints. Run every 1000 - 2000 insertions or updates to the molecule or reaction structure column. If it fails, the index is marked FAILED. Rebuild again, or drop and recreate the index. Call MDLAUX.PENDINGINVERSIONS to determine number of molecules or reactions with pending sequential keys. The keyword INVERT is provided for compatibility with ISIS/Direct Reactions. BIOVIA recommends that you use UPDATEPENDINGINVERSIONS instead. The index is not rebuilt unless you specify a rebuild option or an environment change.</p> <p>BIOVIA recommends that you use MDLAUX.UPDATEPENDINGINVERSIONS instead of ALTER INDEX to invert sequential keys. You can use MDLAUX.UPDATEPENDINGINVERSIONS while other users are searching and registering.</p>
UPDATEPENDING FASTSEARCH	No		<p>Updates pending Fastsearch records. Run every 1000 - 2000 insertions or updates to the molecule or reaction structure column. If it fails, the index is marked FAILED. Rebuild again, or drop and recreate the index. Returns an error if RECREATEFASTSEARCH was interrupted.</p>

Parameter	Value	Persist	Function
			<p>RECREATEFASTSEARCH must run to completion before you call UPDATEPENDINGFASTSEARCH.</p> <p>The index is not rebuilt unless you specify a rebuild option or an environment change.</p> <p>BIOVIA recommends that you use MDLAUX . UPDATEPENDINGFASTSEARCH instead of ALTER INDEX to update pending Fastsearch records.</p> <p>You can use MDLAUX . UPDATEPENDINGFASTSEARCH while other users are searching and registering.</p>
FS_BUFFER_POOL	Yes	Yes	Value is one of DEFAULT, KEEP, or RECYCLE. Specifies which buffer pool to use when caching the Fastsearch index LOB. See the description of storage management, see Parameters for CREATE INDEX.
TEMPDIR	Yes	Yes	<p>Specifies the directory in which temporary files are created. This directory should have at least 20 GB of free space. If a directory name is not specified, the operating system default is used. For example, on Windows, the files are placed in C:\windows\temp, or on Linux, the files are placed in /var/tmp.</p> <div> <p>Note: To remove the current TEMPDIR directory stored in the index and use the default directory, specify TEMPDIR without any argument ('tempdir') or with an empty string as the argument ('tempdir='').</p> </div>
DISABLE	Yes	Yes	<p>Specifies that one or more searching aids be disabled. Disabling a search aid such as the Fastsearch index speeds up registration, but slows searching. Valid values for this keyword are one or more comma-separated words:</p> <ul style="list-style-type: none"> ■ FLEXMATCH ■ NEMAKEY ■ KEYS ■ FASTSEARCH ■ FINGERPRINT ■ FORMULA <p>For example, 'DISABLE=flexmatch,keys'.</p> <div> <p>Note: Disabling NEMAKEY prevents FLEXMATCH searches from using NEMA keys to exact-match searches. For more information, see Disable NEMA Keys for Exact-match Searching.</p> </div>
LOGTABLE	Yes		Name of an external log table. Information about the index operation will be inserted into this table as well as into the

Parameter	Value	Persist	Function
			internal log table. The table will be created if it does not exist. Specify either a table name in the current schema or a schema-qualified table name, for example: LOGTABLE=LOGT2017 LOGTABLE=LOGUSER.LOGT2017 For more information see the section Logging Information .
SEQUENCE_EXPAND_LIMIT	Yes	Yes	Specifies the size cutoff for SCSR template expansion during registration and searching of biopolymer sequences. Sequences with a size smaller than or the same as the cutoff will be expanded into full chemistry during registration and searching. The cutoff can either be a number of monomer units such as amino acids, or a molecular weight. Specify the number of monomers as a numeric value optionally with the suffix 'monomers', for example: 'SEQUENCE_EXPAND_LIMIT=60' 'SEQUENCE_EXPAND_LIMIT=60monomers' Specify the molecular weight as a numeric value with the suffix 'amu', for example: 'SEQUENCE_EXPAND_LIMIT=6000amu'. Do not include a space between the number and 'monomers' or 'amu'. If this option is not present SCSR template expansion will not occur, the domain index will act the same as it did in Direct 2017 R2 and earlier. For more information see Biopolymer Searching in Direct .
TRUNCATELOGTABLE	No		Use with the LOGTABLE parameter. If this parameter is present the external log table will be truncated if it already exists.

Chapter 11:

Performance Tuning

Direct Performance

This section explains how to improve the performance of Direct.

Fastsearch Index Cache and Searching

The Fastsearch index is stored in one or more LOBs within the molecule domain index. Access to the Fastsearch LOB occurs at random locations, reading relatively small amounts of data with each access. The LOB itself is quite large, about 600 MB for each one million molecules.

Reading data from a LOB is slow relative to reading data from other column types in Oracle Database Server, so efficient buffering of the LOB data is essential for providing fast substructure search capacity.

There are two mechanisms you can use for buffering the Fastsearch index LOB.

- **Provided by Oracle Database Server**

This buffering mechanism uses the Oracle Database Server buffer cache. The Fastsearch index LOBs are created with caching enabled. Thus, Oracle Database Server buffers their data using either the DEFAULT buffer cache or, if you so specify, the KEEP buffer cache. See the FS_BUFFER_POOL parameter in [Parameters for CREATE INDEX](#).

This is the preferred mechanism for buffering Fastsearch index LOB data, because it allows all users who are connected to the Oracle Database Server database to share the cache.

If you do not specify the FS_BUFFER_POOL parameter during CREATE INDEX or ALTER INDEX, the Fastsearch index LOBs use the Oracle Database Server DEFAULT buffer cache. The size of the DEFAULT buffer cache is controlled automatically by Oracle Database Server.

- **Alternatively, you can use the database parameter DB_CACHE_SIZE to set the cache size and create your molecule domain index with the FS_BUFFER_POOL set to KEEP.**

The DB_KEEP_CACHE_SIZE value can be determined in the following way. If your table uses a non-standard database block size, set the parameter DB_nK_CACHE_SIZE, where n represents a value of 2, 4, 8, 16, or greater block size.

Structure searching speed might be improved by setting the Oracle Database Server KEEP buffer cache to provide buffering for Fastsearch index data. The default cache is normally used for all object types. Fastsearch data must persist in order to provide search speed improvement. Thus, using the DEFAULT buffer to cache such data might not significantly improve search performance.

Specify an explicit value for the FS_BUFFER_POOL with KEEP tells Oracle Database Server to buffer the Fastsearch index LOBs using the KEEP cache. If the KEEP cache is dedicated to a smaller number of uses, data persists in it for a longer time, and structure searching speed is improved.

Structure search speed is improved by buffering a larger quantity of data. Increase the size of the existing buffer cache by one-quarter the size of the Fastsearch index, if possible, to provide better search performance.

To determine the current size of the Fastsearch LOB data, issue:

```
SQL> SELECT SUM(DBMS_LOB.GETLENGTH(FSINDEX))/1024 "FS KBytes" FROM  
indexname_FSIX;
```

where indexname_FSIX is the name of the Fastsearch index on the table.

IMPORTANT! When you issue SQL commands, statements that vary only by constants and parameter values are reparsed. By contrast, in PL/SQL, such statements are not reparsed. To prevent SQL statements that vary only by constants and parameter values from being reparsed, set the parameter `CURSOR_SHARING` to `'FORCE'`.

Oracle Real Application Clusters

Oracle Real Application Cluster (RAC) software allows you to configure multiple computer systems to access a single Oracle database. An Oracle RAC configuration allows you to create a scalable system with high availability. This means that if one computer in the cluster becomes unavailable, the cluster continues to be available, and therefore, Oracle Database Server continues to be available to users. Read and write activities from different computers are coordinated to the same set of disks. Such coordination is managed by sending messages over a private network between the computer systems.

Consider these points when you set up RAC systems that will use Direct:

- Multiple computer systems that read data from and write data to the same disks increase the importance of caching disk data in the Oracle Database Server System Global Area (SGA). As a result, you might need to tune the `DB_CACHE_SIZE` parameter and other disk caching parameters to ensure optimal performance.
- Coordination activities can bottleneck on the private network between the systems. Registration throughput can be optimized by running registration tasks on one node.

When you implement an Oracle RAC configuration, the SQL `COMMIT` command involves transmitting updated copies of data buffers over the private network to systems that hold obsolete copies of these data buffers. Decreasing the frequency with which your applications issue the `COMMIT` command can improve throughput. However, reducing the frequency of commits can result in error messages such as:

ORA-00060 Deadlock detected

To avoid errors, applications that you use to perform registration must either:

- Issue `COMMIT` statements frequently enough to avoid deadlocks, or
- Rollback failed transactions, pause, and retry the transaction whenever a deadlock occurs.

Segregating registration activities such that most registration to a particular table runs on the same node can reduce the need for coordination messages, thus improving throughput.

Known Issues with Oracle RAC

When you set up an Oracle RAC configuration, the initialization parameter `PARALLEL_ADAPTIVE_MULTI_USER` is set to `TRUE` by default. This setting might result in error messages such as:

ORA-10382 parallel query server interrupt (reset)

To avoid this error, either:

- Set the parameter `PARALLEL_ADAPTIVE_MULTI_USER` to `FALSE`, or
- Ensure that your registration applications commit data frequently enough to not encounter this problem.

For more information, see Oracle Database Server bug 5090822 at Metalink.

Appendix A:

Direct Index Tables

Direct creates secondary tables to implement features of the domain index. The table that follows lists and briefly explains these secondary tables.

Note: The table contents are not intended to be directly addressed by customer applications and might change from release to release. You should use the Direct SQL API (functions and procedures) to access the data contents of these tables.

Description of the Direct Index Tables

Table	Usage	How/when updated
indexname_ PROP	Used to contain the Ptable, Salts, and other definitions specific to that particular domain index.	Generally only set when the domain index is first created.
indexname_ FLEX	Used for Flexmatch searches.	Inserts, updates and deletions to this table are made immediately when a structure is inserted, updated, or deleted.
indexname_ FMLA	Used for formula searches.	Inserts, updates, and deletions to this table are made immediately when a structure is inserted, updated, or deleted.
indexname_ CONV	Used to relate structure table rows and index result rows.	Inserts, updates, and deletions to this table are made immediately when a structure is newly registered, modified or deleted. Additionally, this table is rebuilt during the import of a domain index from an Oracle .dmp file.
indexname_ FSIX	Used for Fastsearch (SSS or RSS) searches.	Updated at discrete points – that is, when the database administrator decides to incorporate all the changes (new structures and new deletions) since the last update.
indexname_ FSUP	Used to store pending inserts to the FS index.	Inserts to this table are made immediately when a structure is inserted or updated. This information is then later used to update the Fastsearch index, and then dropped.
indexname_ IKY2	Used to store the inverted SSS keys and some SSS searches, and these keys are used for similarity searching.	Updated at discrete points – that is, when the database administrator decides to incorporate all the changes (new structures and new deletions) since the last update.
indexname_ SKY2	Used to store the sequential SSS key index until inversion can be performed.	Inserts, updates and deletions to this table

Table	Usage	How/when updated
		are made immediately when a structure is newly registered, modified or deleted. This information is later used to update the inverted key index, and then dropped.
indexname_ LOG	Used to store messages from the most recent index rebuild or scan operation.	The table is written only when the database administrator runs an index create, recreate or scan function.
indexname_ CCLK	Used to prevent very similar structures from being inserted until their uniqueness can be verified. This allows greater registration throughput since dissimilar structures can be registered simultaneously.	Used only while registration is actually occurring.
indexname_ SGRP	Used to store Sgroup data items for use during FLEXMATCH and RXNFLEXMATCH searching.	Inserts, updates, and deletions to this table are made immediately when a structure is newly registered, modified, or deleted.
indexname_ IRKY	Used to store the inverted reaction keys. These keys are used for some RSS searches and for reaction similarity searching.	Updated at discrete points. That is, when the database administrator decides to incorporate all the changes (new reactions and new deletions) since the last update.
indexname_ SRKY	Used to contain the reaction key index until inversion can be performed.	Inserts, updates, and deletions to this table are made immediately when a reaction is newly registered, modified, or deleted. This information is later used to update the inverted key index, and then dropped.
indexname_ RFLX	Used for RXNFLEXMATCH searching.	Inserts, updates, and deletions to this table are made immediately when a reaction is newly registered, modified, or deleted.
indexname_ CTAB	Used to store molecules or reactions for registration duplication checking.	Inserts, updates, and deletions to this table are made immediately when a structure is newly registered, modified, or deleted.
indexname_ NEC	Used for FLEXMATCH searches when NEMAKEY is enabled.	Inserts, updates, and deletions to this table are made immediately when a structure is newly registered, modified, or deleted.
indexname_ GENX	Used for generic molecule exact-match searches.	Inserts, updates, and deletions to this table are made immediately when a structure is newly registered, modified, or deleted.
indexname_ BSQ	Used for biopolymer substructure searching when the query does not contain any modified or cross-linked monomers.	Inserts, updates, and deletions to this table are made immediately when a structure containing biopolymer monomers is newly registered, modified, or deleted.

Appendix B:

Direct SDFFile Import Utility

The DirectSDFFile import utility is designed to read a file of molecule structures and associated data, and to store the records in a simple Oracle table. This is intended as the first step in the process of registering structures and related data to a molecule database. After the data has been imported into Oracle, you can apply Oracle-based tools to validate the data, apply business rules, and perform any other processing needed to register the data into your molecule database.

To use the SDFFile import utility, you:

1. Create the intermediate table into which the data will be read.
2. Import the data.
3. Perform any data validations, apply business rules or other transformations on the data.
4. Register the data to the target molecule database
5. Drop the intermediate table.

In some cases, it might be possible to register directly to the target molecule database. However, this limits your ability to process the data before it is registered. The SDFFile import utility inserts every record and registers the data as provided. Using an intermediate table makes it possible for you to post-process the data in any way that is required before registering it to your molecule database.

Note: If your SDFFile contains non-ASCII characters, before you run `importSDFFile` set the `NLS_LANG` environment variable to a language setting which matches the characters in your SDFFile. Failure to do this may result in Oracle storing the wrong characters.

Syntax of the importSDFFile Utility

Note: `importSDFFile` writes records that could not be registered to a file named `log.sdf`. This file makes it easier to correct and import any failed records.

The syntax of the `importSDFFile` utility is as follows:

```
importSDFFile [-nowait] [-scan ] [ -2d ] [ -fields fieldsfile ] [-errors N ]  
[-reportfield fieldname] inputfile table [ connectstring ]
```

The text in *italic* print indicates variables that are specific to your site, for example:

```
importSDFFile -scan -fields namexlate.txt strange.sdf temptab > table_  
create.sql
```

```
importSDFFile -scan -2d acdupdate.sdf temptab acdsc20081/acdsc20081@myora1  
importSDFFile -fields vendorfieldxlate.txt incoming_mols.sdf molfiletab  
test/test
```

Processing Options of the importSDFFile Utility

Create the Intermediate Table

The `importSDFFile` program can be used to produce the SQL command that is needed to create the intermediate table, for example:


```
importSDFFile -scan unknown.sdf moltab
```

When the `-scan` option is specified but the connection string is not specified, the `importSDFFile` program scans the contents of the first 1000 structural records in the SDFFile to see what data fields are specified in the file. The program uses the maximum string length and the data content of the fields to determine what data type to use.

Data fields that contain only a single integer value are assigned the type `NUMBER(10)`. Data fields that contain a single floating-point number are assigned the type `NUMBER`. Other data fields up to 4000 bytes in length are assigned the data type `VARCHAR2(4000)`. All other fields are assigned the data type `CLOB`. In the absence of the `-2d` switch, the program assumes that the CTAB column is a `MOLFILE` and assigns it a `CLOB` data type. If `-2d` is specified, then the CTAB column is assigned a `BLOB` data type.

The table name parameter supplies the table name to be used in the `CREATE TABLE` command. If the connection string is not specified, the `CREATE TABLE` command displays the table creation SQL. Any `fieldsfile` entries that are needed to adapt the SDFFile field names to names that Oracle can use display as lines of text before the `CREATE TABLE` command.

If the connection string is specified, and the table does not exist, then the table is created and the import proceeds. In this case, the input file might be read twice, but the first pass will stop after 1000 structural records are read. Do not use the `-scan` option with the connect string specified if the table already exists.

You can combine `-scan` with `-2d` and with `-fields`. When you specify the connect string and combine `-scan` and `-2d`, the table is created and the structures are imported as binary CTABs. If you specify the connect string and the `-scan` switch but the table already exists, the program will report an error and will not import the data.

Import as MOLFILE

If you do not specify the `-2d` switch, the CTAB data type is presumed to be `MOLFILE`. This is useful in cases when you want to preserve the input molfile itself, or when you want to process the structure with tools like BIOVIA Pipeline Pilot Server before registering the structure to your molecule database.

Import as 2D

If you specify the `-2d` switch, the CTAB data type is presumed to be `BLOB`. The structure is processed with the Direct MOL operator, which turns the input structure into a binary CTAB.

Note: The Direct cartridge must be installed for binary CTABs to be registered.

Specify a Table in Another Schema

You can create and/or import records into a table in another schema by specifying the `table` argument as `schema.table`. You can also specify a local synonym name as the `table` argument. The current connection, as specified in the connection string, must be able to create and/or insert records into the table in the other schema.

Oracle O/S Authentication

To connect to Oracle using O/S authentication, omit the username and password from the connect string.

The `"/` character is a *required* part of the syntax. For example:

```
importSDFFile -2d myfile.sdf moltab /
```

```
importSDFFile -2d myfile.sdf moltab /@dbalias
```

Rename SDFFile Fields

A file of field names can be used to specify Oracle column names that are different from the names as they appear in the SDFFile. Supply the fields file using the `-fields` switch `-fields fieldsfile`, for example:

```
importSDFFile -2d -fields renames.txt inputfile.sdf moltab test/test
```

The fields file format consists of one field name pair per line, where the first name on the line is the Oracle column name and the second name on the line is the field name as it appears in the SDFFile. Either name can be enclosed in double-quotation marks to accommodate spaces and special characters.

The special names:

- *structure designates the CTAB column if the table uses a column name other than CTAB.

- *molname designates a column that will contain the molecule name stored in each molfile in the SDFFile. If there is no molecule name, the value is NULL.

- *recordnumber designates a column that will contain the one-base record number of each molecule record in the SDFFile. The first record will write a value of 1, the second record a value of 2, and so on.

For example:

```
MOLFILE      *structure
TWO_THIRDS    "2/3-rds"
```

NOWAIT Option

After each record is inserted, an Oracle COMMIT is performed to complete the transaction. However, to maximize import speed, you can use the `-nowait` switch to instruct the `importSDFFile` program to use the Oracle COMMIT WRITE NOWAIT BATCH command instead of an ordinary COMMIT. Consult your Oracle documentation for more information about the risks and benefits of using COMMIT WRITE NOWAIT BATCH.

ERRORS N Option

`importSDFFile` checks the Direct error stack for messages after every record. This decreases performance somewhat, but it provides the complete error stack if an error occurs. Increase performance by specifying an argument to the `-errors N` option that is larger than 1. For example, `-errors 10` will cause the Direct error stack to be checked every 10 records. In rare cases, increasing the error stack might cause messages to be truncated or cause missing messages.

REPORTFIELD Option

Use the REPORTFIELD option to specify the name of a field whose value should be included in any warning or error messages. If this option is not specified only the record number is included with warning and error messages. For example, if the REPORTFIELD option is not specified you might see a warning such as:

```
Error on record 3
MDL-2012: Warning: Removed invalid atom stereo center(s)
```

Using the option `-reportfield cdbregno` would instead show the warning as:

```
Error on record 3, cdbregno=27
MDL-2012: warning: Removed invalid atom stereo center(s)
```

Numeric and Date Field Processing

The `importSDFile` program passes all data fields to Oracle in character form. Default Oracle processing rules for numeric and date fields apply.

If you import numeric values into table columns whose data type is specified as `NUMBER`, Oracle converts the values using the `NLS_NUMERIC_CHARACTERS` settings of your session. If the decimal separator does not match the setting, the conversion can fail. You might want to import numeric values into `CHAR` or `VARCHAR2` columns, and convert the values with your own application after the import.

If you import date values into table columns whose data type is specified as `DATE`, Oracle converts the values using the `NLS_DATE_FORMAT` settings your session. If the input date strings do not match the setting, the conversion can fail. You might want to import date values into `CHAR` or `VARCHAR2` columns, and convert the values with your own application after the import.

Range-valued fields must be imported as character values and interpreted by your application.

Error Handling

In most cases, the `importSDFile` program stops importing if it encounters an Oracle error. The following error codes are suppressed and ignored:

```
ora-00100 no data found
ora-01403 no data found
ora-12652 string truncated
ora-24347 warning of a NULL column in an aggregate function
```

The following error codes are reported, and the current record transaction is rolled back, but the import continues. These error codes represent common conditions, such as duplicate records, formatting errors and constraint violations.

```
ora-00001 unique constraint (schema.constraintname) violated
ora-00054 resource busy and acquire with NOWAIT specified
ora-00060 deadlock detected while waiting for resource
ora-00910 specified length too long for its datatype
ora-00912 input parameter too long
ora-00972 identifier is too long
ora-01400 cannot insert NULL into (column)
ora-01426 numeric overflow
ora-01555 snapshot too old
ora-01704 string literal too long
ora-01722 invalid number
ora-01724 floating point precision is out of range
ora-01841 (full) year must be between -4713 and +9999, and not be 0
ora-01854 julian date must be between 1 and 5373484
ora-01858 a non-numeric character was found where a numeric was expected
ora-01859 a non-alphabetic character was found where an alphabetic was expected
ora-01861 literal does not match format string
ora-01862 the numeric value does not match the length of the format item
ora-01863 the year is not supported for the current calendar
ora-01864 the date is out of range for the current calendar
ora-01865 not a valid era
ora-01866 the datetime class is invalid
ora-01867 the interval is invalid
ora-01868 the leading precision of the interval is too small
ora-01870 the intervals or datetimes are not mutually comparable
```

ora-01871 the number of seconds must be less than 60
ora-01873 the leading precision of the interval is too small
ora-01874 time zone hour must be between -12 and 14
ora-01875 time zone minute must be between -59 and 59
ora-01876 year must be at least -4713
ora-01878 specified field not found in datetime or interval
ora-01879 the hh25 field must be between 0 and 24
ora-01880 the fractional seconds must be between 0 and 999999999
ora-01881 timezone region id is invalid
ora-02290 check constraint violated
ora-02291 integrity constraint
ora-20100 errors or warnings from BIOVIA Direct
ora-29875 error during ODICIndexIndexInsert

The Direct error stack displays after any transaction is rolled back, or once for every record that is read.

The Direct error stack cannot be displayed unless the target schema has the Direct synonyms defined and has rights to run the Direct cartridge. If the Direct cartridge is not installed, the `importSDFFile` program is not be able to register binary CTABs. In this case, the program is only capable of importing molfiles or displaying the table creation SQL.

SDFFile Import Example

The following example creates the table `NEWMOLS` in the `ISIS` schema on the database identified by the Oracle database alias `F11R2A` and loads 3 molecules into the table.

The local machine is a Red Hat Linux system. The remote system could be any system where Direct is installed. The local system has to have Direct installed, even if it is a files-only installation.

```
cat tiny.sdf
```

```
REACCS8208099310382D 1 0.00371 0.00000 0

1 0 0 0 0 0 1 v2000
0.0000 1.5400 0.0000 c 0 0 0 0 0 0
M END
> <REGNO>
1

> <MOLID>
Carbon

$$$$

-ISIS- 10299809172D

2 1 0 0 0 0 0 0 0 0999 v2000
4.7583 -3.0708 0.0000 K 0 0 0 0 0 0 0 0 0 0
5.4728 -2.6583 0.0000 S 0 0 0 0 0 0 0 0 0 0
1 2 1 0 0 0 0
M END
> <REGNO>
2
```

> <MOLID>

K-S

\$\$\$\$

REACCS8208099310382D 1 0.00371 0.00000 0

1 0 0 0 0 0 1 v2000

0.0000 1.5400 0.0000 Fe 0 0 0 0 0 0

M END

> <REGNO>

3

> <MOLID>

Iron

\$\$\$\$

\$ importSDFFile -scan -2d tiny.sdf newmols direct/direct@f11r2a

BIOVIA Direct

Revision 2021 (Linux 64-bit Oracle12) (21.1.0.x.x)

(c) Copyright Dassault Systèmes, 1999-2019

Read 3 records in 0.464 seconds

\$ sqlplus direct/direct@f11r2a

SQL*Plus: Release 11.2.0.1.0 Production on Mon Jun 24 10:43:00 2013
Copyright (c) 1982, 2010, Oracle. All rights reserved.

Connected to:

Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing
options

SQL>

MOLID

Carbon

K-S

Iron

SQL>

MOLFMLA(CTAB)

C H4

H K S

Fe

SQL>

Appendix C:

Direct RFile Import Utilities

The Extended RFile Import Utility

The Direct extended import utility is designed to read a file of molecule or reaction structures and associated data, and to store the records in one or more Oracle tables. This is intended as the first step in the process of registering structures and related data to a molecule or reaction database. After the data has been imported into Oracle, you can apply Oracle-based tools to validate the data, apply business rules, and perform any other processing needed to register the data into your molecule or reaction database.

Direct provides two utilities for importing RFiles:

- The Extended Rfile Import Utility - (`importRFileEx`) - customizable and can read RFiles generated by any external program.
- The [Rfile Import Utility](#) (`importRFile`) - reads REACCS and ISIS RFiles.

If you need further assistance, contact BIOVIA consulting to help.

To use the RFile import utility, you:

1. Scan the input file or a small representative example if there are multiple input files.
2. Create the intermediate tables into which the data will be read.
3. Load the data.
4. Perform any data validations, and apply business rules or other transformations on the data.
5. Register the data to the target molecule or reaction database.
6. Drop the intermediate tables.

Note: If your RFile contains non-ASCII characters, before you run `importRFileEx` set the `NLS_LANG` environment variable to a language setting which matches the characters in your RFile. Failure to do this may result in Oracle storing the wrong characters. Because of limitations in the character handling in `importRFileEx`, non-ASCII characters which occur in field names will be replaced with underscores.

Extended RFile Import Utility Operation

The *extended* RFile import utility, `importRFileEx`, is able to perform the scan, table creation, and load steps in one pass. It reads RFiles created with Pipeline Pilot or external programs. `importRFile` reads RFiles created by REACCS.

`importRFileEx` uses a table definition file to correlate RFile data field names with Oracle table and column names and defines the primary/foreign key relationships between the tables in the hierarchy. The user can provide a table definition file as an input argument or create one from an RFile using the `SCAN` option to `importRFileEx`.

Typical usage would be to scan the RFile to create a table definition file, edit the table definition file if desired to change table or column names, then load the RFile using the edited table definition file as input. The load operation can optionally create the tables in Oracle.

See also

[Tables and Columns](#)[Using importRDFFileEx](#)

Tables and Columns

The complete list of program arguments is shown in the following table. Not all of these are required, it depends on whether `importRDFFileEx` is being used to scan an rdf file, load an RDFFile, create tables or a combination of these.

RDFFILE=<i>filename</i>	Name of the input RDFFile
CONNSTR= <i>Oracle-connection-string</i>	Oracle connection string, for example: user/password user/password@db
LOGFILE= <i>filename</i>	Name of log file, all output is written to this file as well as to the terminal.
SCAN[= <i>number-of-records-to-scan</i>]	Scans the input RDFFile and optionally creates a table definition file, the name specified by the TABLEDEFINITIONS argument. Scans the entire RDFFile unless the optional numeric number-of-records-to-scan is provided.
TABLEDEFINITIONS= <i>filename</i>	Specifies the name of the table definitions file. This can be either an input or an output argument depending on the operation.
TEXTSTRUCTURE	If present, reactions and molecules will be stored as CLOB rxnfiles and molfiles. If absent they will be stored as BLOB internal chemistry formats.
PRIMARYKEY= <i>fieldname</i> \$RIREG \$REREG \$MIREG \$MEREG	Specifies the primary key for the RDFFile. It may be the name of a field within the RDFFile or one of the keywords \$RIREG, \$REREG, \$MIREG or \$MEREG. These keywords refer to data lines in the RDFFile which also contain the same keyword. Unless renamed in the table definition file, the column will be named INTREG if \$RIREG or \$MIREG are used and EXTREG if \$REREG or \$MEREG are used.
TABLEPREFIX= <i>prefix</i>	When scanning an RDFFile, specifies a prefix to add to each table name.
LOAD	Load data from the input RDFFile into Oracle. If SCAN is not also specified then a TABLEDEFINITIONS input argument is required.
CREATETABLES	Creates tables in Oracle. If not specified the tables must already exist.
CREATETABLESQL= <i>filename</i>	Writes a SQL script which will create the tables. Use this instead of CREATETABLES if you need to modify table storage parameters.
PRIMARYKEYFXN= <i>"function"</i>	Use with LOAD to specify an Oracle operation or function to be applied to the top-level primary key value. Use the keyword :VAL to represent the value, for example: "(1000 + :VAL)"

See also

[Table Definitions File](#)

Using importRDFFileEx

The following are specific scenarios for using `importRDFFileEx`:

Scan an RDFFile, create the tables in Oracle, and load the RDFFile into Oracle:

```
importRDFFileEx scan load connstr=john/doe
```

Scan the first 1000 records of an RDFFile, create the table definitions file and a SQL script for creating the tables. Use the field `RXNREGNO` as the primary key. Because it does not create tables or load data it does not need an Oracle connection:

```
importRDFFileEx scan=1000 primarykey=rxnregno  
tabledefinitions=c:\tdefs.txt createtablessql=c:\tsql.sql
```

Read an existing table definitions file, create the tables and load the RDFFile:

```
importRDFFileEx load tabledefinitions=c:\tdefs.txt rdf=c:\trdf.rdf  
connstr=john/doe createtables
```

Read an existing table definitions file and load the RDFFile into existing Oracle tables, add 1000 to the primary key value, for example `RXNREGNO`, value in each of the tables containing it:

```
importRDFFileEx load tabledefinitions=c:\tdefs.txt rdf=c:\trdf.rdf  
primarykeyfxn="(1000+:VAL)" connstr=john/doe
```

Table Definitions File

The table definition file associates each RDFFile field with a specific Oracle column. It also provides information needed to define the hierarchy between tables. Before describing the table definition file note the tables needed for an RDFFile:

- A single top-level table contains a primary key column, a reaction or molecule structure column which may either be a CLOB (molfile or rxnfile) or a BLOB (binary molecule or reaction) and optionally other data columns.
- If the RDFFile contains hierarchical data there are one or more tables which have a foreign key matching the top-level table's primary key. These tables will contain their own primary key which may include the foreign key and other data columns.
- If the hierarchy extends more than two levels then there are more tables which have foreign key relationships with the tables at the second level, and so on down the hierarchy.

Example 1

The following is a partial RDFFile:

```
$RFMT $RIREG 1  
Contents of rxnfile  
$DTYPE RXN:RXNREGNO  
$DATUM 1  
$DTYPE RXN:VARIATION(1):LINE_NO  
$DATUM 1  
$DTYPE RXN:VARIATION(1):MDLNUMBER  
$DATUM RXCI92000010  
$DTYPE RXN:VARIATION(1):LITTEXT(1):LINE_NO  
$DATUM 1  
$DTYPE RXN:VARIATION(1):LITTEXT(1):LITTEXT  
$DATUM JANSSEN, A. J. M.; KLUNDER, A. J. H.; ZWANENBURG, B., Tetrahedron  
[TETRAB+  
, 47 (35), p. 7409-7416, 1991
```


`importRDFfileEx` names the top-level table `RXNTABLE`. The utility always uses `RXNTABLE` for a reaction RDFFile and `MOLTABLE` for a molecule RDFFile. The table has a primary key column. It can contain the contents of the `RXNREGNO` field or the value of the number following the `$RIREG`, (reaction internal registry number) keyword. It also has a column named `RCTAB` which contains the binary reaction.

In addition to `RXNTABLE` there are two other tables: `VARIATION` and `LITTEXT`.

`VARIATION` contains a column for the `RXNTABLE`'s primary key value, for example `RXNREGNO`, a column for its line number, the number in parenthesis following the field name `VARIATION`, a column for its `LINE_NO` value which is redundant with the line number, and a column for the `MDLNUMBER` value. `VARIATION`'s primary key is the first two of those columns.

`LITTEXT` contains the `VARIATION` table's two primary key columns, its own line number column whose value is the number in parentheses following `LITTEXT`, and a `LITTEXT` column which contains the literature reference for the reaction.

If `importRDFfileEx` is invoked as:

```
importRDFfileEx scan load primarykey=$rireg rdf=filename
connstr=user/password
```

The tables are:

```
RXNTABLE
INTREG NUMBER(9)          primary key
RCTAB  BLOB
RXNREGNO      NUMBER(9) VARIATION
INTREG NUMBER(9)          primary key, foreign key to RXNTABLE
VARIATION_LNO NUMBER(9)    primary key
LINE_NO       NUMBER(9) MDLNUMBER      VARCHAR2(4000)
```

`LITTEXT`

```
INTREG          NUMBER(9)          primary key, foreign key to VARIATION
VARIATION_LNO   NUMBER(9)          primary key, foreign key to VARIATION
LITTEXT_LNO     NUMBER(9)          primary key
LINE_NO         NUMBER(9)
LITTEXT         VARCHAR2(4000)
```

Line number columns are always named `tablename_LNO`. If you use the program argument `PRIMARYKEY=$RIREG` or `PRIMARYKEY=$MIREG`, the resulting column is named `INTREG`. If you use the program argument `PRIMARYKEY=$RREG` or `PRIMARYKEY=$MREG`, the resulting column is named `EXTREG`. You can rename any or all of the columns in the table definition file.

The table definition file contains one row for each column plus one additional row for each table that is not top-level. There is no additional row for the top-level table. Each row in the file consists of six comma-delimited fields:

- Table name.
- Column name.
- Primary key – either the table and column (`TABLE.COLUMN`) of the parent primary key value which is inherited by this table, or YES if this column is a primary key in this table.
- Type – Oracle data type used by `CREATETABLES` and `CREATETABLESQL` options, such as `NUMBER(9)` or `VARCHAR2(30)`.

- Maximum column length – this is detected during a SCAN operation and recorded here, it is not used by the program in any way.
- Name in RDFFile – a keyword \$RFMT, \$MFMT, \$RIREG, \$REREG, \$MIREG or \$MIREG; the name following \$DTYPE but without any line numbers; or the name following \$DTYPE with a (N) to denote the line number used as a primary key.

Using the [partial RDFFile](#) from above as an example, its table definition file would contain the following lines:

```
#
# Scan of all records in rdf file test1.rdf
#
# Table          - Name of table
# Column         - Name of column
# Primary key    - Table and column (as TABLE.COLUMN) of a parent primary key
#                value which is inherited by this table, or YES if this
#                column is a primary key in this table
# Type          - Oracle data type used by CREATETABLES and
#                CREATETABLESQL options
# Max length     - Maximum length of text data value in RDFFile
# RdfFile name   - Name of field in RDFFile not including line numbers,
#                suffix "(N)" indicates a field value computed from a
#                line number
#
# Table          , Column          , Primary key   , Type      , Max length, RdfFile name
RXNTABLE         , INTREG             , YES          , NUMBER(9) , 1
, $RIREG
RXNTABLE         , RCTAB              , BLOB         ,
, $RFMT
RXNTABLE         , RXNREGNO           ,              , NUMBER(9) , 1
, RXN:RXNREGNO
VARIATION        ,                    ,              ,
, RXN:VARIATION
VARIATION        , INTREG             , RXNTABLE.INTREG , NUMBER(9) , 1
,
VARIATION        , VARIATION_LNO      , YES          , NUMBER(9) , 1
, RXN:VARIATION(N)
VARIATION        , LINE_NO            ,              , NUMBER(9) , 1
, RXN:VARIATION:LINE_NO
VARIATION        , MDLNUMBER          ,              , VARCHAR2(4000),
12              , RXN:VARIATION:MDLNUMBER
LITTEXT          ,                    ,              ,
, RXN:VARIATION:LITTEXT
LITTEXT          , INTREG             , RXNTABLE.INTREG , NUMBER(9) , 1
,
LITTEXT          , VARIATION_LNO      , VARIATION.VARIATION_LNO, NUMBER(9) , 1
,
LITTEXT          , LITTEXT_LNO        , YES          , NUMBER(9) , 1
, RXN:VARIATION:LITTEXT(N)
LITTEXT          , LINE_NO            ,              , NUMBER(9) , 1
, RXN:VARIATION:LITTEXT:LINE_NO
LITTEXT          , LITTEXT            ,              , VARCHAR2(4000),
103             , RXN:VARIATION:LITTEXT:LITTEXT
```

The top-level table, named RXNTABLE, consists of three lines:

- The keyword \$RIREG in the last column tells `importRDFileEx` that the value for this column should come from the value that follows the \$RIREG keyword in the RDFile for each record. This RDFile contains less than 9 records thus the maximum length is one digit. This line also has a value of YES in the primary key column, that tells `importRDFileEx` that this column is a primary key for the table. There may be multiple primary key columns, each must have a YES.
- The keyword \$RFMT in the last column tells `importRDFileEx` that the value for this column comes from the \$RFMT top-level structure data in the RDFile. Because the type is a BLOB the data will be converted from a rxnfile to the Direct internal binary reaction format. If the type were CLOB it would be left as the rxnfile text.
- The third line defines the column for the RXNREGNO data field.

The VARIATION table entries consist of five lines:

- The line with a blank second column entry defines the RDFile path name to this level in the hierarchy and associated that with a table. The last column contains the path to the name, without any parenthetical line numbers present.
- The second line defines the parent, or top-level table's primary key value that is inherited by this table. There is no entry in the last RDFile name column. Instead the third primary key column locates the primary key in the parent table – RXNREGNO. INTREG, meaning the INTREG column in the RXNREGNO table. If there are multiple parent primary key values there must be one line here for each of them.
- The third line defines the remaining column for the primary key in this VARIATION table as the line number which occurs in the RDFile. The last column contains RXN: VARIATION(N) with the (N) suffix indicating that the actual number which occurs in the RDFile should be inserted into the table.
You do not have to use the line number, but there must be at least one column from this table which completes the primary key for the table. There may be more than one column.
- The fourth and fifth lines name additional data fields in the VARIATION table.

The entries for LITTEXT follow the same format as for VARIATION, because LITTEXT is beneath VARIATION in the hierarchy it has three rows defining its primary key.

You can edit the table definition file by removing unwanted rows, fields in the RDFile, or by changing the table name, column name, primary key or type. Do not change the entries in the last column of the file. These must match the actual \$DTYPE lines in the RDFile. If you change a table or column name, be sure to change it everywhere that it occurs in the table definition file – both in the first or second columns of the file and also wherever it is used to define a primary key column.

IMPORTANT! You can use any fields as primary keys, as long as they are unique. The program does not enforce uniqueness. Using non-unique values will corrupt your data.

When reading an RDFile into existing tables you might need to adjust the primary key value in the RDFile, such as the value of RXNREGNO, so that it does not conflict with the values already in the various tables. As long as you have only one top-level primary key column, you can apply an Oracle function or operator to the primary key value read from the RDFile before inserting the value into the tables. Use the PRIMARYKEYFXN parameter to specify the operation, use :VAL to represent the value read in from the RDFile. Enclose the operation or function in double quotes. For example, if the primary key is numeric you could add 1000000 to the input value:

```
importRDFileEx load primarykeyfxn="(1000000 + :VAL)"
```

Or if the primary is a text value you could add a prefix:

```
importRDFileEx load primarykeyfxn="'NEW_' || :VA
```

Example 2

This is an example of a molecule RDFFile containing chemical supplier data:

```
$RDFILE 1
$DATM
$MFMT
Contents of molfile
$DTYPE MOL:CAPID
$DATUM CAP00000001
$DTYPE MOL:STATUS
$DATUM 1
$DTYPE MOL:CAS
$DATUM 27469-60-9
$DTYPE MOL:MDLNUMBER
$DATUM 38660
$DTYPE MOL:INCHIKEY
$DATUM TTXIFFYPVGLSE-UHFFFAOYSA-N
$DTYPE MOL:SYNONYMS(1):CPDSYNONYM
$DATUM 1 [bis(4-Fluorophenyl)methyl]-piperazine
$DTYPE MOL:SYNONYMS(2):CPDSYNONYM
$DATUM 1-(4,4'-Difluorobenzhydryl)piperazine
$DTYPE MOL:SYNONYMS(3):CPDSYNONYM
$DATUM 1-Bis(4-fluorophenyl)methyl piperazine
$DTYPE MOL:PRODUCTS(1):CAPID
$DATUM CAP00000001
$DTYPE MOL:PRODUCTS(1):SUPPRECID
$DATUM 9563209
$DTYPE MOL:PRODUCTS(1):SUPPCODE
$DATUM APOLLO
$DTYPE MOL:PRODUCTS(1):SUPPCATNO
$DATUM PC5636
$DTYPE MOL:PRODUCTS(1):CPDTYPE
$DATUM R
$DTYPE MOL:PRODUCTS(1):AVAIL
$DATUM A
$DTYPE MOL:PRODUCTS(1):SUPPNAME
$DATUM Apollo Scientific Ltd.
$DTYPE MOL:PRODUCTS(1):PRODNAME
$DATUM 1-(4,4'-Difluorobenzhydryl)piperazine 97%
$DTYPE MOL:PRODUCTS(1):PRICES(1):SUPPRECID
$DATUM 9563209
$DTYPE MOL:PRODUCTS(1):PRICES(1):SEQNO
$DATUM 1
$DTYPE MOL:PRODUCTS(1):PRICES(1):QTY
$DATUM 1g
$DTYPE MOL:PRODUCTS(1):PRICES(1):PRICE
$DATUM GBP 35
$DTYPE MOL:PRODUCTS(2):CAPID
$DATUM CAP00000001
$DTYPE MOL:PRODUCTS(2):SUPPRECID
$DATUM 17800549
$DTYPE MOL:PRODUCTS(2):SUPPCODE
$DATUM IBS
```

```

$DTYPE MOL:PRODUCTS(2):SUPPCATNO
$DATUM BB_SC-1095
$DTYPE MOL:PRODUCTS(2):CPDTYPE
$DATUM R$DTYPE MOL:PRODUCTS(2):AVAIL
$DATUM A
$DTYPE MOL:PRODUCTS(2):SUPPNAME
$DATUM InterBioScreen Ltd.
$DTYPE MOL:PRODUCTS(2):PRODNAME
$DATUM 1-Bis(4-fluorophenyl)methyl piperazine 97%
$DTYPE MOL:PRODUCTS(2):PRICES(1):SUPPRECID
$DATUM 17800549
$DTYPE MOL:PRODUCTS(2):PRICES(1):SEQNO
$DATUM 1
$DTYPE MOL:PRODUCTS(2):PRICES(1):QTY
$DATUM 500mg
$DTYPE MOL:PRODUCTS(2):PRICES(1):PRICE
$DATUM USD 36
$DTYPE MOL:PRODUCTS(2):PRICES(2):SUPPRECID
$DATUM 17800549
$DTYPE MOL:PRODUCTS(2):PRICES(2):SEQNO
$DATUM 2
$DTYPE MOL:PRODUCTS(2):PRICES(2):QTY
$DATUM 1g
$DTYPE MOL:PRODUCTS(2):PRICES(2):PRICE
$DATUM USD 43

```

We can scan the file to create a table definitions file using the following command.

There is no \$MIREG or \$MEREK keyword so when we scan the file we must specify a primary key field:

```

C:\> importRDFFileEx scan ^
                        primarykey=capid ^
                        rdfile=cap_example.rdf ^
                        tabledefinitions=tbldef.txt

```

The table definitions file created by importRDFFileEx contains the following entries:

#	Table	Column	Primary key	Type	Max length
	RDFFile name				
	MOLTABLE, CTAB			BLOB	
	\$MFMT				
	MOLTABLE, CAPID		YES	VARCHAR2(4000)	11
	MOL:CAPID				
	MOLTABLE, STATUS			NUMBER(9)	1
	MOL:STATUS				
	MOLTABLE, CAS			VARCHAR2(4000)	10
	MOL:CAS				
	MOLTABLE, MDLNUMBER			NUMBER(9)	5
	MOL:MDLNUMBER				
	MOLTABLE, INCHIKEY			VARCHAR2(4000)	27
	MOL:INCHIKEY				
	SYNONYMS,				
	MOL:SYNONYMS				
	SYNONYMS, CAPID		MOLTABLE.CAPID	VARCHAR2(4000)	11
	SYNONYMS, SYNONYMS_LNO		YES	NUMBER(9)	1

```

MOL : SYNONYMS (N)
SYNONYMS, CPDSYNONYM      , VARCHAR2 (4000), 40      ,
MOL : SYNONYMS : CPDSYNONYM
PRODUCTS,                  ,                  ,          ,
MOL : PRODUCTS
PRODUCTS, CAPID            , MOLTABLE.CAPID      , VARCHAR2 (4000), 11      ,
PRODUCTS, PRODUCTS_LNO, YES      , NUMBER (9)          , 1        ,
MOL : PRODUCTS (N)
PRODUCTS, CAPID1          , VARCHAR2 (4000), 11      ,
MOL : PRODUCTS : CAPID
PRODUCTS, SUPPRECID       , NUMBER (9)          , 8        ,
MOL : PRODUCTS : SUPPRECID
PRODUCTS, SUPPCODE        , VARCHAR2 (4000), 6        ,
MOL : PRODUCTS : SUPPCODE
PRODUCTS, SUPPCATNO       , VARCHAR2 (4000), 10       ,
MOL : PRODUCTS : SUPPCATNO
PRODUCTS, CPDTYPE         , VARCHAR2 (4000), 1        ,
MOL : PRODUCTS : CPDTYPE
PRODUCTS, AVAIL           , VARCHAR2 (4000), 1        ,
MOL : PRODUCTS : AVAIL
PRODUCTS, SUPPNAME        , VARCHAR2 (4000), 22       ,
MOL : PRODUCTS : SUPPNAME
PRODUCTS, PRODNAME        , VARCHAR2 (4000), 42       ,
MOL : PRODUCTS : PRODNAME
PRICES ,                   ,                  ,          ,
MOL : PRODUCTS : PRICES
PRICES , CAPID            , MOLTABLE.CAPID      , VARCHAR2 (4000), 11      ,
PRICES , PRODUCTS_LNO, PRODUCTS.PRODUCTS_LNO, NUMBER (9)          , 1        ,
PRICES , PRICES_LNO , YES      , NUMBER (9)          , 1        ,
MOL : PRODUCTS : PRICES (N)
PRICES , SUPPRECID       , NUMBER (9)          , 8        ,
MOL : PRODUCTS : PRICES : SUPPRECID
PRICES , SEQNO           , NUMBER (9)          , 1        ,
MOL : PRODUCTS : PRICES : SEQNO
PRICES , QTY             , VARCHAR2 (4000), 5        ,
MOL : PRODUCTS : PRICES : QTY
PRICES , PRICE           , VARCHAR2 (4000), 6        ,
MOL : PRODUCTS : PRICES : PRICE

```

importRDFFileEx has renamed the CAPID column in the PRODUCTS table to CAPID1 because the name conflicts with the inherited top-level primary key field. Inherited primary key fields are automatically added to all child tables.

The program has added columns SYNONYMS_LNO, PRODUCTS_LNO and PRICES_LNO so that each child table has a valid complete primary key.

We would like to simplify the table definitions:

- We want to delete the CAPID1 column; we have an automatic inherited copy of the top-level CAPID column.
- The SYNONYMS table does not need a complete primary key, so we can remove SYNONYMS_LNO.
- We know that the primary key for PRODUCTS is (CAPID,SUPPRECID) so we can remove PRODUCTS_LNO and replace it with SUPPRECID in both the PRODUCTS table and the PRICES table. Note that we need to inherit SUPPRECID in PRICES.

- We know that the primary key for PRICES is (CAPID,SUPPRECID,SEQNO) so we can remove PRICES_LNO and use SEQNO instead.
- We can adjust some column widths and types if desired.

After making these changes (highlighted in yellow) we have the following new table definitions file:

# Table	Column	Primary key	Type	Max length
MOLTABLE, CTAB			BLOB	
\$MFMT				
MOLTABLE, CAPID	YES		VARCHAR2(11)	11
MOL:CAPID				
MOLTABLE, STATUS			NUMBER(9)	1
MOL:STATUS				
MOLTABLE, CAS			VARCHAR2(20)	10
MOL:CAS				
MOLTABLE, MDLNUMBER			NUMBER(9)	5
MOL:MDLNUMBER				
MOLTABLE, INCHIKEY			VARCHAR2(27)	27
MOL:INCHIKEY				
SYNONYMS,				
MOL:SYNONYMS				
SYNONYMS, CAPID	MOLTABLE.CAPID		VARCHAR2(11)	11
# Removed SYNONYMS_LNO				
SYNONYMS, CPDSYNONYM			VARCHAR2(4000)	40
MOL:SYNONYMS:CPDSYNONYM				
PRODUCTS,				
MOL:PRODUCTS				
PRODUCTS, CAPID	MOLTABLE.CAPID		VARCHAR2(11)	11
# Removed PRODUCTS_LNO, using SUPPRECID instead				
# Removed CAPID1, redundant now				
PRODUCTS, SUPPRECID	YES		NUMBER(9)	8
MOL:PRODUCTS:SUPPRECID				
PRODUCTS, SUPPCODE			VARCHAR2(4000)	6
MOL:PRODUCTS:SUPPCODE				
PRODUCTS, SUPPCATNO			VARCHAR2(4000)	10
MOL:PRODUCTS:SUPPCATNO				
PRODUCTS, CPDTYPE			VARCHAR2(10)	1
MOL:PRODUCTS:CPDTYPE				
PRODUCTS, AVAIL			VARCHAR2(10)	1
MOL:PRODUCTS:AVAIL				
PRODUCTS, SUPPNAME			VARCHAR2(4000)	22
MOL:PRODUCTS:SUPPNAME				
PRODUCTS, PRODNAME			VARCHAR2(4000)	42
MOL:PRODUCTS:PRODNAME				
PRICES				
MOL:PRODUCTS:PRICES				
PRICES, CAPID	MOLTABLE.CAPID		VARCHAR2(11)	11
# Removed PRODUCTS_LNO, using SUPPRECID instead				
PRICES, SUPPRECID	PRODUCTS.SUPPRECID		NUMBER(9)	8
MOL:PRODUCTS:PRICES:SUPPRECID				
# Removed PRICES_LNO, using SEQNO instead				
PRICES, SEQNO	YES		NUMBER(9)	1
MOL:PRODUCTS:PRICES:SEQNO				

```
PRICES      , QTY              , VARCHAR2(30), 5      ,  
MOL:PRODUCTS:PRICES:QTY  
PRICES      , PRICE           , VARCHAR2(30), 6      ,  
MOL:PRODUCTS:PRICES:PRICE
```

We can read in the RDFFile using this new table definitions file as input, we also create a file containing the SQL commands that will be executed by `importRDFFileEx` as it creates tables:

```
C:\> importRDFFileEx load ^  
                        connstr=chemistryuser/password ^  
                        createtables ^  
                        tabledefinitions=new_tbldef.txt ^  
                        rdfile=cap_example.rdf ^  
                        createtablessql=cap_create.sql
```

The RDFFile Import Utility

The Direct RDFFile import utility is designed to read a file of molecule or reaction structures and associated data, and to store the records in one or more Oracle tables. This is intended as the first step in the process of registering structures and related data to a molecule or reaction database. After the data has been imported into Oracle, you can apply Oracle-based tools to validate the data, apply business rules, and perform any other processing needed to register the data into your molecule or reaction database.

Direct provides two utilities for importing RDFFiles:

- The RDFFile Import Utility (`importRDFFile`) - read REACCS and ISIS RDFFiles.
- The [Extended RdfFile Import Utility](#) (`importRDFFileEx`) - customizable and can read RDFFiles generated by any external program.

Note: The purpose of the Direct RDFFile import utility is to load data based on the BIOVIA reactions data model. If you have an existing molecule or reaction database or want to set-up a specific data model, you need to transform the data hierarchy to map onto the set of tables. This step can be done within Oracle by Oracle utilities.

The RDFFile import utility does not provide any means to connect table object names and data types.

If you need further assistance, contact BIOVIA consulting to help.

To use the RDFFile import utility, you:

1. Scan the input file (or a small representative example if there are multiple input files).
2. Create the intermediate table(s) into which the data will be read.
3. Load the data.
4. Perform any data validations, and apply business rules or other transformations on the data.
5. Register the data to the target molecule or reaction database.
6. Drop the intermediate table(s).

Note: If your RDFFile contains non-ASCII characters, before you run `importRDFFile` set the `NLS_LANG` environment variable to a language setting which matches the characters in your RDFFile. Failure to do this may result in Oracle storing the wrong characters. Because of limitations in the character handling in `importRDFFile`, non-ASCII characters which occur in field names will be replaced with underscores.

RDFFile Import Utility Operation

The RDFFile import utility is able to perform the scan, table creation, and load steps in one pass. However, you might want to separate the scan and load passes either to permit customizing of the table creation SQL or to efficiently load large volumes of data.

Note the following prerequisites for an RDFFile import:

- The RDFFile must include a \$MIREG or \$RIREG entry on the \$MFMT or \$RFMT line of each record, respectively.
- The values of the \$MIREG or \$RIREG entries must be numeric and unique for each record in the RDFFile to be imported.
- If the RDFFile also contains a data field named *molregno* or *rxnregno*, respectively, the value in a particular record must be identical to the \$MIREG or \$RIREG value of the same record. The data field must otherwise be renamed to be imported.
- The root name of the data tree must be either MOL or RXN, or the data part of the RDFFile will not be imported.

During the scan pass, the RDFFile import utility generates a table creation SQL file, a table drop SQL file, and a database description XML file. During the load pass, the XML file is read and used to register the contents of the RDFFile. When the scan and load passes are performed in a single step, the RDFFile import utility runs the table creation SQL script.

If you separate the scan pass and load pass into separate calls of the RDFFile import utility, you must run the table creation SQL script before invoking the RDFFile import utility for the load pass. This separation offers you an opportunity to edit the table creation SQL script.

The RDFFile import utility uses the data-type path name on each \$DTYPE line to determine where in the hierarchy that data type belongs. The scan pass discerns the structure of the hierarchy and the data type of each field. At the end of the scan pass, the RDFFile import utility writes out an XML file, which describes the hierarchy, as well as two SQL files: one for table creation, and one for later use in dropping the tables.

Note: The name on the \$DTYPE line must consist of ASCII characters. The name will be uppercased based on the locale setting of the system where the importRDFFile program is run.

Tables and Columns

Parent fields become tables. Leaf data fields become columns in those tables. RXNREGNO, MOLREGNO, and line-number columns are used to relate the rows of these tables together.

Because Oracle places many limitations on the names of tables and columns, the RDFFile import utility might alter the name of a table or column to:

- shorten it
- make names unique
- prevent collision of names with Oracle reserved words
- remove restricted characters

tableprefix Parameter

The RDFFile import utility always quotes table, index, and column names to allow the maximum flexibility in object naming, and to support European special characters and Kanji characters in names. Because of this, specify the `tableprefix` parameter in uppercase, unless you intend to create object names with lowercase characters in the names.

The RDFFile import utility also uses the `tableprefix` parameter to form file names, and so this parameter must be valid for use as a file name on your computer.

If the `tableprefix` is `apple`, then the intermediate files are named:

```
apple.xml  
apple.sql  
apple_DROP.sql
```

In this case, a table of molecules created for this RDFFile will be named `apple_MOL`, which is a name that needs to be quoted in SQL syntax. To have names that do not need to be quoted in SQL syntax, you need to specify the `tableprefix` in uppercase, for example, `APPLE`. This results in the intermediate files named:

```
APPLE.xml  
APPLE.sql  
APPLE_DROP.sql
```

Handling of Specialized Conventions

The `importRDFFile` program has no knowledge of the relationship between reactions and molecules and has no knowledge of specialized conventions, such as the convergence of sub-hierarchies at the MOL parent. Consequently, the RDFFile import utility might create multiple tables to contain lower-level molecule or reaction structures and data. There might be, for example, distinct reactant molecule and product molecule tables. The `importRDFFile` program is only one part of the registration process. Additional processing steps might be required after the data is loaded in order to produce a meaningful reaction database.

Syntax of the importRDFFile Program

The syntax of the `importRDFFile` program is as follows:

```
importRDFFile [-nowait] [-scan ] [-load] [ -char ] inputfile tableprefix  
[ connectstring ]
```

The text in *italic* print indicates variables that are specific to your site, for example:

```
importRDFFile -scan archive1.rdf OLDDB  
importRDFFile -load archive1.rdf OLDDB myuser/mypw  
importRDFFile -load archive2.rdf OLDDB myuser/mypw  
importRDFFile -char incoming_mols.rdf MOLFILETAB test/test
```

Processing Options of the importRDFFile Program

Create the Intermediate Table

The `importRDFFile` program loads data based on an XML description. The program creates that description during its scan pass. The scan pass also is used to generate the table creation and table drop SQL scripts. You can limit the amount of data that is scanned by specifying the `-scan` switch on the command line and supplying a small, representative, RDFFile as the input.

For example, if `archive1.rdf` is a relatively small RDFFile that contains data of all the possible fields which can appear in the hierarchy, and `archive.rdf` is a much larger file (containing the rest of the structure and data you want to load) then you can efficiently load the data using the following steps.

1. `importRDFFile -scan archive.rdf OLDDB`
2. `sqlplus myuser/mypw <OLDDB.sql`

3. `importRDFFile -load archive1.rdf OLDDDB myuser/mypw`
4. `importRDFFile -load archive2.rdf OLDDDB myuser/mypw`

where:

- The first `importRDFFile` command is used to scan `archive1.rdf` and produce the files `OLDDDB.xml`, `OLDDDB.sql`, and `OLDDDB_DROP.sql`.
- The `SQL*Plus` command is used to create the tables.
- The remaining `importRDFFile` commands are used to load the data.

You can perform exactly the same task using just two commands:

```
importRDFFile archive1.rdf OLDDDB myuser/mypw
importRDFFile -load archive2.rdf OLDDDB myuser/mypw
```

In this second case, by specifying neither `-scan` or `-load`, the `importRDFFile` program scans the file `archive1.rdf`, creates the tables, and loads the file. Since the `OLDDDB.xml` file is still present in the working directory, you can use it to load `archive2.rdf` without any need to scan the file.

IMPORTANT! Do not attempt to edit the XML file generated by `importRDFFile`. You can edit the table creation SQL file to alter the characteristics of the tables, such as adding tablespace names, but do not alter the table names or column names, as this will cause the subsequent load to fail.

Import Structures in Character(-char) Mode

Molecule and reaction structures at the top of the hierarchy are loaded in binary mode such as BLOB unless you specify the `-char` switch. If you specify `-char`, the top-level molecule or reaction structures are loaded as data type CLOB. This is useful in cases where you want to:

- preserve the original form of the molfile or reaction file
- process the structure with tools such as BIOVIA Cheshire before registering the structure to your molecule or reaction database.

Note: Embedded molfiles or reaction files at lower levels are always loaded as CLOB.

Oracle O/S Authentication

To connect to Oracle using O/S authentication, omit the user name and password from the connect string.

The `"/` character is a *required* part of the syntax. For example:

```
importRDFFile myfile.rdf MY /
importRDFFile myfile.rdf MY /@dbalias
```

NOWAIT Option

After each record is inserted, an Oracle `COMMIT` is performed to complete the transaction. However, to maximize import speed, you can use the `-nowait` switch to instruct the `importRDFFile` program to use the Oracle `COMMIT WRITE NOWAIT BATCH` command instead of an ordinary `COMMIT`. Consult your Oracle documentation for more information about the risks and benefits of using `COMMIT WRITE NOWAIT BATCH`.

Numeric and Date Field Processing

The `importRDFFile` program passes all data fields to Oracle in character form. Default Oracle processing rules for numeric and date fields apply.

If you import numeric values into table columns whose data type is specified as NUMBER, Oracle converts the values using the NLS_NUMERIC_CHARACTERS settings of your session. If the decimal separator does not match the setting, the conversion can fail. You might want to import numeric values into CHAR or VARCHAR2 columns, and convert the values with your own application after the import.

Note: To alter the data type of a column, you must use separate scan and load phases, so that you have an opportunity to edit and run the table creation SQL script.

If you import date values into table columns whose data type is specified as DATE, Oracle converts the values using the NLS_DATE_FORMAT settings of your session. If the input date strings do not match the setting, the conversion can fail. You might want to import date values into CHAR or VARCHAR2 columns, and convert the values with your own application after the import.

Range-valued fields must be imported as character values and interpreted by your application.

Control/C Interrupt

You can terminate a load by pressing Ctrl C.

Error Handling

In most cases, the exportSDFile or exportRDFFile utility stops exporting if it encounters an Oracle error.

The following error codes are suppressed and ignored:

- ora-00100 no data found
- ora-01403 no data found
- ora-12652 string truncated
- ora-24347 warning of a NULL column in an aggregate function

The following error codes are reported, and the current record might be omitted from the output, but the export will continue. These error codes represent common conditions, such as formatting errors.

- ora-00001 unique constraint (schema.constraintname) violated
- ora-00054 resource busy and acquire with NOWAIT specified
- ora-00060 deadlock detected while waiting for resource
- ora-00910 specified length too long for its datatype
- ora-00912 input parameter too long
- ora-00972 identifier is too long
- ora-01400 cannot insert NULL into (column)
- ora-01426 numeric overflow
- ora-01555 snapshot too old
- ora-01704 string literal too long
- ora-01722 invalid number
- ora-01724 floating point precision is out of range
- ora-01841 (full) year must be between -4713 and +9999, and not be 0
- ora-01854 julian date must be between 1 and 5373484
- ora-01858 a non-numeric character was found where a numeric was expected
- ora-01859 a non-alphabetic character was found where an alphabetic was expected
- ora-01861 literal does not match format string
- ora-01862 the numeric value does not match the length of the format item
- ora-01863 the year is not supported for the current calendar
- ora-01864 the date is out of range for the current calendar
- ora-01865 not a valid era
- ora-01866 the datetime class is invalid

```

ora-01867 the interval is invalid
ora-01868 the leading precision of the interval is too small
ora-01870 the intervals or datetimes are not mutually comparable
ora-01871 the number of seconds must be less than 60
ora-01873 the leading precision of the interval is too small
ora-01874 time zone hour must be between -12 and 14
ora-01875 time zone minute must be between -59 and 59
ora-01876 year must be at least -4713
ora-01878 specified field not found in datetime or interval
ora-01879 the hh25 field must be between 0 and 24
ora-01880 the fractional seconds must be between 0 and 999999999
ora-01881 timezone region id is invalidora-02290 check constraint violated
ora-02291 integrity constraint
ora-20100 errors or warnings from BIOVIA Direct
ora-29875 error during ODICIndexIndexInsert

```

The Direct error stack is displayed after any error.

The Direct error stack cannot be displayed unless the target schema has the Direct synonyms defined and has rights to run the Direct cartridge. If the Direct cartridge is not installed, the exportSDFFile or exportRDFFile program are not able to format binary CTABs. In this case, the program is only capable of exporting columns already formatted as text.

Tables Created by RDFFile Import Utility

For a reaction RDFFile, the importRDFFile program creates tables with the following suffixes added to the tableprefix parameter:

_RXN	Table of reactions and top-level data fields
_PRD	Table for use in relating reactions to product molecules (can be empty)
_RCT	Table for use in relating reactions to reactant molecules (can be empty)
_parentname	Table(s) created to store data for the fields under 'parentname'.

For a molecule RDFFile, the importRDFFile program creates tables with the following suffixes added to the tableprefix parameter:

_MOL	Table of molecules and top-level data fields
_parentname	Table(s) created to store data for the fields under 'parentname'.

The reaction RDFFile that includes molecules and molecule data have molecules stored in character (molfile) form and the tables created reflect the hierarchy invoked by the field pathnames that appear in the RDFFile. This means that there are separate tables for reactant and product molecules and their data, and for catalyst and solvent molecules and their data if present. The importRDFFile program does not attempt to merge the subtrees at the MOL parent, but creates separate tables. You might want to reorganize the data into fewer tables after the import has completed.

Duplicate Checking

The importRDFFile program does not provide structure duplicate checking. BIOVIA recommends that duplicate checking be performed as a post-processing step.

Domain Index Creation

The importRDFfile program does not create molecule or reaction domain indexes on any tables.

Appendix D:

SDFFile and RDFFile Export Utilities

Overview of the Export Utilities

The Direct SDFFile and RDFFile export utilities are designed to create a file of molecule or reaction structures and their properties from a simple Oracle table with one molecule column or one reaction column. The molecule or reaction table can have additional columns for properties. The supported data types for properties are:

- CHAR
- VARCHAR2
- CLOB
- DATE
- NUMBER

Other types of data or more complex relationships are outside of the scope of the SDFFile and RDFFile export utilities.

The SDFFile export utility is designed to create simple molecule SDFFiles. An SDFFile contains one or more records that consist of a molecule structure (molfile) followed by data fields, all of which must have a simple one-to-one relationship to the molecule structure.

The RDFFile export utility is designed to create simple reaction RDFFiles or molecule RDFFiles with a single-level hierarchy. Note that representing the relationship between reactions and molecules is outside the scope of this utility. A reaction RDFFile contains one or more records that consist of a reaction (rxnfile) followed by data fields, all of which must have a simple one-to-one relationship to the reaction structure.

The SDFFile and RDFFile export utilities draw data from two kinds of record sources.

Either:

- A table or view specified with the `-table` switch.

or

- One or more SELECT statements supplied in a file with the `-sql` switch.

The choice of which columns to use, what names to use for the columns, how they should be formatted, and which rows should be selected are controlled by configuring the record source using Oracle tools.

Some examples of how to configure a record source:

- Create an intermediate table that contains the desired rows and columns, and where the column names and values are exactly as they need to appear in the RDFFile or SDFFile.
- Create a view that selects the desired rows and columns and that provides column names and any data formatting needed to create the desired output.
- Create a .SQL file that contains one or more SELECT statements that produces the desired record set, columns, column names, and formatting.

Typical steps in using the SDFFile export or RDFFile export utility are:

1. Create an intermediate table or view that presents the data exactly as desired.
2. Export the data, specifying the `-table` switch and the table or view name.
3. Drop the intermediate table or view.

or

1. Create a .SQL file that selects the desired records and formats them appropriately.
2. Export the data using the `-sql` switch and specifying the name of the .SQL file.

Usage

The following sections explain how to use the SDFFile and RDFFile Export utilities.

ExportSDFFile

Syntax

```
exportSDFFile [ -ctab molcol ] [ -sql sqlfile | -table tab_or_vu ]  
connectstring outputfile
```

Examples

```
exportSDFFile -table sample2d isis/ sample2d.sdf  
exportSDFFile -ctab Molecule -table myview isis/ myview.sdf  
exportSDFFile -ctab Structure -sql myselection.sql isis/isis@mynode  
mydata.sdf
```

Designating the Molecule Column

Use the `-ctab` switch to specify the name of the molecule column. If the `-ctab` switch is not present, the `exportSDFFile` utility looks for a column named CTAB, and if none is found, the utility attempts to use the first CLOB or BLOB column that it finds.

If the CTAB column is a BLOB and the `-table` switch was specified, then the `exportSDFFile` utility constructs an alternative SQL SELECT statement using the column names from the table or view, for example:

```
SELECT MOLFILE(ctab_column) as ctab_column, other columns,... FROM table_or_  
view
```

Note: If the `-sql` switch is specified, the CTAB column must be a character type (such as CLOB).

Specifying the Record Source

To select all rows and columns from a table or view, specify the `-table` switch and the table or view name.

To supply one or more SELECT statements, create a text file that contains the SELECT statements and specify that file using the `-sql` switch. The lines of the file are processed in the order in which they appear, and the output of each SELECT statement is written out to the SDFFile.

Controlling Record Order

Use an ORDER BY clause in your view or SELECT statement to control the record order. For example:

```
SELECT MOLFILE(CTAB) AS CTAB FROM SAMPLE2D ORDER BY CORP_ID ASC;
```

Controlling Which Records are Included

Use an appropriate WHERE clause in your VIEW or SELECT statement to restrict the record set, or use multiple SELECT statements to specify the desired set of records.

Examples:


```
CREATE OR REPLACE VIEW MYVU AS SELECT MOLFILE(CTAB) AS CTAB FROM SAMPLE2D
WHERE CDBREGNO IN (11,13,17);
SELECT MOLFILE(CTAB) AS CTAB FROM SAMPLE2D WHERE CDBREGNO=11;
SELECT MOLFILE(CTAB) AS CTAB FROM SAMPLE2D WHERE CDBREGNO=13;
SELECT MOLFILE(CTAB) AS CTAB FROM SAMPLE2D WHERE CDBREGNO=17;
```

Renaming Columns

You can create a table that uses the column names as you want them to appear in the SDFFile, or you can use Oracle column aliases in the CREATE VIEW or SELECT statements.

Examples:

```
CREATE OR REPLACE VIEW MYVU AS SELECT MOLFILE(CTAB) AS CTAB, CORP_ID AS
EXTREG, MOLWT(CTAB) AS "MOL.WT" FROM SAMPLE2D WHERE CDBREGNO IN (11,13,17);
SELECT MOLFILE(CTAB) AS CTAB, CORP_ID AS "Extreg", molwt(ctab) as "MOL.WT"
FROM SAMPLE2D WHERE CDBREGNO=11;
```

Formatting Columns

The exportSDFFile utility fetches all data fields from Oracle in character form. Default Oracle processing rules for converting to character form apply. The presence of fields that cannot be converted might cause the export to fail.

You can create a table that formats each column, in character form, exactly as you want it to appear, or you can apply formatting in a CREATE VIEW or SELECT statement.

Examples:

```
CREATE TABLE TEMP1 AS SELECT MOLFILE(CTAB) AS CTAB, CORP_ID AS EXTREG, TO_
CHAR(F_DATE, 'yyyy-MON-dd') as "Date" FROM SAMPLE2D;
SELECT MOLFILE(CTAB) AS CTAB, CORP_ID AS EXTREG, TO_CHAR(F_DATE, 'yyyy-MON-
dd') as "Date" FROM SAMPLE2D ORDER BY CORP_ID;
```

To control the appearance of numeric and date fields, you can either specify the precise conversion to character form in a VIEW or SELECT statement or you can set the Oracle NLS_NUMERIC_CHARACTERS and NLS_DATE_FORMAT environment settings.

The -sql option allows any simple SQL statement, and so it is possible to use ALTER SESSION commands in the -sql script to control Oracle environment settings.

Oracle O/S Authentication

To connect to Oracle using O/S authentication, omit the username and password from the connect string.

The "/" character is a *required* part of the syntax. For example:

```
importSDFFile -2d myfile.sdf moltab /
importSDFFile -2d myfile.sdf moltab /@dbalias
```

ExportRDFFile

Syntax

```
exportRDFFile [-intreg regnocol | -extreg extregcol] [ -ctab molcol | -rctab
rxnocol ] [-sql sqlfile | -table tab_or_vu ] connectstring outputfile
```

Examples

```
exportRDFFile -ctab -table sample2d .rdf
exportRDFFile -ctab -intreg cdbregno sample2d sample2d2.rdf
exportRDFFile -intreg rxnregno -sql myrxns.sql isis/isis myrxns.rdf
```

Specifying a MOLREGNO or RXNREGNO Column

You can specify the column to use to supply the value for the MOLREGNO for molecule RDFFiles or RXNREGNO for reaction RDFFiles by supplying the column name on the `-intreg` switch.

If you are creating a molecule RDFFile for importing with the Direct `importRDFFileEx` or `importRDFFile` utility, you must either specify the `-intreg` column, or you need to apply a *before insert* trigger to the target table before importing the data so that the MOLREGNO is supplied. The Direct `importRDFFileEx` and `importRDFFile` utilities ignore columns named MOLREGNO and RXNREGNO in the data portion of the RDFFile.

The `-intreg` column value appears on the \$MFMF line after the \$MIREG keyword.

Specifying an extreg Column

You can optionally specify a molecule or reaction extreg column using the `-extreg` switch. This value is added to the record of each structure next to the \$MEREK or \$REREK keyword on the \$MFMF or \$RFMF line.

Note: This value is not used by the Direct `importRDFFile` or `importRDFFileEx` utility.

Specifying a Molecule RDFFile Versus a Reaction RDFFile

Use the `-ctab` keyword to indicate that the RDFFile is a molecule RDFFile. The default is to assume that the file is a reaction RDFFile.

Designating the Molecule Column

Use the `-ctab` switch to specify the name of the molecule column. This switch also has the side-effect of designating the file as a molecule RDFFile.

If the CTAB column is a BLOB and the `-table` switch was specified, then the `exportRDFFile` utility constructs an alternative SQL SELECT statement using the column names from the table or view, for example:

```
SELECT MOLFILE(ctab_column) as ctab_column, other columns,... FROM table_or_
view
```

If the `-sql` switch is specified, the CTAB column must be a character type, such as CLOB.

Specifying the Record Source

To select all rows and columns from a table or view, specify the `-table` switch and the table or view name.

To supply one or more SELECT statements, create a text file that contains the SELECT statements and specify that file using the `-sql` switch. The lines of the file are processed in the order in which they appear, and the output of each SELECT statement is written out to the RDFFile.

Controlling Record Order

Use an ORDER BY clause in your view or SELECT statement to control the record order, for example:

```
SELECT RXNFILE(RCTAB) AS RCTAB FROM SAMPLERX_REACTION ORDER BY RXNMDLNUMBER;
```

Controlling Included Records

Use an appropriate WHERE clause in your VIEW or SELECT statement to restrict the record set, or use multiple SELECT statements to specify the desired set of records.

Examples:

```
CREATE OR REPLACE VIEW MYVU AS SELECT RXNFILE(RCTAB) AS RCTAB FROM SAMPLERX_
REACTION WHERE RXNMDLNUMBER LIKE 'RXCI9%';

SELECT RXNFILE(RCTAB) AS RCTAB FROM SAMPLERX_REACTION
WHERE RXNMDLNUMBER = 'RXCI92000009';

SELECT RXNFILE(RCTAB) AS RCTAB FROM SAMPLERX_REACTION
WHERE RXNMDLNUMBER = 'RXCI92000010';

SELECT RXNFILE(RCTAB) AS RCTAB FROM SAMPLERX_REACTION
WHERE RXNMDLNUMBER = 'RXCI92000014';
```

Renaming Columns

You can create a table that uses the column names as you want them to appear in the SDFFile, or you can use Oracle column aliases in the CREATE VIEW or SELECT statements.

Examples:

```
CREATE OR REPLACE VIEW MYVU AS SELECT RXNFILE(RCTAB) AS RCTAB, RXNMDLNUMBER
AS EXTREG FROM SAMPLERX_REACTION;

SELECT RXNFILE(RCTAB) AS RCTAB, RXNMDLNUMBER AS "Extreg" FROM SAMPLERX_
REACTION;
```

Formatting Columns

The exportRDFFile utility fetches all data fields from Oracle in character form. Default Oracle processing rules for converting to character form apply. The presence of fields that cannot be converted can cause the export to fail.

You can format the columns of table using CREATE VIEW or SELECT statement.

Examples:

```
CREATE TABLE TEMP1 AS SELECT RXNFILE(RCTAB) AS RCTAB, RXNMDLNUMBER AS
EXTREG, substr(rxnmdlnumber,1,4) as "Class" FROM SAMPLERX_REACTION;

SELECT RXNFILE(RCTAB) AS RCTAB, RXNMDLNUMBER AS EXTREG, substr
(rxnmdlnumber,1,4) as "Class" FROM SAMPLERX_REACTION;
```

To control the appearance of numeric and date fields, you can either specify the precise conversion to character form in a VIEW or SELECT statement, or you can set the Oracle NLS_NUMERIC_CHARACTERS and NLS_DATE_FORMAT environment settings.

The -sql option allows any simple SQL statement, and so it is possible to use ALTER SESSION commands in the -sql script to control Oracle environment settings.

Oracle O/S Authentication

To connect to Oracle using O/S authentication, omit the username and password from the connect string. The "/" character is a required part of the syntax, for example:

```
exportRDFFile -sql select.sql / myfile.rdf
exportRDFFile -table moltab /@dbalias myfile.rdf
```

SQL File Syntax

The `exportSDFFile` and `exportRDFFile` utilities accept a file of simple SQL commands and comment lines. A comment line begins with a semicolon. A SQL command ends with a semicolon.

Do not supply comments after the terminating semicolon for a SQL command. Supply comments only on comment lines, for example:

```
; this is a comment line. The next line is a SQL command.  
SELECT MOLFILE(CTAB) AS CTAB FROM SAMPLE2D;
```

The text lines below make up one SQL command.

```
SELECT  
  MOLFILE  
  (ctab) AS  
  CTAB  
FROM  
  SAMPLE2D;
```

Error Handling

In most cases, the `exportSDFFile` or `exportRDFFile` utility stops exporting if it encounters an Oracle error.

The following error codes are suppressed and ignored:

```
ora-00100 no data found  
ora-01403 no data found  
ora-12652 string truncated  
ora-24347 warning of a NULL column in an aggregate function
```

The following error codes are reported, and the current record might be omitted from the output, but the export will continue. These error codes represent common conditions, such as formatting errors.

```
ora-00001 unique constraint (schema.constraintname) violated  
ora-00054 resource busy and acquire with NOWAIT specified  
ora-00060 deadlock detected while waiting for resource  
ora-00910 specified length too long for its datatype  
ora-00912 input parameter too long  
ora-00972 identifier is too long  
ora-01400 cannot insert NULL into (column)  
ora-01426 numeric overflow  
ora-01555 snapshot too old  
ora-01704 string literal too long  
ora-01722 invalid number  
ora-01724 floating point precision is out of range  
ora-01841 (full) year must be between -4713 and +9999, and not be 0  
ora-01854 julian date must be between 1 and 5373484  
ora-01858 a non-numeric character was found where a numeric was expected  
ora-01859 a non-alphabetic character was found where an alphabetic was expected  
ora-01861 literal does not match format string  
ora-01862 the numeric value does not match the length of the format item  
ora-01863 the year is not supported for the current calendar  
ora-01864 the date is out of range for the current calendar  
ora-01865 not a valid era
```

ora-01866 the datetime class is invalid
ora-01867 the interval is invalid
ora-01868 the leading precision of the interval is too small
ora-01870 the intervals or datetimes are not mutually comparable
ora-01871 the number of seconds must be less than 60
ora-01873 the leading precision of the interval is too small
ora-01874 time zone hour must be between -12 and 14
ora-01875 time zone minute must be between -59 and 59
ora-01876 year must be at least -4713
ora-01878 specified field not found in datetime or interval
ora-01879 the hh25 field must be between 0 and 24
ora-01880 the fractional seconds must be between 0 and 999999999
ora-01881 timezone region id is invalidora-02290 check constraint violated
ora-02291 integrity constraint
ora-20100 errors or warnings from BIOVIA Direct
ora-29875 error during ODICIndexIndexInsert

The Direct error stack is displayed after any error.

The Direct error stack cannot be displayed unless the target schema has the Direct synonyms defined and has rights to run the Direct cartridge. If the Direct cartridge is not installed, the exportSDFFile or exportRDFFile program are not able to format binary CTABs. In this case, the program is only capable of exporting columns already formatted as text.

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
1	H	1	1.0078250321	99.9885	1e+099
1	D	2	2.0141017780	0.0115	1e+099
1	T	3	3.0160492675		388781329.3056
1	H	4	4.02783		1.39e-022
1	H	5	5.03954		9.1e-022
1	H	6	6.04494		2.9e-022
2	He	3	3.0160293097	0.000137	1e+099
2	He	4	4.0026032497	99.999863	1e+099
2	He	5	5.012220		7.e-022
2	He	6	6.0188881		0.8067
2	He	7	7.028030		2.9e-021
2	He	8	8.033922		0.119
2	He	9	9.043820		7.e-021
2	He	10	10.052400		2.7e-021
3	Li	4	4.02718		9.1e-023
3	Li	5	5.012540		3.7e-022
3	Li	6	6.0151223	7.59	1e+099
3	Li	7	7.0160040	92.41	1e+099
3	Li	8	8.0224867		0.8403
3	Li	9	9.0267891		0.1783
3	Li	10	10.035481		2.e-021
3	Li	11	11.043796		8.75e-003
3	Li	12	12.05378		1.e-008
4	Be	5	5.04079		1.e-099
4	Be	6	6.019726		5.e-021
4	Be	7	7.0169292		4598208
4	Be	8	8.00530509		6.7e-017
4	Be	9	9.0121821	100	1e+099
4	Be	10	10.0135337		47650958260000
4	Be	11	11.021658		13.81
4	Be	12	12.026921		2.15e-002

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
4	Be	13	13.03613		5.e-010
4	Be	14	14.04282		4.35e-003
5	B	7	7.029920		3.5e-022
5	B	8	8.0246067		0.77
5	B	9	8.0246067		0.77
5	B	10	10.0129370	19.9	1e+099
5	B	11	11.0093055	80.1	1e+099
5	B	12	12.0143521		2.02e-002
5	B	13	13.0177803		1.733e-002
5	B	14	14.025404		1.25e-002
5	B	15	15.031097		9.87e-003
5	B	16	16.039810		1.9e-010
5	B	17	17.04693		5.08e-003
5	B	18	18.05617		2.6e-008
5	B	19	19.06373		2.92e-003
6	C	8	8.037675		2.e-021
6	C	9			
6	C	10	10.0168531		19.290
6	C	11	11.0114338		1223.4
6	C	12	12.0000000	98.93	1e+099
6	C	13	13.0033548378	1.07	1e+099
6	C	14	14.003241988		179874478200
6	C	15	15.0105993		2.449
6	C	16	16.014701		0.747
6	C	17	17.022584		0.193
6	C	18	18.026760		9.2e-002
6	C	19	19.03525		4.62e-002
6	C	20	20.04032		1.6e-002
6	C	21	21.04934		3.e-008
6	C	22	22.05645		6.2e-003
7	N	10	10.04262		2.e-022
7	N	11	11.02680		5.9e-022
7	N	12	12.0186132		1.1e-002

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
7	N	13	13.00573858		597.9
7	N	14	14.0030740052	99.632	1e+099
7	N	15	15.0001088984	0.368	1e+099
7	N	16	16.0061014		7.13
7	N	17	17.008450		4.173
7	N	18	18.014082		0.622
7	N	19	19.017027		0.271
7	N	20	20.023370		0.13
7	N	21	21.02709		8.7e-002
7	N	22	22.03444		1.39e-002
7	N	23	23.04051		1.45e-002
7	N	24	24.05050		5.2e-008
8	O	12	12.034405		5.8e-022
8	O	13	13.024810		8.58e-003
8	O	14	14.00859529		70.598
8	O	15	15.0030654		122.24
8	O	16	15.9949146221	99.757	1e+099
8	O	17	16.99913150	0.038	1e+099
8	O	18	17.9991604	0.205	1e+099
8	O	19	19.003579		26.464
8	O	20	20.0040762		13.51
8	O	21	21.008655		3.42
8	O	22	22.009970		2.25
8	O	23	23.01569		9.e-002
8	O	24	24.02037		6.5e-002
8	O	25	25.02914		5.e-008
8	O	26	26.03775		4.e-008
9	F	14	14.03608		1.e-099
9	F	15	15.01801		4.1e-022
9	F	16	16.011466		1.1e-020
9	F	17	17.00209524		64.49
9	F	18	18.0009377		6586.26
9	F	19	18.99840320	100	1e+099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
9	F	20	19.99998132		11.163
9	F	21	20.9999489		4.158
9	F	22	22.002999		4.23
9	F	23	23.003570		2.23
9	F	24	24.008100		0.4
9	F	25	25.012090		5.e-002
9	F	26	26.01963		1.02e-002
9	F	27	27.02689		4.9e-003
9	F	28	28.03567		4.e-008
9	F	29	29.04326		2.6e-003
10	Ne	16	16.025757		9.e-021
10	Ne	17	17.017700		0.1092
10	Ne	18	18.0056971		1.672
10	Ne	19	19.0018798		17.296
10	Ne	20	19.9924401759	90.48	1e+099
10	Ne	21	20.99384674	0.27	1e+099
10	Ne	22	21.99138551	9.25	1e+099
10	Ne	23	22.99446734		37.24
10	Ne	24	23.993615		202.8
10	Ne	25	24.997790		0.602
10	Ne	26	26.000460		0.197
10	Ne	27	27.00762		3.2e-002
10	Ne	28	28.01211		1.83e-002
10	Ne	29	29.01935		1.56e-002
10	Ne	30	30.02387		5.8e-003
10	Ne	31	31.03311		3.4e-003
10	Ne	32	32.03991		3.5e-003
11	Na	18	18.02718		1.3e-021
11	Na	19	19.013879		4.e-008
11	Na	20	20.007348		0.4479
11	Na	21	20.9976551		22.49
11	Na	22	21.9944368		82107965.967552
11	Na	23	22.98976967	100	1e+099

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
11	Na	24	23.99096333		53852.4
11	Na	25	24.9899544		59.1
11	Na	26	25.992590		1.077
11	Na	27	26.994010		0.301
11	Na	28	27.998890		3.05e-002
11	Na	29	29.00281		4.49e-002
11	Na	30	30.00923		4.84e-002
11	Na	31	31.01360		1.7e-002
11	Na	32	32.01965		1.29e-002
11	Na	33	33.02739		8.2e-003
11	Na	34	34.03490		5.5e-003
11	Na	35	35.04418		1.5e-003
12	Mg	20	20.018863		9.e-002
12	Mg	21	21.011714		0.122
12	Mg	22	21.9995741		3.857
12	Mg	23	22.9941249		11.317
12	Mg	24	23.98504190	78.99	1e+099
12	Mg	25	24.98583702	10.00	1e+099
12	Mg	26	25.98259304	11.01	1e+099
12	Mg	27	26.98434074		567.48
12	Mg	28	27.9838767		75294
12	Mg	29	28.988550		1.30
12	Mg	30	29.990460		0.335
12	Mg	31	30.996550		0.23
12	Mg	32	31.99915		9.5e-002
12	Mg	33	33.00559		9.05e-002
12	Mg	34	34.00907		2.e-002
12	Mg	35	35.01749		7.e-002
12	Mg	36	36.02245		1.e-099
12	Mg	37			
13	Al	21	21.02804		3.5e-008
13	Al	22	22.01952		5.9e-002
13	Al	23	23.007265		0.47

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
13	Al	24	23.999941		2.053
13	Al	25	24.9904286		7.183
13	Al	26	25.98689166		22626315942000
13	Al	27	26.98153844	100	1e+099
13	Al	28	27.98191018		134.484
13	Al	29	28.9804448		393.6
13	Al	30	29.982960		3.60
13	Al	31	30.983946		0.644
13	Al	32	31.988120		3.17e-002
13	Al	33	32.990870		4.17e-002
13	Al	34	33.99693		5.63e-002
13	Al	35	34.99994		3.86e-002
13	Al	36	36.00635		9.e-002
13	Al	37	37.01031		1.e-099
13	Al	38	38.01690		1.e-099
13	Al	39	39.02190		1.e-099
14	Si	22	22.03453		2.9e-002
14	Si	23	23.02552		4.23e-002
14	Si	24	24.011546		0.14
14	Si	25	25.004107		0.22
14	Si	26	25.992330		2.234
14	Si	27	26.98670476		4.16
14	Si	28	27.9769265327	92.2297	1e+099
14	Si	29	28.97649472	4.6832	1e+099
14	Si	30	29.97377022	3.0872	1e+099
14	Si	31	30.97536327		9438
14	Si	32	31.9741481		4165514242.56
14	Si	33	32.978001		6.18
14	Si	34	33.978576		2.77
14	Si	35	34.984580		0.78
14	Si	36	35.98669		0.45
14	Si	37	36.99300		9.e-002
14	Si	38	37.99598		1.e-099

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
14	Si	39	39.00230		1.e-099
14	Si	40	40.00580		1.e-099
14	Si	41	41.01270		1.e-099
14	Si	42	42.01610		1.e-099
15	P	24	24.03435		1.e-099
15	P	25	25.02026		3.e-008
15	P	26	26.01178		3.e-002
15	P	27	26.999190		0.26
15	P	28	27.992312		0.2703
15	P	29	28.9818014		4.142
15	P	30	29.9783138		149.88
15	P	31	30.97376151	100	1e+099
15	P	32	31.97390716		1232323.2
15	P	33	32.9717253		2189376
15	P	34	33.973636		12.43
15	P	35	34.9733142		47.3
15	P	36	35.978260		5.6
15	P	37	36.979610		2.31
15	P	38	37.98447		0.64
15	P	39	38.98642		0.19
15	P	40	39.99105		0.153
15	P	41	40.99480		0.15
15	P	42	42.00009		0.12
15	P	43	43.00331		3.3e-002
15	P	44	44.00988		1.e-099
15	P	45	45.01514		1.e-099
15	P	46	46.02383		1.e-099
16	S	26	26.02788		1.e-099
16	S	27	27.01880		2.1e-002
16	S	28	28.00437		0.125
16	S	29	28.996610		0.187
16	S	30	29.984903		1.178
16	S	31	30.9795544		2.572

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
16	S	32	31.97207069	94.93	1e+099
16	S	33	32.97145850	0.76	1e+099
16	S	34	33.96786683	4.29	1e+099
16	S	35	34.96903214		7560864
16	S	36	35.96708088	0.02	1e+099
16	S	37	36.97112572		303
16	S	38	37.971163		10218
16	S	39	38.975140		11.5
16	S	40	39.97547		8.8
16	S	41	40.98003		1.99
16	S	42	41.98149		1.013
16	S	43	42.98660		0.26
16	S	44	43.98832		0.123
16	S	45	44.99482		8.2e-002
16	S	46	45.99957		1.e-099
16	S	47	47.00762		1.e-099
16	S	48	48.01299		1.e-099
16	S	49	49.02201		2.e-007
17	Cl	28	28.02851		1.e-099
17	Cl	29	29.01411		2.e-008
17	Cl	30	30.00477		3.e-008
17	Cl	31	30.992420		0.15
17	Cl	32	31.985689		0.298
17	Cl	33	32.9774518		2.511
17	Cl	34	33.97376197		1.5264
17	Cl	35	34.96885271	75.78	1e+099
17	Cl	36	35.96830695		9498634726000
17	Cl	37	36.96590260	24.22	1e+099
17	Cl	38	37.96801055		2234.4
17	Cl	39	38.9680077		3336
17	Cl	40	39.970420		81
17	Cl	41	40.970650		38.4
17	Cl	42	41.97317		6.8

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
17	Cl	43	42.97420		3.07
17	Cl	44	43.97854		0.56
17	Cl	45	44.97970		0.4
17	Cl	46	45.98412		0.22
17	Cl	47	46.98795		1.e-099
17	Cl	48	47.99485		1.e-099
17	Cl	49	48.99989		1.e-099
17	Cl	50	50.00773		1.e-099
17	Cl	51	51.01353		1.e-099
18	Ar	30	30.02156		2.e-008
18	Ar	31	31.01213		1.44e-002
18	Ar	32	31.997660		9.8e-002
18	Ar	33	32.989930		0.173
18	Ar	34	33.980270		0.845
18	Ar	35	34.9752567		1.775
18	Ar	36	35.96754628	0.3365	1e+099
18	Ar	37	36.9667759		3027456
18	Ar	38	37.9627322	0.0632	1e+099
18	Ar	39	38.964313		8488813115.52
18	Ar	40	39.962383123	99.6003	1e+099
18	Ar	41	40.9645008		6576.6
18	Ar	42	41.963050		1038222868.032
18	Ar	43	42.965670		322.2
18	Ar	44	43.965365		712.2
18	Ar	45	44.968090		21.48
18	Ar	46	45.968090		8.4
18	Ar	47	46.97219		0.58
18	Ar	48	47.97507		1.e-099
18	Ar	49	48.98218		0.17
18	Ar	50	49.98594		8.5e-002
18	Ar	51	50.99324		1.e-099
18	Ar	52	51.99817		1.e-099
18	Ar	53	53.00623		1.e-099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
19	K	32	32.02192		1.e-099
19	K	33	33.00726		2.5e-008
19	K	34	33.99841		4.e-008
19	K	35	34.988012		0.178
19	K	36	35.981293		0.342
19	K	37	36.97337691		1.226
19	K	38	37.9690801		458.16
19	K	39	38.9637069	93.2581	1e+099
19	K	40	39.96399867	0.0117	3.9477714426e+016
19	K	41	40.96182597	6.7302	1e+099
19	K	42	41.9624031		44496
19	K	43	42.960716		80280
19	K	44	43.961560		1327.8
19	K	45	44.960700		1038
19	K	46	45.961976		105
19	K	47	46.961678		17.50
19	K	48	47.965513		6.8
19	K	49	48.967450		1.26
19	K	50	49.97278		0.472
19	K	51	50.97638		0.365
19	K	52	51.98261		0.105
19	K	53	52.98712		3.e-002
19	K	54	53.99399		1.e-002
19	K	55	54.99939		1.e-099
20	Ca	34	34.01412		3.5e-008
20	Ca	35	35.004770		2.57e-002
20	Ca	36	35.993090		0.102
20	Ca	37	36.985872		0.1811
20	Ca	38	37.976319		0.44
20	Ca	39	38.9707177		0.8596
20	Ca	40	39.9625912	96.941	1e+099
20	Ca	41	40.9622783		3218806452000
20	Ca	42	41.9586183	0.647	1e+099

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
20	Ca	43	42.9587668	0.135	1e+099
20	Ca	44	43.9554811	2.086	1e+099
20	Ca	45	44.9561859		14054688
20	Ca	46	45.9536928	0.004	1e+099
20	Ca	47	46.9545465		391910.4
20	Ca	48	47.952534	0.187	1.672517078e+027
20	Ca	49	48.955673		523.08
20	Ca	50	49.957518		13.9
20	Ca	51	50.96147		10.0
20	Ca	52	51.96510		4.6
20	Ca	53	52.97005		9.e-002
20	Ca	54	53.97468		1.e-099
20	Ca	55	54.98055		1.e-099
20	Ca	56	55.98579		1.e-099
20	Ca	57	56.99236		1.e-099
21	Sc	36	36.01492		1.e-099
21	Sc	37	37.00305		1.e-099
21	Sc	38	37.99470		3.e-007
21	Sc	39	38.984790		3.e-007
21	Sc	40	39.977964		0.1823
21	Sc	41	40.9692513		0.5963
21	Sc	42	41.9655168		0.6813
21	Sc	43	42.9611510		14007.6
21	Sc	44	43.9594030		14292
21	Sc	45	44.9559102	100	1e+099
21	Sc	46	45.9551703		7239456
21	Sc	47	46.9524080		289370.88
21	Sc	48	47.952235		157212
21	Sc	49	48.950024		3432
21	Sc	50	49.952187		102.5
21	Sc	51	50.953603		12.4
21	Sc	52	51.95665		8.2
21	Sc	53	52.95924		3

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
21	Sc	54	53.96300		0.26
21	Sc	55	54.96743		0.12
21	Sc	56	55.97266		1.e-099
21	Sc	57	56.97704		1.3e-002
21	Sc	58	57.98307		1.2e-002
21	Sc	59	58.98804		1.e-099
22	Ti	38	38.00977		1.2e-007
22	Ti	39	39.00132		3.1e-002
22	Ti	40	39.99050		5.33e-002
22	Ti	41	40.983130		8.09e-002
22	Ti	42	41.973032		0.199
22	Ti	43	42.968523		0.509
22	Ti	44	43.9596902		1893415564.8
22	Ti	45	44.9581243		11088
22	Ti	46	45.9526295	8.25	1e+099
22	Ti	47	46.9517638	7.44	1e+099
22	Ti	48	47.9479471	73.72	1e+099
22	Ti	49	48.9478708	5.41	1e+099
22	Ti	50	49.9447921	5.18	1e+099
22	Ti	51	50.9466160		345.6
22	Ti	52	51.946898		102
22	Ti	53	52.94973		32.7
22	Ti	54	53.95087		1.5
22	Ti	55	54.95512		0.49
22	Ti	56	55.95799		0.164
22	Ti	57	56.96290		6.e-002
22	Ti	58	57.96611		5.4e-002
22	Ti	59	58.97196		3.e-002
22	Ti	60	59.97564		2.2e-002
22	Ti	61	60.98202		1.e-099
23	V	40	40.01109		1.e-099
23	V	41	40.99974		1.e-099
23	V	42	41.99123		5.5e-008

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
23	V	43	42.98065		1.e-099
23	V	44	43.974400		0.111
23	V	45	44.965782		0.547
23	V	46	45.9601995		0.4225
23	V	47	46.9549069		1956
23	V	48	47.9522545		1380110.4
23	V	49	48.9485169		28512000
23	V	50	49.9471628	0.250	4.7335389e+024
23	V	51	50.9439637	99.750	1e+099
23	V	52	51.9447797		224.58
23	V	53	52.944343		96
23	V	54	53.946444		49.8
23	V	55	54.94724		6.54
23	V	56	55.95036		0.216
23	V	57	56.95236		0.35
23	V	58	57.95665		0.191
23	V	59	58.95930		7.5e-002
23	V	60	59.96450		0.122
23	V	61	60.96741		4.7e-002
23	V	62	61.97314		3.35e-002
23	V	63	62.97675		1.7e-002
24	Cr	42	42.00643		1.4e-002
24	Cr	43	42.997710		2.16e-002
24	Cr	44	43.98547		5.4e-002
24	Cr	45	44.97916		5.e-002
24	Cr	46	45.968362		0.26
24	Cr	47	46.962907		0.5
24	Cr	48	47.954036		77616
24	Cr	49	48.9513411		2538
24	Cr	50	49.9460496	4.345	1e+099
24	Cr	51	50.9447718		2393496
24	Cr	52	51.9405119	83.789	1e+099
24	Cr	53	52.9406538	9.501	1e+099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
24	Cr	54	53.9388849	2.365	1e+099
24	Cr	55	54.9408442		209.82
24	Cr	56	55.940645		356.4
24	Cr	57	56.94375		21.1
24	Cr	58	57.94425		7.0
24	Cr	59	58.94863		0.46
24	Cr	60	59.94973		0.56
24	Cr	61	60.95409		0.261
24	Cr	62	61.95580		0.199
24	Cr	63	62.96186		0.129
24	Cr	64	63.96420		4.3e-002
24	Cr	65	64.97037		2.7e-002
25	Mn	44	44.00687		1.05e-007
25	Mn	45	44.99451		7.e-008
25	Mn	46	45.98672		3.7e-002
25	Mn	47	46.97610		0.1
25	Mn	48	47.968870		0.1581
25	Mn	49	48.959623		0.382
25	Mn	50	49.9542440		0.2839
25	Mn	51	50.9482155		2772
25	Mn	52	51.9455701		483062.4
25	Mn	53	52.9412947		116760626200000
25	Mn	54	53.9403632		26959392
25	Mn	55	54.9380496	100	1e+099
25	Mn	56	55.9389094		9284.04
25	Mn	57	56.938287		85.4
25	Mn	58	57.939990		3.0
25	Mn	59	58.940450		4.59
25	Mn	60	59.94319		51
25	Mn	61	60.94446		0.67
25	Mn	62	61.94797		0.671
25	Mn	63	62.94981		0.275
25	Mn	64	63.95373		8.88e-002

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
25	Mn	65	64.95610		9.2e-002
25	Mn	66	65.96082		6.44e-002
25	Mn	67	66.96382		4.5e-002
26	Fe	45	45.01456		4.9e-003
26	Fe	46	46.00081		9.e-003
26	Fe	47	46.99289		2.18e-002
26	Fe	48	47.98056		4.4e-002
26	Fe	49	48.97361		7.e-002
26	Fe	50	49.962990		0.155
26	Fe	51	50.956825		0.305
26	Fe	52	51.948117		29790
26	Fe	53	52.9453123		510.6
26	Fe	54	53.9396148	5.845	1e+099
26	Fe	55	54.9382980		86371306.68096
26	Fe	56	55.9349421	91.754	1e+099
26	Fe	57	56.9353987	2.119	1e+099
26	Fe	58	57.9332805	0.282	1e+099
26	Fe	59	58.9348805		3844368
26	Fe	60	59.934077		47335389000000
26	Fe	61	60.936749		358.8
26	Fe	62	61.936770		68
26	Fe	63	62.94012		6.1
26	Fe	64	63.94087		2.0
26	Fe	65	64.94494		1.3
26	Fe	66	65.94598		0.44
26	Fe	67	66.95000		0.394
26	Fe	68	67.95251		0.187
26	Fe	69	68.95770		0.109
27	Co	48	48.00176		1.e-099
27	Co	49	48.98972		3.5e-008
27	Co	50	49.98154		4.4e-002
27	Co	51	50.97072		1.e-099
27	Co	52	51.963590		0.115

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
27	Co	53	52.954225		0.242
27	Co	54	53.9484641		0.19323
27	Co	55	54.9420031		63108
27	Co	56	55.9398439		6672672
27	Co	57	56.9362962		23478336
27	Co	58	57.9357576		6122304
27	Co	59	58.9332002	100	1e+099
27	Co	60	59.9338222		166346024.445504
27	Co	61	60.9324794		5940
27	Co	62	61.934054		90
27	Co	63	62.933615		26.9
27	Co	64	63.935814		0.3
27	Co	65	64.936485		1.20
27	Co	66	65.93983		0.194
27	Co	67	66.94061		0.425
27	Co	68	67.94436		0.2
27	Co	69	68.94520		0.227
27	Co	70	69.94981		0.125
27	Co	71	70.95173		9.7e-002
27	Co	72	71.95641		9.e-002
28	Ni	50	49.99593		9.1e-003
28	Ni	51	50.98772		1.e-099
28	Ni	52	51.975680		3.8e-002
28	Ni	53	52.96846		4.5e-002
28	Ni	54	53.957910		0.104
28	Ni	55	54.951336		0.2047
28	Ni	56	55.942136		524880
28	Ni	57	56.939800		128160
28	Ni	58	57.9353479	68.0769	1e+099
28	Ni	59	58.9343516		3187249526000
28	Ni	60	59.9307906	26.2231	1e+099
28	Ni	61	60.9310604	1.1399	1e+099
28	Ni	62	61.9283488	3.6345	1e+099

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
28	Ni	63	62.9296729		3158848300.608
28	Ni	64	63.9279696	0.9256	1e+099
28	Ni	65	64.9300880		9061.92
28	Ni	66	65.929115		196560
28	Ni	67	66.931570		21
28	Ni	68	67.931845		29
28	Ni	69	68.93518		11.5
28	Ni	70	69.93614		6.0
28	Ni	71	70.94000		2.56
28	Ni	72	71.94130		1.57
28	Ni	73	72.94608		0.84
28	Ni	74	73.94791		0.68
28	Ni	75	74.95297		0.6
28	Ni	76	75.95533		0.47
28	Ni	77	76.96083		1.e-099
28	Ni	78	77.96380		1.e-099
29	Cu	52	51.99718		1.e-099
29	Cu	53	52.98555		3.e-007
29	Cu	54	53.97671		7.5e-008
29	Cu	55	54.96605		1.e-099
29	Cu	56	55.95856		9.3e-002
29	Cu	57	56.949216		0.1963
29	Cu	58	57.9445407		3.204
29	Cu	59	58.9395041		81.5
29	Cu	60	59.9373681		1422
29	Cu	61	60.9334622		11998.8
29	Cu	62	61.932587		580.38
29	Cu	63	62.9296011	69.17	1e+099
29	Cu	64	63.9297679		45720
29	Cu	65	64.9277937	30.83	1e+099
29	Cu	66	65.9288730		307.2
29	Cu	67	66.927750		222588
29	Cu	68	67.929640		31.1

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
29	Cu	69	68.929425		171
29	Cu	70	69.932409		44.5
29	Cu	71	70.932620		19.4
29	Cu	72	71.93552		6.6
29	Cu	73	72.93649		4.2
29	Cu	74	73.94020		1.594
29	Cu	75	74.94170		1.224
29	Cu	76	75.94599		0.641
29	Cu	77	76.94795		0.469
29	Cu	78	77.95281		0.342
29	Cu	79	78.95528		0.188
29	Cu	80	79.96189		1.e-099
30	Zn	54	53.99295		1.e-099
30	Zn	55	54.98398		1.e-099
30	Zn	56	55.97238		3.6e-002
30	Zn	57	56.96491		3.8e-002
30	Zn	58	57.954600		8.4e-002
30	Zn	59	58.949270		0.182
30	Zn	60	59.941832		142.8
30	Zn	61	60.939514		89.1
30	Zn	62	61.934334		33069.6
30	Zn	63	62.9332156		2308.2
30	Zn	64	63.9291466	48.63	1e+099
30	Zn	65	64.9292451		21086784
30	Zn	66	65.9260368	27.90	1e+099
30	Zn	67	66.9271309	4.10	1e+099
30	Zn	68	67.9248476	18.75	1e+099
30	Zn	69	68.9265535		3384
30	Zn	70	69.925325	0.62	1e+099
30	Zn	71	70.927727		147
30	Zn	72	71.926861		167400
30	Zn	73	72.929780		23.5
30	Zn	74	73.929460		95.6

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
30	Zn	75	74.932940		10.2
30	Zn	76	75.93339		5.7
30	Zn	77	76.93709		2.08
30	Zn	78	77.93857		1.47
30	Zn	79	78.94268		0.995
30	Zn	80	79.94441		0.545
30	Zn	81	80.95048		0.29
30	Zn	82	81.95484		1.e-099
31	Ga	56	55.99491		1.e-099
31	Ga	57	56.98293		1.e-099
31	Ga	58	57.97425		1.e-099
31	Ga	59	58.96337		1.e-099
31	Ga	60	59.95706		7.e-002
31	Ga	61	60.94917		0.168
31	Ga	62	61.944180		0.11599
31	Ga	63	62.93914		32.4
31	Ga	64	63.936838		157.62
31	Ga	65	64.9327393		912
31	Ga	66	65.931592		34164
31	Ga	67	66.9282049		281767.68
31	Ga	68	67.9279835		4062.6
31	Ga	69	68.925581	60.108	1e+099
31	Ga	70	69.926028		1268.4
31	Ga	71	70.9247050	39.892	1e+099
31	Ga	72	71.9263694		50760
31	Ga	73	72.925170		17496
31	Ga	74	73.926940		487.2
31	Ga	75	74.926501		126
31	Ga	76	75.92893		32.6
31	Ga	77	76.929280		13.2
31	Ga	78	77.931660		5.09
31	Ga	79	78.93292		2.847
31	Ga	80	79.93659		1.697

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
31	Ga	81	80.93775		1.217
31	Ga	82	81.94316		0.599
31	Ga	83	82.94687		0.308
31	Ga	84	83.95234		8.5e-002
32	Ge	58	57.99101		1.e-099
32	Ge	59	58.98175		1.e-099
32	Ge	60	59.97019		1.e-099
32	Ge	61	60.96379		3.9e-002
32	Ge	62	61.95465		0.13
32	Ge	63	62.94964		0.142
32	Ge	64	63.94157		63.7
32	Ge	65	64.93944		30.9
32	Ge	66	65.933850		8136
32	Ge	67	66.932738		1134
32	Ge	68	67.928097		23410080
32	Ge	69	68.927972		140580
32	Ge	70	69.9242504	20.84	1e+099
32	Ge	71	70.9249540		987552
32	Ge	72	71.9220762	27.54	1e+099
32	Ge	73	72.9234594	7.73	1e+099
32	Ge	74	73.9211782	36.28	1e+099
32	Ge	75	74.9228595		4966.8
32	Ge	76	75.9214027	7.61	4.985994308e+028
32	Ge	77	76.9235485		40680
32	Ge	78	77.922853		5280
32	Ge	79	78.92540		18.98
32	Ge	80	79.925445		29.5
32	Ge	81	80.92882		8
32	Ge	82	81.92955		4.55
32	Ge	83	82.93451		1.85
32	Ge	84	83.93731		0.954
32	Ge	85	84.94269		0.54
32	Ge	86	85.94627		1.e-099

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
33	As	60	59.99313		6.e-017
33	As	61	60.98062		6.1e-017
33	As	62	61.97320		6.2e-017
33	As	63	62.96369		6.3e-017
33	As	64	63.95757		4.e-002
33	As	65	64.94948		0.17
33	As	66	65.94437		9.577e-002
33	As	67	66.93919		42.5
33	As	68	67.93679		151.6
33	As	69	68.932280		912
33	As	70	69.930930		3156
33	As	71	70.927115		235008
33	As	72	71.926753		93600
33	As	73	72.923825		6937920
33	As	74	73.9239291		1535328
33	As	75	74.9215964	100	1e+099
33	As	76	75.9223939		93121.92
33	As	77	76.9206477		139788
33	As	78	77.921829		5442
33	As	79	78.920948		540.6
33	As	80	79.922578		15.2
33	As	81	80.922133		33.3
33	As	82	81.92450		19.1
33	As	83	82.92498		13.4
33	As	84	83.92906		4.02
33	As	85	84.93181		2.021
33	As	86	85.93623		0.945
33	As	87	86.93958		0.61
33	As	88	87.94456		1.e-099
33	As	89	88.94923		1.e-099
34	Se	65	64.96466		5.e-002
34	Se	66	65.95521		3.3e-002
34	Se	67	66.95009		0.133

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
34	Se	68	67.94187		35.5
34	Se	69	68.939560		27.4
34	Se	70	69.93350		2466
34	Se	71	70.93227		284.4
34	Se	72	71.927112		725760
34	Se	73	72.926767		25740
34	Se	74	73.9224766	0.89	1e+099
34	Se	75	74.9225236		10348905.6
34	Se	76	75.9192141	9.37	1e+099
34	Se	77	76.9199146	7.63	1e+099
34	Se	78	77.9173095	23.77	1e+099
34	Se	79	78.9184998		9309293170000
34	Se	80	79.9165218	49.61	1e+099
34	Se	81	80.9179929		1107
34	Se	82	81.9167000	8.73	3.061021822e+027
34	Se	83	82.919119		1338
34	Se	84	83.918465		186
34	Se	85	84.922240		31.7
34	Se	86	85.924271		15.3
34	Se	87	86.928520		5.50
34	Se	88	87.931420		1.53
34	Se	89	88.93602		0.41
34	Se	90	89.93942		1.e-099
34	Se	91	90.94537		0.27
34	Se	92	91.94933		1.e-099
35	Br	67	66.96479		1.e-099
35	Br	68	67.95825		1.5e-006
35	Br	69	68.95018		2.4e-008
35	Br	70	69.94462		7.91e-002
35	Br	71	70.93925		21.4
35	Br	72	71.93650		78.6
35	Br	73	72.93179		204
35	Br	74	73.929891		1524

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
35	Br	75	74.925776		5802
35	Br	76	75.924542		58320
35	Br	77	76.921380		205329.6
35	Br	78	77.921146		387.6
35	Br	79	78.9183376	50.69	1e+099
35	Br	80	79.9185300		1060.8
35	Br	81	80.916291	49.31	1e+099
35	Br	82	81.916805		127015.2
35	Br	83	82.915180		8640
35	Br	84	83.916504		1908
35	Br	85	84.915608		174
35	Br	86	85.918797		55.1
35	Br	87	86.920711		55.65
35	Br	88	87.924070		16.36
35	Br	89	88.926390		4.40
35	Br	90	89.930630		1.910
35	Br	91	90.933970		0.541
35	Br	92	91.939260		0.343
35	Br	93	92.94310		0.102
35	Br	94	93.94868		7.e-002
36	Kr	69	68.96532		3.2e-002
36	Kr	70	69.95601		5.7e-002
36	Kr	71	70.95051		0.1
36	Kr	72	71.94191		17.16
36	Kr	73	72.93893		28.6
36	Kr	74	73.933260		690
36	Kr	75	74.931034		257.4
36	Kr	76	75.925948		53280
36	Kr	77	76.924668		4464
36	Kr	78	77.920386	0.35	1e+099
36	Kr	79	78.920083		126144
36	Kr	80	79.916378	2.28	1e+099
36	Kr	81	80.916592		7226536054000

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
36	Kr	82	81.9134846	11.58	1e+099
36	Kr	83	82.914136	11.49	1e+099
36	Kr	84	83.911507	57.00	1e+099
36	Kr	85	84.912527		340057435.43808
36	Kr	86	85.9106103	17.30	1e+099
36	Kr	87	86.9133543		4578
36	Kr	88	87.914447		10224
36	Kr	89	88.917630		189
36	Kr	90	89.919524		32.32
36	Kr	91	90.923440		8.57
36	Kr	92	91.926153		1.840
36	Kr	93	92.93127		1.286
36	Kr	94	93.93436		0.21
36	Kr	95	94.93984		0.114
36	Kr	96	95.94307		8.e-002
36	Kr	97	96.94856		6.3e-002
37	Rb	71	70.96532		1.e-099
37	Rb	72	71.95908		1.5e-006
37	Rb	73	72.95037		3.e-008
37	Rb	74	73.94447		6.476e-002
37	Rb	75	74.938569		19.0
37	Rb	76	75.935071		36.5
37	Rb	77	76.930407		226.2
37	Rb	78	77.928141		1059.6
37	Rb	79	78.923997		1374
37	Rb	80	79.922519		33.4
37	Rb	81	80.918994		16473.6
37	Rb	82	81.918208		76.38
37	Rb	83	82.915112		7447680
37	Rb	84	83.914385		2831328
37	Rb	85	84.9117893	72.17	1e+099
37	Rb	86	85.9111671		1610668.8
37	Rb	87	86.9091835	27.83	1.55354746698e+018

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
37	Rb	88	87.911319		1066.8
37	Rb	89	88.912280		909
37	Rb	90	89.914809		158
37	Rb	91	90.916534		58.4
37	Rb	92	91.919725		4.492
37	Rb	93	92.922033		5.84
37	Rb	94	93.926407		2.702
37	Rb	95	94.929319		0.3775
37	Rb	96	95.934284		0.203
37	Rb	97	96.937340		0.1699
37	Rb	98	97.941700		0.114
37	Rb	99	98.94542		5.03e-002
37	Rb	100	99.94987		5.1e-002
37	Rb	101	100.95320		3.2e-002
37	Rb	102	101.95921		3.7e-002
38	Sr	73	72.96597		2.5e-002
38	Sr	74	73.95631		1.e-099
38	Sr	75	74.94992		8.8e-002
38	Sr	76	75.94161		8.9
38	Sr	77	76.93776		9.0
38	Sr	78	77.932179		159
38	Sr	79	78.929707		135
38	Sr	80	79.924525		6378
38	Sr	81	80.923213		1338
38	Sr	82	81.918401		2191104
38	Sr	83	82.917555		116676
38	Sr	84	83.913425	0.56	1e+099
38	Sr	85	84.912933		5603299.2
38	Sr	86	85.9092624	9.86	1e+099
38	Sr	87	86.9088793	7.00	1e+099
38	Sr	88	87.9056143	82.58	1e+099
38	Sr	89	88.9074529		4365792
38	Sr	90	89.9077376		908523901.8432

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
38	Sr	91	90.910210		34668
38	Sr	92	91.911030		9576
38	Sr	93	92.914022		445.38
38	Sr	94	93.915360		75.3
38	Sr	95	94.919358		23.90
38	Sr	96	95.921680		1.07
38	Sr	97	96.926149		0.429
38	Sr	98	97.928471		0.653
38	Sr	99	98.93332		0.269
38	Sr	100	99.93535		0.202
38	Sr	101	100.94052		0.118
38	Sr	102	101.94302		6.9e-002
38	Sr	103	102.94895		1.e-099
38	Sr	104	103.95233		1.e-099
39	Y	77	76.94962		6.3e-002
39	Y	78	77.94350		5.4e-002
39	Y	79	78.93735		14.8
39	Y	80	79.93434		30.1
39	Y	81	80.929130		70.4
39	Y	82	81.92679		8.30
39	Y	83	82.922350		424.8
39	Y	84	83.92039		4.6
39	Y	85	84.916427		9648
39	Y	86	85.914888		53064
39	Y	87	86.9108778		287280
39	Y	88	87.9095034		9214560
39	Y	89	88.9058479	100	1e+099
39	Y	90	89.9071514		230400
39	Y	91	90.907303		5055264
39	Y	92	91.908947		12744
39	Y	93	92.909582		36648
39	Y	94	93.911594		1122
39	Y	95	94.912824		618

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
39	Y	96	95.915898		5.34
39	Y	97	96.918131		3.75
39	Y	98	97.922220		0.548
39	Y	99	98.924635		1.470
39	Y	100	99.927760		0.735
39	Y	101	100.93031		0.426
39	Y	102	101.933560		0.3
39	Y	103	102.93694		0.224
39	Y	104	103.94145		0.18
39	Y	105	104.94509		1.e-099
39	Y	106	105.95022		1.e-099
40	Zr	79	78.94916		5.6e-002
40	Zr	80	79.94055		4.6
40	Zr	81	80.93682		5.5
40	Zr	82	81.93109		32
40	Zr	83	82.92865		41.6
40	Zr	84	83.92325		1554
40	Zr	85	84.92147		471.6
40	Zr	86	85.916470		59400
40	Zr	87	86.914817		6048
40	Zr	88	87.910226		7205760
40	Zr	89	88.908889		282276
40	Zr	90	89.9047037	51.45	1e+099
40	Zr	91	90.9056450	11.22	1e+099
40	Zr	92	91.9050401	17.15	1e+099
40	Zr	93	92.9064756		48282096780000
40	Zr	94	93.9063158	17.38	1e+099
40	Zr	95	94.9080427		5532364.8
40	Zr	96	95.908276	2.80	7.57366224e+026
40	Zr	97	96.910951		60840
40	Zr	98	97.912746		30.7
40	Zr	99	98.916511		2.1
40	Zr	100	99.917760		7.1

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
40	Zr	101	100.921140		2.3
40	Zr	102	101.922980		2.9
40	Zr	103	102.92660		1.3
40	Zr	104	103.92878		1.2
40	Zr	105	104.93305		0.6
40	Zr	106	105.93591		1.e-099
40	Zr	107	106.94086		1.e-099
40	Zr	108	107.94428		1.e-099
41	Nb	81	80.94905		4.4e-008
41	Nb	82	81.94313		5.1e-002
41	Nb	83	82.93670		4.1
41	Nb	84	83.93357		9.8
41	Nb	85	84.92791		20.9
41	Nb	86	85.925040		88
41	Nb	87	86.920360		225
41	Nb	88	87.91796		870
41	Nb	89	88.913500		7308
41	Nb	90	89.911264		52560
41	Nb	91	90.906991		21458709734.4
41	Nb	92	91.9071932		1.0950253322e+015
41	Nb	93	92.9063775	100	1e+099
41	Nb	94	93.9072835		640605597800
41	Nb	95	94.9068352		3023222.4
41	Nb	96	95.908100		84060
41	Nb	97	96.9080971		4326
41	Nb	98	97.910331		2.86
41	Nb	99	98.911618		15.0
41	Nb	100	99.914181		1.5
41	Nb	101	100.915252		7.1
41	Nb	102	101.918040		1.3
41	Nb	103	102.919140		1.5
41	Nb	104	103.92246		4.9
41	Nb	105	104.92393		2.95

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
41	Nb	106	105.92819		0.92
41	Nb	107	106.93031		0.3
41	Nb	108	107.93501		0.193
41	Nb	109	108.93763		0.19
41	Nb	110	109.94268		0.17
42	Mo	83	82.94874		2.3e-002
42	Mo	84	83.94009		3.8e-003
42	Mo	85	84.93659		3.2
42	Mo	86	85.93070		19.6
42	Mo	87	86.92733		14.05
42	Mo	88	87.921953		480
42	Mo	89	88.919481		126.6
42	Mo	90	89.913936		20016
42	Mo	91	90.911751		929.4
42	Mo	92	91.906810	14.84	1e+099
42	Mo	93	92.906812		126227704000
42	Mo	94	93.9050876	9.25	1e+099
42	Mo	95	94.9058415	15.92	1e+099
42	Mo	96	95.9046789	16.68	1e+099
42	Mo	97	96.9060210	9.55	1e+099
42	Mo	98	97.9054078	24.13	1e+099
42	Mo	99	98.9077116		237384
42	Mo	100	99.907477	9.63	2.68233871e+026
42	Mo	101	100.910347		876.6
42	Mo	102	101.910297		678
42	Mo	103	102.913200		67.5
42	Mo	104	103.913760		60
42	Mo	105	104.916970		35.6
42	Mo	106	105.918134		8.73
42	Mo	107	106.92169		3.5
42	Mo	108	107.92358		1.09
42	Mo	109	108.92781		0.53
42	Mo	110	109.92973		0.3

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
42	Mo	111	110.93451		1.e-099
42	Mo	112	111.93684		1.e-099
42	Mo	113	112.94203		1.e-099
43	Tc	85	84.94894		1.1e-007
43	Tc	86	85.94288		5.5e-002
43	Tc	87	86.93653		2.18
43	Tc	88	87.93283		5.8
43	Tc	89	88.92754		12.8
43	Tc	90	89.92356		8.7
43	Tc	91	90.91843		188.4
43	Tc	92	91.915260		255
43	Tc	93	92.910248		9900
43	Tc	94	93.909656		17580
43	Tc	95	94.907656		72000
43	Tc	96	95.907871		369792
43	Tc	97	96.906365		82048007600000
43	Tc	98	97.907216		132539089200000
43	Tc	99	98.9062546		6661667078600
43	Tc	100	99.9076576		15.8
43	Tc	101	100.907314		853.2
43	Tc	102	101.909213		5.28
43	Tc	103	102.909179		54.2
43	Tc	104	103.911440		1098
43	Tc	105	104.911660		456
43	Tc	106	105.914355		35.6
43	Tc	107	106.91508		21.2
43	Tc	108	107.91848		5.17
43	Tc	109	108.91963		0.86
43	Tc	110	109.92339		0.92
43	Tc	111	110.92505		0.29
43	Tc	112	111.92924		0.29
43	Tc	113	112.93133		0.17
43	Tc	114	113.93588		0.15

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
43	Tc	115	114.93828		1.e-099
44	Ru	87	86.94918		1.e-099
44	Ru	88	87.94042		1.3
44	Ru	89	88.93611		1.38
44	Ru	90	89.92978		11
44	Ru	91	90.92638		9
44	Ru	92	91.92012		219
44	Ru	93	92.917050		59.7
44	Ru	94	93.911360		3108
44	Ru	95	94.910413		5914.8
44	Ru	96	95.907598	5.54	1e+099
44	Ru	97	96.907555		250560
44	Ru	98	97.905287	1.87	1e+099
44	Ru	99	98.9059393	12.76	1e+099
44	Ru	100	99.9042197	12.60	1e+099
44	Ru	101	100.9055822	17.06	1e+099
44	Ru	102	101.9043495	31.55	1e+099
44	Ru	103	102.9063237		3392064
44	Ru	104	103.905430	18.62	1e+099
44	Ru	105	104.907750		15984
44	Ru	106	105.907327		32278176
44	Ru	107	106.90991		225
44	Ru	108	107.91019		273
44	Ru	109	108.913200		34.5
44	Ru	110	109.91397		11.6
44	Ru	111	110.91756		2.12
44	Ru	112	111.91855		1.75
44	Ru	113	112.92254		0.8
44	Ru	114	113.92400		0.53
44	Ru	115	114.92831		0.74
44	Ru	116	115.93016		1.e-099
44	Ru	117	116.93479		1.e-099
44	Ru	118	117.93703		1.e-099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
45	Rh	89	88.94938		1.e-099
45	Rh	90	89.94287		1.5e-002
45	Rh	91	90.93655		1.74
45	Rh	92	91.93198		4.3
45	Rh	93	92.92574		13.9
45	Rh	94	93.92170		70.6
45	Rh	95	94.91590		301.2
45	Rh	96	95.914518		594
45	Rh	97	96.911340		1842
45	Rh	98	97.910716		523.2
45	Rh	99	98.908132		1391040
45	Rh	100	99.908117		74880
45	Rh	101	100.906164		104137856.064
45	Rh	102	101.906843		17884800
45	Rh	103	102.905504	100	1e+099
45	Rh	104	103.906655		42.3
45	Rh	105	104.905692		127296
45	Rh	106	105.907285		29.80
45	Rh	107	106.906751		1302
45	Rh	108	107.90873		16.8
45	Rh	109	108.908736		80
45	Rh	110	109.91095		28.5
45	Rh	111	110.91166		11
45	Rh	112	111.91461		3.4
45	Rh	113	112.91542		2.80
45	Rh	114	113.91885		1.85
45	Rh	115	114.92012		0.99
45	Rh	116	115.92371		0.68
45	Rh	117	116.92535		0.44
45	Rh	118	117.92943		0.31
45	Rh	119	118.93136		1.e-099
45	Rh	120	119.93578		1.e-099
45	Rh	121	120.93808		1.e-099

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
46	Pd	91	90.94948		1.e-099
46	Pd	92	91.94042		1.1
46	Pd	93	92.93591		1.07
46	Pd	94	93.92877		9.0
46	Pd	95	94.92469		1.e-099
46	Pd	96	95.91822		122
46	Pd	97	96.91648		186
46	Pd	98	97.912721		1062
46	Pd	99	98.911768		1284
46	Pd	100	99.908505		313632
46	Pd	101	100.908289		30492
46	Pd	102	101.905608	1.02	1e+099
46	Pd	103	102.906087		1468022.4
46	Pd	104	103.904035	11.14	1e+099
46	Pd	105	104.905084	22.33	1e+099
46	Pd	106	105.903483	27.33	1e+099
46	Pd	107	106.905128		205120019000000
46	Pd	108	107.903894	26.46	1e+099
46	Pd	109	108.905954		49324.32
46	Pd	110	109.905152	11.72	1e+099
46	Pd	111	110.907640		1404
46	Pd	112	111.907313		75708
46	Pd	113	112.910150		93
46	Pd	114	113.910365		145.2
46	Pd	115	114.913680		25
46	Pd	116	115.914160		11.8
46	Pd	117	116.91784		4.3
46	Pd	118	117.91898		1.9
46	Pd	119	118.92268		0.92
46	Pd	120	119.92403		0.5
46	Pd	121	120.92818		1.e-099
46	Pd	122	121.92980		1.e-099
46	Pd	123	122.93426		1.e-099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
47	Ag	94	93.94278		3.7e-002
47	Ag	95	94.93548		1.74
47	Ag	96	95.93068		4.45
47	Ag	97	96.92400		25.3
47	Ag	98	97.92176		47.5
47	Ag	99	98.91760		124
47	Ag	100	99.916070		120.6
47	Ag	101	100.91280		666
47	Ag	102	101.912000		774
47	Ag	103	102.908972		3942
47	Ag	104	103.908628		4152
47	Ag	105	104.906528		3567456
47	Ag	106	105.906666		1437.6
47	Ag	107	106.905093	51.839	1e+099
47	Ag	108	107.905954		142.2
47	Ag	109	108.904756	48.161	1e+099
47	Ag	110	109.906110		24.6
47	Ag	111	110.905295		643680
47	Ag	112	111.907004		11268
47	Ag	113	112.906566		19332
47	Ag	114	113.908808		4.6
47	Ag	115	114.908760		1200
47	Ag	116	115.911360		160.8
47	Ag	117	116.911680		73.6
47	Ag	118	117.914580		3.76
47	Ag	119	118.91567		6.0
47	Ag	120	119.918790		1.23
47	Ag	121	120.91985		0.79
47	Ag	122	121.92332		0.52
47	Ag	123	122.92490		0.296
47	Ag	124	123.92853		0.172
47	Ag	125	124.93054		0.166
47	Ag	126	125.93450		0.107

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
47	Ag	127	126.93688		7.9e-002
48	Cd	96	95.93977		1.e-099
48	Cd	97	96.93494		2.8
48	Cd	98	97.92758		9.2
48	Cd	99	98.92501		16
48	Cd	100	99.92023		49.1
48	Cd	101	100.91868		81.6
48	Cd	102	101.914780		330
48	Cd	103	102.913419		438
48	Cd	104	103.909848		3462
48	Cd	105	104.909468		3330
48	Cd	106	105.906458	1.25	1e+099
48	Cd	107	106.906614		23400
48	Cd	108	107.904183	0.89	1e+099
48	Cd	109	108.904986		39864960
48	Cd	110	109.903006	12.49	1e+099
48	Cd	111	110.904182	12.80	1e+099
48	Cd	112	111.9027572	24.13	1e+099
48	Cd	113	112.9044009	12.22	2.429883302e+023
48	Cd	114	113.9033581	28.73	1e+099
48	Cd	115	114.905431		192456
48	Cd	116	115.904755	7.49	9.4670778e+026
48	Cd	117	116.907218		8964
48	Cd	118	117.906914		3018
48	Cd	119	118.909920		161.4
48	Cd	120	119.909851		50.80
48	Cd	121	120.912980		13.5
48	Cd	122	121.91350		5.24
48	Cd	123	122.917000		2.10
48	Cd	124	123.917650		1.25
48	Cd	125	124.921250		0.65
48	Cd	126	125.922350		0.515
48	Cd	127	126.926430		0.37

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
48	Cd	128	127.92776		0.28
48	Cd	129	128.93226		0.242
48	Cd	130	129.93398		0.162
49	In	98	97.94224		4.5e-002
49	In	99	98.93461		3.1
49	In	100	99.93115		5.9
49	In	101	100.92656		15.1
49	In	102	101.92471		23.3
49	In	103	102.919914		60
49	In	104	103.91834		108
49	In	105	104.914673		304.2
49	In	106	105.913461		372
49	In	107	106.910292		1944
49	In	108	107.909720		3480
49	In	109	108.907154		15120
49	In	110	109.907169		17640
49	In	111	110.905111		242326.08
49	In	112	111.905533		898.2
49	In	113	112.904061	4.29	1e+099
49	In	114	113.904917		71.9
49	In	115	114.903878	95.71	1.3916604366e+022
49	In	116	115.905260		14.10
49	In	117	116.904516		2592
49	In	118	117.906355		5.0
49	In	119	118.905846		144
49	In	120	119.907960		3.08
49	In	121	120.907849		23.1
49	In	122	121.910280		1.5
49	In	123	122.910439		5.98
49	In	124	123.913180		3.11
49	In	125	124.913600		2.36
49	In	126	125.916460		1.53
49	In	127	126.917340		1.09

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
49	In	128	127.920170		0.84
49	In	129	128.92166		0.611
49	In	130	129.924850		0.29
49	In	131	130.926770		0.28
49	In	132	131.932920		0.206
49	In	133	132.93834		0.165
49	In	134	133.94466		0.14
50	Sn	100	99.93895		1.1
50	Sn	101	100.93606		3
50	Sn	102	101.93049		4.6
50	Sn	103	102.92813		7
50	Sn	104	103.92319		20.8
50	Sn	105	104.92139		34
50	Sn	106	105.916880		115.2
50	Sn	107	106.915670		174
50	Sn	108	107.911970		618
50	Sn	109	108.911287		1080
50	Sn	110	109.907853		14796
50	Sn	111	110.907735		2118
50	Sn	112	111.904821	0.97	1e+099
50	Sn	113	112.905173		9943776
50	Sn	114	113.902782	0.66	1e+099
50	Sn	115	114.903346	0.34	1e+099
50	Sn	116	115.901744	14.54	1e+099
50	Sn	117	116.902954	7.68	1e+099
50	Sn	118	117.901606	24.22	1e+099
50	Sn	119	118.903309	8.59	1e+099
50	Sn	120	119.9021966	32.58	1e+099
50	Sn	121	120.9042369		97308
50	Sn	122	121.9034401	4.63	1e+099
50	Sn	123	122.9057219		11162880
50	Sn	124	123.9052746	5.79	1e+099
50	Sn	125	124.9077849		832896

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
50	Sn	126	125.907654		7258092980000
50	Sn	127	126.910351		7560
50	Sn	128	127.910535		3544.2
50	Sn	129	128.91344		133.8
50	Sn	130	129.913850		223.2
50	Sn	131	130.916920		56.0
50	Sn	132	131.917744		39.7
50	Sn	133	132.923810		1.45
50	Sn	134	133.92846		1.12
50	Sn	135	134.93473		0.53
50	Sn	136	135.93934		0.25
50	Sn	137	136.94579		0.19
51	Sb	103	102.94012		1.e-099
51	Sb	104	103.93629		0.47
51	Sb	105	104.93153		1.12
51	Sb	106	105.92876		0.6
51	Sb	107	106.92415		4.6
51	Sb	108	107.92216		7.4
51	Sb	109	108.918136		17.0
51	Sb	110	109.91676		23.0
51	Sb	111	110.91321		75
51	Sb	112	111.912395		51.4
51	Sb	113	112.909378		400.2
51	Sb	114	113.90910		209.4
51	Sb	115	114.906599		1926
51	Sb	116	115.906797		948
51	Sb	117	116.904840		10080
51	Sb	118	117.905532		216
51	Sb	119	118.903946		137484
51	Sb	120	119.905074		953.4
51	Sb	121	120.9038180	57.21	1e+099
51	Sb	122	121.9051754		235336.32
51	Sb	123	122.9042157	42.79	1e+099

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
51	Sb	124	123.9059375		5201280
51	Sb	125	124.905248		87051674.0072448
51	Sb	126	125.907250		1067040
51	Sb	127	126.906915		332640
51	Sb	128	127.909167		32436
51	Sb	129	128.909150		15840
51	Sb	130	129.911546		2370
51	Sb	131	130.911950		1381.8
51	Sb	132	131.914413		167.4
51	Sb	133	132.915240		150
51	Sb	134	133.920550		0.78
51	Sb	135	134.92517		1.68
51	Sb	136	135.93066		0.923
51	Sb	137	136.93531		0.45
51	Sb	138	137.94096		1.e-099
51	Sb	139	138.94571		1.e-099
52	Te	106	105.93770		7.e-005
52	Te	107	106.93504		3.1e-003
52	Te	108	107.92949		2.1
52	Te	109	108.927460		4.6
52	Te	110	109.922410		18.6
52	Te	111	110.921120		19.3
52	Te	112	111.91706		120
52	Te	113	112.91593		102
52	Te	114	113.91206		912
52	Te	115	114.91158		348
52	Te	116	115.90842		8964
52	Te	117	116.908634		3720
52	Te	118	117.905825		518400
52	Te	119	118.906408		57780
52	Te	120	119.904020	0.09	1e+099
52	Te	121	120.904930		1655424
52	Te	122	121.9030471	2.55	1e+099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
52	Te	123	122.9042730	0.89	1.89341556e+022
52	Te	124	123.9028195	4.74	1e+099
52	Te	125	124.9044247	7.07	1e+099
52	Te	126	125.9033055	18.84	1e+099
52	Te	127	126.905217		33660
52	Te	128	127.9044614	31.74	6.94252372e+031
52	Te	129	128.906596		4176
52	Te	130	129.9062228	34.08	2.492997154e+028
52	Te	131	130.9085219		1500
52	Te	132	131.908524		276825.6
52	Te	133	132.910940		750
52	Te	134	133.911540		2508
52	Te	135	134.91645		19.0
52	Te	136	135.920100		17.63
52	Te	137	136.92532		2.49
52	Te	138	137.92922		1.4
52	Te	139	138.93473		1.e-099
52	Te	140	139.93870		1.e-099
52	Te	141	140.94439		1.e-099
52	Te	142	141.94850		1.e-099
53	I	108	107.94329		3.6e-002
53	I	109	108.93819		1.03e-004
53	I	110	109.93521		0.65
53	I	111	110.93028		2.5
53	I	112	111.92797		3.42
53	I	113	112.923640		6.6
53	I	114	113.92185		2.1
53	I	115	114.91792		78
53	I	116	115.91674		2.91
53	I	117	116.913650		133.2
53	I	118	117.913380		822
53	I	119	118.910180		1146
53	I	120	119.910048		4896

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
53	I	121	120.907366		7632
53	I	122	121.907592		217.8
53	I	123	122.905598		47604.6
53	I	124	123.9062114		360806.4
53	I	125	124.9046241		5132160
53	I	126	125.905619		1117152
53	I	127	126.904468	100	1e+099
53	I	128	127.905805		1499.4
53	I	129	128.904987		495443738200000
53	I	130	129.906674		44496
53	I	131	130.9061242		692988.48
53	I	132	131.907995		8262
53	I	133	132.907806		74880
53	I	134	133.909877		3150
53	I	135	134.910050		23652
53	I	136	135.914660		83.4
53	I	137	136.917873		24.13
53	I	138	137.922380		6.23
53	I	139	138.926090		2.282
53	I	140	139.93121		0.86
53	I	141	140.93483		0.43
53	I	142	141.94018		0
53	I	143	142.94407		1.e-099
53	I	144	143.94961		1.e-099
54	Xe	110	109.94448		0.31
54	Xe	111	110.94163		0.74
54	Xe	112	111.93567		2.7
54	Xe	113	112.93338		2.74
54	Xe	114	113.92815		10.0
54	Xe	115	114.92654		18
54	Xe	116	115.92174		59
54	Xe	117	116.92056		61
54	Xe	118	117.91657		228

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
54	Xe	119	118.91555		348
54	Xe	120	119.912150		2400
54	Xe	121	120.911386		2406
54	Xe	122	121.908550		72360
54	Xe	123	122.908471		7488
54	Xe	124	123.9058958	0.09	1e+099
54	Xe	125	124.9063982		60840
54	Xe	126	125.904269	0.09	1e+099
54	Xe	127	126.905180		3140208
54	Xe	128	127.9035304	1.92	1e+099
54	Xe	129	128.9047795	26.44	1e+099
54	Xe	130	129.9035079	4.08	1e+099
54	Xe	131	130.9050819	21.18	1e+099
54	Xe	132	131.9041545	26.89	1e+099
54	Xe	133	132.905906		453384
54	Xe	134	133.9053945	10.44	1e+099
54	Xe	135	134.907207		32904
54	Xe	136	135.907220	8.87	1e+099
54	Xe	137	136.911563		229.08
54	Xe	138	137.913990		844.8
54	Xe	139	138.918787		39.68
54	Xe	140	139.921640		13.60
54	Xe	141	140.92665		1.73
54	Xe	142	141.92970		1.22
54	Xe	143	142.93489		0.511
54	Xe	144	143.93823		0.388
54	Xe	145	144.94367		0.188
54	Xe	146	145.94730		0.146
54	Xe	147	146.95301		0.13
55	Cs	112	111.95033		5.e-004
55	Cs	113	112.94454		1.67e-005
55	Cs	114	113.94142		0.57
55	Cs	115	114.93594		1.4

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
55	Cs	116	115.93291		0.7
55	Cs	117	116.928640		8.4
55	Cs	118	117.926555		14
55	Cs	119	118.922371		43.0
55	Cs	120	119.920678		61.2
55	Cs	121	120.917184		155
55	Cs	122	121.916122		21.18
55	Cs	123	122.912990		352.2
55	Cs	124	123.912246		30.9
55	Cs	125	124.909725		2700
55	Cs	126	125.909448		98.4
55	Cs	127	126.907418		22500
55	Cs	128	127.907748		218.4
55	Cs	129	128.906063		115416
55	Cs	130	129.906706		1752.6
55	Cs	131	130.905460		837129.6
55	Cs	132	131.906430		559785.6
55	Cs	133	132.905447	100	1e+099
55	Cs	134	133.906713		65158740.969984
55	Cs	135	134.905972		72580929800000
55	Cs	136	135.907306		1137024
55	Cs	137	136.907084		951980944.747968
55	Cs	138	137.911011		2004.6
55	Cs	139	138.913358		556.2
55	Cs	140	139.917277		63.7
55	Cs	141	140.920044		24.84
55	Cs	142	141.924292		1.689
55	Cs	143	142.927330		1.791
55	Cs	144	143.932030		0.994
55	Cs	145	144.935390		0.582
55	Cs	146	145.940160		0.323
55	Cs	147	146.94386		0.225
55	Cs	148	147.94890		0.146

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
55	Cs	149	148.95272		1.e-099
55	Cs	150	149.95797		1.e-099
55	Cs	151	150.96200		1.e-099
56	Ba	114	113.95094		0.53
56	Ba	115	114.94771		0.45
56	Ba	116	115.94168		1.3
56	Ba	117	116.93886		1.75
56	Ba	118	117.93344		5.2
56	Ba	119	118.93105		5.4
56	Ba	120	119.92605		24
56	Ba	121	120.92449		29.7
56	Ba	122	121.92026		117
56	Ba	123	122.91885		162
56	Ba	124	123.915088		660
56	Ba	125	124.91462		210
56	Ba	126	125.911244		6000
56	Ba	127	126.91112		762
56	Ba	128	127.908309		209952
56	Ba	129	128.908674		8028
56	Ba	130	129.906310	0.106	1e+099
56	Ba	131	130.906931		993600
56	Ba	132	131.905056	0.101	1e+099
56	Ba	133	132.906002		331663293.1008
56	Ba	134	133.904503	2.417	1e+099
56	Ba	135	134.905683	6.592	1e+099
56	Ba	136	135.904570	7.854	1e+099
56	Ba	137	136.905821	11.232	1e+099
56	Ba	138	137.905241	71.698	1e+099
56	Ba	139	138.908835		4986
56	Ba	140	139.910599		1101772.8
56	Ba	141	140.914406		1096.2
56	Ba	142	141.916448		636
56	Ba	143	142.920617		14.5

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
56	Ba	144	143.922940		11.5
56	Ba	145	144.926920		4.31
56	Ba	146	145.930110		2.22
56	Ba	147	146.93399		0.893
56	Ba	148	147.93768		0.612
56	Ba	149	148.94246		0.344
56	Ba	150	149.94562		0.3
56	Ba	151	150.95070		1.e-099
56	Ba	152	151.95416		1.e-099
56	Ba	153	152.95961		1.e-099
57	La	117	116.95001		2.35e-002
57	La	118	117.94657		1.e-099
57	La	119	118.94099		1.e-099
57	La	120	119.93807		2.8
57	La	121	120.93301		5.3
57	La	122	121.93071		8.7
57	La	123	122.92624		17
57	La	124	123.92453		29.21
57	La	125	124.92067		64.8
57	La	126	125.91937		54
57	La	127	126.91616		306
57	La	128	127.91545		310.8
57	La	129	128.912670		696
57	La	130	129.91232		522
57	La	131	130.91011		3540
57	La	132	131.910110		17280
57	La	133	132.90840		14083.2
57	La	134	133.908490		387
57	La	135	134.906971		70200
57	La	136	135.907650		592.2
57	La	137	136.906470		1893415560000
57	La	138	137.907107	0.090	3.218806452e+018
57	La	139	138.906348	99.910	1e+099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
57	La	140	139.909473		144987.84
57	La	141	140.910957		14112
57	La	142	141.914074		5466
57	La	143	142.916059		852
57	La	144	143.919590		40.8
57	La	145	144.921640		24.8
57	La	146	145.925700		6.27
57	La	147	146.927820		4.015
57	La	148	147.93219		1.26
57	La	149	148.93437		1.05
57	La	150	149.93857		0.51
57	La	151	150.94156		1.e-099
57	La	152	151.94611		1.e-099
57	La	153	152.94945		1.e-099
57	La	154	153.95440		1.e-099
57	La	155	154.95813		1.e-099
58	Ce	119	118.95276		1.e-099
58	Ce	120	119.94664		1.e-099
58	Ce	121	120.94367		1.1
58	Ce	122	121.93801		1.e-099
58	Ce	123	122.93551		3.8
58	Ce	124	123.93052		9.1
58	Ce	125	124.92854		9.3
58	Ce	126	125.92410		51.0
58	Ce	127	126.92275		29
58	Ce	128	127.91887		235.8
58	Ce	129	128.91809		210
58	Ce	130	129.91469		1374
58	Ce	131	130.91442		612
58	Ce	132	131.91149		12636
58	Ce	133	132.91155		5820
58	Ce	134	133.90903		273024
58	Ce	135	134.909146		63720

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
58	Ce	136	135.907140	0.185	1e+099
58	Ce	137	136.907780		32400
58	Ce	138	137.905986	0.251	1e+099
58	Ce	139	138.906647		11892182.4
58	Ce	140	139.905434	88.450	1e+099
58	Ce	141	140.908271		2808691.2
58	Ce	142	141.909240	11.114	1e+099
58	Ce	143	142.912381		118940.4
58	Ce	144	143.913643		24616224
58	Ce	145	144.917230		180.6
58	Ce	146	145.918690		811.2
58	Ce	147	146.922510		56.4
58	Ce	148	147.92439		56
58	Ce	149	148.928290		5.3
58	Ce	150	149.93023		4.0
58	Ce	151	150.93404		1.02
58	Ce	152	151.93638		1.1
58	Ce	153	152.94058		1.e-099
58	Ce	154	153.94332		1.e-099
58	Ce	155	154.94804		1.e-099
58	Ce	156	155.95126		1.e-099
58	Ce	157	156.95634		1.e-099
59	Pr	121	120.95536		0.6
59	Pr	122	121.95165		1.e-099
59	Pr	123	122.94596		1.e-099
59	Pr	124	123.94296		1.2
59	Pr	125	124.93783		3.3
59	Pr	126	125.93531		3.12
59	Pr	127	126.93083		4.2
59	Pr	128	127.92880		2.84
59	Pr	129	128.92486		30
59	Pr	130	129.92338		40.0
59	Pr	131	130.92006		90

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
59	Pr	132	131.91912		89.4
59	Pr	133	132.91620		390
59	Pr	134	133.91567		0
59	Pr	135	134.91314		1440
59	Pr	136	135.912650		786
59	Pr	137	136.910680		4608
59	Pr	138	137.910749		87
59	Pr	139	138.908932		15876
59	Pr	140	139.909071		203.4
59	Pr	141	140.907648	100	1e+099
59	Pr	142	141.910040		68832
59	Pr	143	142.910812		1172448
59	Pr	144	143.913301		1036.8
59	Pr	145	144.914507		21542.4
59	Pr	146	145.917590		1449
59	Pr	147	146.918980		804
59	Pr	148	147.92218		137.4
59	Pr	149	148.923791		135.6
59	Pr	150	149.927000		6.19
59	Pr	151	150.928230		18.90
59	Pr	152	151.93160		3.63
59	Pr	153	152.93365		4.28
59	Pr	154	153.93739		2.3
59	Pr	155	154.93999		1.e-099
59	Pr	156	155.94412		1.e-099
59	Pr	157	156.94717		1.e-099
59	Pr	158	157.95178		1.e-099
59	Pr	159	158.95523		1.e-099
60	Nd	126	125.94307		1.e-099
60	Nd	127	126.94050		1.8
60	Nd	128	127.93539		1.e-099
60	Nd	129	128.93325		4.9
60	Nd	130	129.92878		21

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
60	Nd	131	130.92710		33
60	Nd	132	131.92312		93.6
60	Nd	133	132.92221		70
60	Nd	134	133.91865		510
60	Nd	135	134.91824		744
60	Nd	136	135.915020		3042
60	Nd	137	136.914640		2310
60	Nd	138	137.91193		18144
60	Nd	139	138.911920		1782
60	Nd	140	139.909310		291168
60	Nd	141	140.909605		8964
60	Nd	142	141.907719	27.2	1e+099
60	Nd	143	142.909810	12.2	1e+099
60	Nd	144	143.910083	23.8	7.226536054e+022
60	Nd	145	144.912569	8.3	1e+099
60	Nd	146	145.913112	17.2	1e+099
60	Nd	147	146.916096		948672
60	Nd	148	147.916889	5.7	1e+099
60	Nd	149	148.920144		6220.8
60	Nd	150	149.920887	5.6	2.114314042e+026
60	Nd	151	150.923825		746.4
60	Nd	152	151.924680		684
60	Nd	153	152.927695		31.6
60	Nd	154	153.92948		25.9
60	Nd	155	154.93263		8.9
60	Nd	156	155.93520		5.49
60	Nd	157	156.93927		1.e-099
60	Nd	158	157.94187		1.e-099
60	Nd	159	158.94639		1.e-099
60	Nd	160	159.94939		1.e-099
60	Nd	161	160.95433		1.e-099
61	Pm	128	127.94826		1.0
61	Pm	129	128.94316		1.e-099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
61	Pm	130	129.94045		2.6
61	Pm	131	130.93580		6.3
61	Pm	132	131.93375		6.3
61	Pm	133	132.92972		15
61	Pm	134	133.92849		22
61	Pm	135	134.92462		49
61	Pm	136	135.92345		107
61	Pm	137	136.92071		1.e-099
61	Pm	138	137.91945		10
61	Pm	139	138.916760		249
61	Pm	140	139.915800		9.2
61	Pm	141	140.913607		1254
61	Pm	142	141.912950		40.5
61	Pm	143	142.910928		22896000
61	Pm	144	143.912586		31363200
61	Pm	145	144.912744		558557591.616
61	Pm	146	145.914692		174509801.2224
61	Pm	147	146.915134		82786439.878272
61	Pm	148	147.917468		463795.2
61	Pm	149	148.918329		191088
61	Pm	150	149.920979		9648
61	Pm	151	150.921203		102240
61	Pm	152	151.923490		247.2
61	Pm	153	152.924113		315
61	Pm	154	153.926550		103.8
61	Pm	155	154.928100		41.5
61	Pm	156	155.931060		26.70
61	Pm	157	156.93320		10.56
61	Pm	158	157.93669		4.8
61	Pm	159	158.93913		1.47
61	Pm	160	159.94299		1.e-099
61	Pm	161	160.94586		1.e-099
61	Pm	162	161.95029		1.e-099

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
61	Pm	163	162.95352		1.e-099
62	Sm	130	129.94863		1.e-099
62	Sm	131	130.94589		1.2
62	Sm	132	131.94082		4.0
62	Sm	133	132.93873		2.90
62	Sm	134	133.93402		10
62	Sm	135	134.93235		10.3
62	Sm	136	135.92830		47
62	Sm	137	136.92705		45
62	Sm	138	137.92354		186
62	Sm	139	138.922302		154.2
62	Sm	140	139.918991		889.2
62	Sm	141	140.918469		612
62	Sm	142	141.915193		4349.4
62	Sm	143	142.914624		525
62	Sm	144	143.911995	3.07	1e+099
62	Sm	145	144.913406		29376000
62	Sm	146	145.913037		3.250363378e+015
62	Sm	147	146.914893	14.99	3.345034156e+018
62	Sm	148	147.914818	11.24	2.20898482e+023
62	Sm	149	148.917180	13.82	1e+099
62	Sm	150	149.917271	7.38	1e+099
62	Sm	151	150.919928		2840123347.2
62	Sm	152	151.919728	26.75	1e+099
62	Sm	153	152.922094		166622.4
62	Sm	154	153.922205	22.75	1e+099
62	Sm	155	154.924636		1338
62	Sm	156	155.925526		33840
62	Sm	157	156.928350		481.8
62	Sm	158	157.929990		318
62	Sm	159	158.93320		11.37
62	Sm	160	159.93514		9.6
62	Sm	161	160.93883		4.8

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
62	Sm	162	161.94122		2.4
62	Sm	163	162.94536		1.e-099
62	Sm	164	163.94828		1.e-099
62	Sm	165	164.95298		1.e-099
63	Eu	132	131.95416		1.e-099
63	Eu	133	132.94890		1.e-099
63	Eu	134	133.94632		0.5
63	Eu	135	134.94172		1.5
63	Eu	136	135.93950		3.3
63	Eu	137	136.93521		8.4
63	Eu	138	137.93345		12.1
63	Eu	139	138.92984		17.9
63	Eu	140	139.928080		1.51
63	Eu	141	140.924890		40.7
63	Eu	142	141.923400		2.36
63	Eu	143	142.920287		155.4
63	Eu	144	143.918774		10.2
63	Eu	145	144.916261		512352
63	Eu	146	145.917200		398304
63	Eu	147	146.916741		2082240
63	Eu	148	147.918154		4708800
63	Eu	149	148.917926		8043840
63	Eu	150	149.919698		1164450572.352
63	Eu	151	150.919846	47.81	1e+099
63	Eu	152	151.921740		427186108.34496
63	Eu	153	152.921226	52.19	1e+099
63	Eu	154	153.922975		271168665.80544
63	Eu	155	154.922889		150245680.759488
63	Eu	156	155.924751		1312416
63	Eu	157	156.925419		54648
63	Eu	158	157.927840		2754
63	Eu	159	158.929084		1086
63	Eu	160	159.93197		38

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
63	Eu	161	160.93368		26
63	Eu	162	161.93704		10.6
63	Eu	163	162.93921		1.e-099
63	Eu	164	163.94299		1.e-099
63	Eu	165	164.94572		1.e-099
63	Eu	166	165.94997		1.e-099
63	Eu	167	166.95305		1.e-099
64	Gd	136	135.94707		1.e-099
64	Gd	137	136.94465		2.2
64	Gd	138	137.93997		4.7
64	Gd	139	138.93808		5.7
64	Gd	140	139.93395		15.8
64	Gd	141	140.93221		14
64	Gd	142	141.92823		70.2
64	Gd	143	142.92674		39
64	Gd	144	143.92279		268.2
64	Gd	145	144.921690		1380
64	Gd	146	145.918305		4170528
64	Gd	147	146.919089		137016
64	Gd	148	147.918110		2354146685.568
64	Gd	149	148.919336		801792
64	Gd	150	149.918655		56486897540000
64	Gd	151	150.920344		10713600
64	Gd	152	151.919788	0.20	3.408148008e+021
64	Gd	153	152.921746		20770560
64	Gd	154	153.920862	2.18	1e+099
64	Gd	155	154.922619	14.80	1e+099
64	Gd	156	155.922120	20.47	1e+099
64	Gd	157	156.923957	15.65	1e+099
64	Gd	158	157.924101	24.84	1e+099
64	Gd	159	158.926385		66524.4
64	Gd	160	159.927051	21.86	1e+099
64	Gd	161	160.929666		218.76

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
64	Gd	162	161.930981		504
64	Gd	163	162.93399		68
64	Gd	164	163.93586		45
64	Gd	165	164.93938		10.3
64	Gd	166	165.94160		4.8
64	Gd	167	166.94557		1.e-099
64	Gd	168	167.94836		1.e-099
64	Gd	169	168.95287		1.e-099
65	Tb	138	137.95287		1.e-099
65	Tb	139	138.94803		1.6
65	Tb	140	139.94554		2.4
65	Tb	141	140.94116		3.5
65	Tb	142	141.93886		0.597
65	Tb	143	142.93475		12
65	Tb	144	143.93253		~1
65	Tb	145	144.92888		1.e-099
65	Tb	146	145.927180		8
65	Tb	147	146.924037		5904
65	Tb	148	147.924300		3600
65	Tb	149	148.923242		14824.8
65	Tb	150	149.923654		12528
65	Tb	151	150.923098		63392.4
65	Tb	152	151.924070		63000
65	Tb	153	152.923431		202176
65	Tb	154	153.924690		77400
65	Tb	155	154.923500		459648
65	Tb	156	155.924744		462240
65	Tb	157	156.924021		2240541751.68
65	Tb	158	157.925410		5680246694.4
65	Tb	159	158.925343	100	1e+099
65	Tb	160	159.927164		6246720
65	Tb	161	160.927566		596678.4
65	Tb	162	161.929480		456

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
65	Tb	163	162.930644		1170
65	Tb	164	163.93335		180
65	Tb	165	164.93488		126.6
65	Tb	166	165.93805		25.6
65	Tb	167	166.94005		19
65	Tb	168	167.94364		8.2
65	Tb	169	168.94622		1.e-099
65	Tb	170	169.95025		1.e-099
65	Tb	171	170.95330		1.e-099
66	Dy	140	139.95379		1.e-099
66	Dy	141	140.95119		0.9
66	Dy	142	141.94627		2.3
66	Dy	143	142.94383		5.6
66	Dy	144	143.93907		9.1
66	Dy	145	144.93695		9.5
66	Dy	146	145.93272		33.2
66	Dy	147	146.930880		40
66	Dy	148	147.927180		198
66	Dy	149	148.927334		252
66	Dy	150	149.925580		430.2
66	Dy	151	150.926180		1074
66	Dy	152	151.924714		8568
66	Dy	153	152.925761		23040
66	Dy	154	153.924422		94670778000000
66	Dy	155	154.925749		35640
66	Dy	156	155.924278	0.06	1e+099
66	Dy	157	156.925461		29304
66	Dy	158	157.924405	0.10	1e+099
66	Dy	159	158.925736		12476160
66	Dy	160	159.925194	2.34	1e+099
66	Dy	161	160.926930	18.91	1e+099
66	Dy	162	161.926795	25.51	1e+099
66	Dy	163	162.928728	24.90	1e+099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
66	Dy	164	163.929171	28.18	1e+099
66	Dy	165	164.931700		8402.4
66	Dy	166	165.932803		293760
66	Dy	167	166.935650		372
66	Dy	168	167.93723		522
66	Dy	169	168.94030		39
66	Dy	170	169.94267		1.e-099
66	Dy	171	170.94648		1.e-099
66	Dy	172	171.94911		1.e-099
66	Dy	173	172.95344		1.e-099
67	Ho	142	141.95986		0.4
67	Ho	143	142.95469		1.e-099
67	Ho	144	143.95164		0.7
67	Ho	145	144.94688		2.4
67	Ho	146	145.94410		3.6
67	Ho	147	146.93984		5.8
67	Ho	148	147.93727		2.2
67	Ho	149	148.933790		21.1
67	Ho	150	149.93335		76.8
67	Ho	151	150.931681		35.2
67	Ho	152	151.931740		161.8
67	Ho	153	152.930195		120.6
67	Ho	154	153.930596		705.6
67	Ho	155	154.929079		2880
67	Ho	156	155.92971		3360
67	Ho	157	156.928190		756
67	Ho	158	157.928950		678
67	Ho	159	158.927709		1983
67	Ho	160	159.928726		1536
67	Ho	161	160.927852		8928
67	Ho	162	161.929092		900
67	Ho	163	162.928730		144215151820
67	Ho	164	163.930231		1740

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
67	Ho	165	164.930319	100	1e+099
67	Ho	166	165.932281		96588
67	Ho	167	166.933126		11160
67	Ho	168	167.935500		179.4
67	Ho	169	168.936868		282
67	Ho	170	169.939610		165.6
67	Ho	171	170.94146		53
67	Ho	172	171.94482		25
67	Ho	173	172.94729		1.e-099
67	Ho	174	173.95115		1.e-099
67	Ho	175	174.95405		1.e-099
68	Er	144	143.96059		1.e-099
68	Er	145	144.95746		0.9
68	Er	146	145.95212		1.7
68	Er	147	146.94931		~2.5
68	Er	148	147.94444		4.6
68	Er	149	148.94217		4
68	Er	150	149.93776		18.5
68	Er	151	150.93746		23.5
68	Er	152	151.935080		10.3
68	Er	153	152.935093		37.1
68	Er	154	153.932777		223.8
68	Er	155	154.933200		318
68	Er	156	155.931020		1170
68	Er	157	156.931950		1119
68	Er	158	157.92991		8244
68	Er	159	158.930681		2160
68	Er	160	159.929080		102888
68	Er	161	160.930001		11556
68	Er	162	161.928775	0.14	1e+099
68	Er	163	162.930029		4500
68	Er	164	163.929197	1.61	1e+099
68	Er	165	164.930723		37296

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
68	Er	166	165.930290	33.61	1e+099
68	Er	167	166.932045	22.93	1e+099
68	Er	168	167.932368	26.78	1e+099
68	Er	169	168.934588		812160
68	Er	170	169.935460	14.93	1e+099
68	Er	171	170.938026		27057.6
68	Er	172	171.939352		177480
68	Er	173	172.94240		86.04
68	Er	174	173.94434		192
68	Er	175	174.94793		72
68	Er	176	175.95029		1.e-099
68	Er	177	176.95437		1.e-099
69	Tm	146	145.96650		0.24
69	Tm	147	146.96108		0.58
69	Tm	148	147.95755		0.7
69	Tm	149	148.95265		0.9
69	Tm	150	149.94967		1.e-099
69	Tm	151	150.94543		4.17
69	Tm	152	151.94430		8.0
69	Tm	153	152.942028		1.48
69	Tm	154	153.94142		8.1
69	Tm	155	154.939192		21.6
69	Tm	156	155.939010		83.8
69	Tm	157	156.93676		217.8
69	Tm	158	157.93700		238.8
69	Tm	159	158.934810		547.8
69	Tm	160	159.93509		564
69	Tm	161	160.93340		1812
69	Tm	162	161.933970		1302
69	Tm	163	162.932648		6516
69	Tm	164	163.933451		120
69	Tm	165	164.932432		108216
69	Tm	166	165.933553		27720

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
69	Tm	167	166.932849		799200
69	Tm	168	167.934170		8043840
69	Tm	169	168.934211	100	1e+099
69	Tm	170	169.935798		11111040
69	Tm	171	170.936426		60589298.0736
69	Tm	172	171.938396		228960
69	Tm	173	172.939600		29664
69	Tm	174	173.942160		324
69	Tm	175	174.943830		912
69	Tm	176	175.94699		111
69	Tm	177	176.94904		90
69	Tm	178	177.95264		1.e-099
69	Tm	179	178.95534		1.e-099
70	Yb	148	147.96676		1.e-099
70	Yb	149	148.96348		0.7
70	Yb	150	149.95799		1.e-099
70	Yb	151	150.95525		1.6
70	Yb	152	151.95017		3.04
70	Yb	153	152.94921		4.2
70	Yb	154	153.94624		0.409
70	Yb	155	154.94579		1.793
70	Yb	156	155.942850		26.1
70	Yb	157	156.942660		38.6
70	Yb	158	157.939858		89.4
70	Yb	159	158.94015		103.2
70	Yb	160	159.93756		288
70	Yb	161	160.93785		252
70	Yb	162	161.93575		1132.2
70	Yb	163	162.93627		663
70	Yb	164	163.93452		4548
70	Yb	165	164.935398		594
70	Yb	166	165.933880		204120
70	Yb	167	166.934947		1050

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
70	Yb	168	167.933894	0.13	1e+099
70	Yb	169	168.935187		2767046.4
70	Yb	170	169.934759	3.04	1e+099
70	Yb	171	170.936322	14.28	1e+099
70	Yb	172	171.9363777	21.83	1e+099
70	Yb	173	172.9382068	16.13	1e+099
70	Yb	174	173.9388581	31.83	1e+099
70	Yb	175	174.9412725		361584
70	Yb	176	175.942568	12.76	1e+099
70	Yb	177	176.945257		6879.6
70	Yb	178	177.946643		4440
70	Yb	179	178.95017		480
70	Yb	180	179.95233		144
70	Yb	181	180.95615		1.e-099
71	Lu	150	149.97267		4.6e-002
71	Lu	151	150.96715		8.06e-002
71	Lu	152	151.96361		0.65
71	Lu	153	152.95869		0.9
71	Lu	154	153.95710		1.e-099
71	Lu	155	154.95423		6.86e-002
71	Lu	156	155.95291		0.494
71	Lu	157	156.950102		6.8
71	Lu	158	157.94917		10.6
71	Lu	159	158.946620		12.1
71	Lu	160	159.94602		36.1
71	Lu	161	160.94354		77
71	Lu	162	161.94322		82.2
71	Lu	163	162.94120		238.2
71	Lu	164	163.94122		188.4
71	Lu	165	164.939610		644.4
71	Lu	166	165.93976		159
71	Lu	167	166.93831		3090
71	Lu	168	167.938700		330

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
71	Lu	169	168.937649		122616
71	Lu	170	169.938472		173836.8
71	Lu	171	170.937910		711936
71	Lu	172	171.939082		578880
71	Lu	173	172.938927		43232988.7296
71	Lu	174	173.9403335		104453425.3248
71	Lu	175	174.9407679	97.41	1e+099
71	Lu	176	175.9426824	2.59	1.214941651e+018
71	Lu	177	176.9437550		574300.8
71	Lu	178	177.945951		1704
71	Lu	179	178.947324		16524
71	Lu	180	179.949880		342
71	Lu	181	180.95197		210
71	Lu	182	181.95521		120
71	Lu	183	182.95757		58
71	Lu	184	183.96117		20
72	Hf	154	153.96425		2
72	Hf	155	154.96276		0.89
72	Hf	156	155.95925		2.3e-002
72	Hf	157	156.95813		0.115
72	Hf	158	157.95465		2.84
72	Hf	159	158.95400		5.20
72	Hf	160	159.950710		13.6
72	Hf	161	160.950330		18.2
72	Hf	162	161.947203		39.4
72	Hf	163	162.94706		40.0
72	Hf	164	163.94442		111
72	Hf	165	164.94454		76
72	Hf	166	165.94225		406.2
72	Hf	167	166.94260		123
72	Hf	168	167.94063		1557
72	Hf	169	168.941160		194.4
72	Hf	170	169.93965		57636

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
72	Hf	171	170.94049		43560
72	Hf	172	171.939460		59011451.7696
72	Hf	173	172.94065		84960
72	Hf	174	173.940040	0.16	6.3113852e+022
72	Hf	175	174.941503		6048000
72	Hf	176	175.9414018	5.26	1e+099
72	Hf	177	176.9432200	18.60	1e+099
72	Hf	178	177.9436977	27.28	1e+099
72	Hf	179	178.9458151	13.62	1e+099
72	Hf	180	179.9465488	35.08	1e+099
72	Hf	181	180.9490991		3662496
72	Hf	182	181.950553		284012334000000
72	Hf	183	182.953530		3841.2
72	Hf	184	183.955450		14832
72	Hf	185	184.95878		210
72	Hf	186	185.96092		156
73	Ta	156	155.97169		0.144
73	Ta	157	156.96815		1.01e-002
73	Ta	158	157.96637		4.9e-002
73	Ta	159	158.96291		1.04
73	Ta	160	159.96136		1.70
73	Ta	161	160.958370		1.e-099
73	Ta	162	161.95715		3.57
73	Ta	163	162.954320		10.6
73	Ta	164	163.95357		14.2
73	Ta	165	164.95082		31.0
73	Ta	166	165.95047		34.4
73	Ta	167	166.94797		79.8
73	Ta	168	167.94779		120
73	Ta	169	168.94592		294
73	Ta	170	169.94609		405.6
73	Ta	171	170.94446		1398
73	Ta	172	171.94474		2208

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
73	Ta	173	172.94354		11304
73	Ta	174	173.944170		4104
73	Ta	175	174.94365		37800
73	Ta	176	175.94474		29124
73	Ta	177	176.944472		203616
73	Ta	178	177.94575		558.6
73	Ta	179	178.945934		57433605.4656
73	Ta	180	179.947466	0.012	29347.2
73	Ta	181	180.947996	99.988	1e+099
73	Ta	182	181.950152		9886752
73	Ta	183	182.951373		440640
73	Ta	184	183.954009		31320
73	Ta	185	184.955559		2964
73	Ta	186	185.958550		630
73	Ta	187	186.96041		1.e-099
73	Ta	188	187.96371		1.e-099
74	W	158	157.97394		1.37e-003
74	W	159	158.97228		8.2e-003
74	W	160	159.96837		9.e-002
74	W	161	160.96709		0.409
74	W	162	161.96334		1.36
74	W	163	162.96253		2.8
74	W	164	163.958980		6.3
74	W	165	164.958340		5.1
74	W	166	165.955020		19.2
74	W	167	166.95467		19.9
74	W	168	167.95186		51
74	W	169	168.95176		76
74	W	170	169.94929		145.2
74	W	171	170.94946		142.8
74	W	172	171.94742		396
74	W	173	172.94783		456
74	W	174	173.94616		1992

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
74	W	175	174.94677		2112
74	W	176	175.94559		9000
74	W	177	176.94662		7920
74	W	178	177.94585		1866240
74	W	179	178.947072		2223
74	W	180	179.946706	0.12	1e+099
74	W	181	180.948198		10471680
74	W	182	181.948206	26.50	1e+099
74	W	183	182.9502245	14.31	1e+099
74	W	184	183.9509326	30.64	1e+099
74	W	185	184.9534206		6488640
74	W	186	185.954362	28.43	1e+099
74	W	187	186.957158		85392
74	W	188	187.958487		6028992
74	W	189	188.96191		696
74	W	190	189.96318		1800
75	Re	160	159.98149		8.6e-004
75	Re	161	160.97766		3.7e-004
75	Re	162	161.97571		0.107
75	Re	163	162.97197		0.39
75	Re	164	163.97032		1.e-099
75	Re	165	164.967050		1.e-099
75	Re	166	165.96580		1.e-099
75	Re	167	166.96256		3.4
75	Re	168	167.96161		4.4
75	Re	169	168.95883		8.1
75	Re	170	169.95816		9.2
75	Re	171	170.95555		15.2
75	Re	172	171.95529		15
75	Re	173	172.95306		120
75	Re	174	173.95311		144
75	Re	175	174.95139		353.4
75	Re	176	175.95157		318

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
75	Re	177	176.95027		840
75	Re	178	177.95085		792
75	Re	179	178.949980		1170
75	Re	180	179.950790		146.4
75	Re	181	180.950065		71640
75	Re	182	181.95121		230400
75	Re	183	182.950821		6048000
75	Re	184	183.952524		3283200
75	Re	185	184.9529557	37.40	1e+099
75	Re	186	185.954987		321261.12
75	Re	187	186.9557508	62.60	1.3001453512e+018
75	Re	188	187.9581123		61214.4
75	Re	189	188.959228		87480
75	Re	190	189.96182		186
75	Re	191	190.963124		588
75	Re	192	191.96596		16
76	Os	162	161.98382		1.87e-003
76	Os	163	162.98205		5.5e-003
76	Os	164	163.97793		2.1e-002
76	Os	165	164.97648		7.1e-002
76	Os	166	165.97253		0.216
76	Os	167	166.97155		0.81
76	Os	168	167.967830		2.06
76	Os	169	168.967080		3.46
76	Os	170	169.963570		7.46
76	Os	171	170.96304		8.3
76	Os	172	171.96008		19.2
76	Os	173	172.95979		22.4
76	Os	174	173.95712		44
76	Os	175	174.95708		84
76	Os	176	175.95495		216
76	Os	177	176.95505		180
76	Os	178	177.95335		300

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
76	Os	179	178.95395		390
76	Os	180	179.95235		1290
76	Os	181	180.95327		6300
76	Os	182	181.952186		79560
76	Os	183	182.95311		46800
76	Os	184	183.952491	0.02	1e+099
76	Os	185	184.954043		8087040
76	Os	186	185.953838	1.59	6.3113852e+022
76	Os	187	186.9557479	1.96	1e+099
76	Os	188	187.9558360	13.24	1e+099
76	Os	189	188.9581449	16.15	1e+099
76	Os	190	189.958445	26.26	1e+099
76	Os	191	190.960928		1330560
76	Os	192	191.961479	40.78	1e+099
76	Os	193	192.964148		108396
76	Os	194	193.965179		189341556.48
76	Os	195	194.96812		390
76	Os	196	195.969620		2094
77	Ir	165	164.98758		1.e-099
77	Ir	166	165.98551		1.05e-002
77	Ir	167	166.98154		3.52e-002
77	Ir	168	167.97997		0.161
77	Ir	169	168.97639		0.78
77	Ir	170	169.97503		0.91
77	Ir	171	170.97178		3.6
77	Ir	172	171.97064		4.4
77	Ir	173	172.96771		9.0
77	Ir	174	173.96680		7.9
77	Ir	175	174.96428		9
77	Ir	176	175.96351		8.3
77	Ir	177	176.96117		30
77	Ir	178	177.96108		12
77	Ir	179	178.95915		79

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
77	Ir	180	179.95925		90
77	Ir	181	180.95764		294
77	Ir	182	181.95813		900
77	Ir	183	182.95681		3480
77	Ir	184	183.95739		11124
77	Ir	185	184.95659		51840
77	Ir	186	185.957951		59904
77	Ir	187	186.957361		37800
77	Ir	188	187.958852		149400
77	Ir	189	188.958716		1140480
77	Ir	190	189.96059		1017792
77	Ir	191	190.960591	37.3	1e+099
77	Ir	192	191.962602		6378652.8
77	Ir	193	192.962924	62.7	1e+099
77	Ir	194	193.965076		69408
77	Ir	195	194.965977		9000
77	Ir	196	195.968380		52
77	Ir	197	196.969636		348
77	Ir	198	197.97228		8
77	Ir	199	198.973790		1.e-099
78	Pt	168	167.98804		2.e-003
78	Pt	169	168.98642		3.7e-003
78	Pt	170	169.98233		1.38e-002
78	Pt	171	170.98125		4.4e-002
78	Pt	172	171.977380		9.84e-002
78	Pt	173	172.97650		0.365
78	Pt	174	173.972811		0.889
78	Pt	175	174.97228		2.52
78	Pt	176	175.96900		6.33
78	Pt	177	176.96845		10.6
78	Pt	178	177.96571		21.1
78	Pt	179	178.96548		21.2
78	Pt	180	179.96322		52

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
78	Pt	181	180.96318		52.0
78	Pt	182	181.96127		132
78	Pt	183	182.96173		390
78	Pt	184	183.95990		1038
78	Pt	185	184.96075		4254
78	Pt	186	185.959430		7488
78	Pt	187	186.96056		8460
78	Pt	188	187.959396		881280
78	Pt	189	188.960832		39132
78	Pt	190	189.959930	0.014	2.05120019e+019
78	Pt	191	190.961685		242092.8
78	Pt	192	191.961035	0.782	1e+099
78	Pt	193	192.962985		1577846304
78	Pt	194	193.962664	32.967	1e+099
78	Pt	195	194.964774	33.832	1e+099
78	Pt	196	195.964935	25.242	1e+099
78	Pt	197	196.967323		71609.4
78	Pt	198	197.967876	7.163	1e+099
78	Pt	199	198.970576		1848
78	Pt	200	199.971424		45000
78	Pt	201	200.974500		150
78	Pt	202	201.97574		158400
79	Au	171	170.99177		3.e-005
79	Au	172	171.99011		4.7e-003
79	Au	173	172.98640		2.5e-002
79	Au	174	173.98492		0.139
79	Au	175	174.98155		1.e-099
79	Au	176	175.98027		1.08
79	Au	177	176.97722		1.46
79	Au	178	177.97598		2.6
79	Au	179	178.97341		7.1
79	Au	180	179.97240		8.1
79	Au	181	180.96995		13.7

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
79	Au	182	181.96962		15.5
79	Au	183	182.96762		42.8
79	Au	184	183.96747		20.6
79	Au	185	184.96581		255
79	Au	186	185.96600		642
79	Au	187	186.96456		504
79	Au	188	187.96509		530.4
79	Au	189	188.96389		1722
79	Au	190	189.964699		2568
79	Au	191	190.963650		11448
79	Au	192	191.964810		17784
79	Au	193	192.964132		63540
79	Au	194	193.965339		136872
79	Au	195	194.965018		16079040
79	Au	196	195.966551		532820.16
79	Au	197	196.966552	100	1e+099
79	Au	198	197.968225		232862.688
79	Au	199	198.968748		271209.6
79	Au	200	199.970720		2904
79	Au	201	200.971641		1560
79	Au	202	201.97379		28.8
79	Au	203	202.975137		53
79	Au	204	203.97771		39.8
79	Au	205	204.97961		31
80	Hg	175	174.99141		1.08e-002
80	Hg	176	175.987410		2.04e-002
80	Hg	177	176.98634		0.1273
80	Hg	178	177.982476		0.269
80	Hg	179	178.98178		1.09
80	Hg	180	179.97832		2.56
80	Hg	181	180.97781		3.6
80	Hg	182	181.97475		10.83
80	Hg	183	182.97456		9.4

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
80	Hg	184	183.97190		30.6
80	Hg	185	184.97198		49.1
80	Hg	186	185.96946		82.8
80	Hg	187	186.96979		114
80	Hg	188	187.96756		195
80	Hg	189	188.96813		456
80	Hg	190	189.96628		1200
80	Hg	191	190.967060		2940
80	Hg	192	191.96557		17460
80	Hg	193	192.966644		13680
80	Hg	194	193.965382		13885047475.2
80	Hg	195	194.966640		37908
80	Hg	196	195.965815	0.15	1e+099
80	Hg	197	196.967195		233784
80	Hg	198	197.966752	9.97	1e+099
80	Hg	199	198.968262	16.87	1e+099
80	Hg	200	199.968309	23.10	1e+099
80	Hg	201	200.970285	13.18	1e+099
80	Hg	202	201.970626	29.86	1e+099
80	Hg	203	202.972857		4027276.8
80	Hg	204	203.973476	6.87	1e+099
80	Hg	205	204.976056		312
80	Hg	206	205.977499		489
80	Hg	207	206.98258		174
80	Hg	208	207.98594		2520
81	Tl	177	176.99688		1.8e-002
81	Tl	178	177.99523		0.255
81	Tl	179	178.99147		0.27
81	Tl	180	179.99019		1.5
81	Tl	181	180.98690		3.2
81	Tl	182	181.98561		2.0
81	Tl	183	182.98270		6.9
81	Tl	184	183.98176		9.7

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
81	Tl	185	184.97910		19.5
81	Tl	186	185.97855		1.e-099
81	Tl	187	186.97617		~51
81	Tl	188	187.97592		71
81	Tl	189	188.97369		138
81	Tl	190	189.97379		156
81	Tl	191	190.97189		1.e-099
81	Tl	192	191.97214		576
81	Tl	193	192.97055		1296
81	Tl	194	193.97105		1980
81	Tl	195	194.96965		4176
81	Tl	196	195.97052		6624
81	Tl	197	196.969540		10224
81	Tl	198	197.970470		19080
81	Tl	199	198.96981		26712
81	Tl	200	199.970945		93960
81	Tl	201	200.970804		262483.2
81	Tl	202	201.972091		1056672
81	Tl	203	202.972329	29.524	1e+099
81	Tl	204	203.973849		119285180.5824
81	Tl	205	204.974412	70.476	1e+099
81	Tl	206	205.976095		252
81	Tl	207	206.977408		286.2
81	Tl	208	207.982005		183.18
81	Tl	209	208.985349		129.66
81	Tl	210	209.990066		78
82	Pb	181	180.99671		4.5e-002
82	Pb	182	181.992676		6.e-002
82	Pb	183	182.99193		0.535
82	Pb	184	183.98820		0.49
82	Pb	185	184.98758		6.3
82	Pb	186	185.98430		4.82
82	Pb	187	186.98403		15.2

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
82	Pb	188	187.98106		25.5
82	Pb	189	188.98088		51
82	Pb	190	189.97818		71
82	Pb	191	190.97820		79.8
82	Pb	192	191.97576		210
82	Pb	193	192.97608		1.e-099
82	Pb	194	193.97397		720
82	Pb	195	194.97447		0
82	Pb	196	195.97271		2220
82	Pb	197	196.97338		480
82	Pb	198	197.97198		8640
82	Pb	199	198.972910		5400
82	Pb	200	199.971816		77400
82	Pb	201	200.972850		33588
82	Pb	202	201.972144		1656738615000
82	Pb	203	202.973375		186742.8
82	Pb	204	203.973029	1.4	1e+099
82	Pb	205	204.974467		482820967800000
82	Pb	206	205.974449	24.1	1e+099
82	Pb	207	206.975881	22.1	1e+099
82	Pb	208	207.976636	52.4	1e+099
82	Pb	209	208.981075		11710.8
82	Pb	210	209.984173		700563758.976
82	Pb	211	210.988731		2166
82	Pb	212	211.9918875		38304
82	Pb	213	212.99650		612
82	Pb	214	213.9997981		1608
83	Bi	185	184.99771		1.e-099
83	Bi	186	185.99648		1.48e-002
83	Bi	187	186.99346		3.2e-002
83	Bi	188	187.99217		4.4e-002
83	Bi	189	188.98951		0.674
83	Bi	190	189.98852		6.3

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
83	Bi	191	190.98605		12.3
83	Bi	192	191.98537		34.6
83	Bi	193	192.98306		67
83	Bi	194	193.98275		95
83	Bi	195	194.98075		183
83	Bi	196	195.98061		306
83	Bi	197	196.97893		558
83	Bi	198	197.97902		618
83	Bi	199	198.97758		1620
83	Bi	200	199.97814		2184
83	Bi	201	200.976970		6480
83	Bi	202	201.977670		6192
83	Bi	203	202.976868		42336
83	Bi	204	203.977805		40392
83	Bi	205	204.977375		1322784
83	Bi	206	205.978483		539395.2
83	Bi	207	206.978455		1038222868.032
83	Bi	208	207.979727		11612948768000
83	Bi	209	208.980383	100	5.99581594e+026
83	Bi	210	209.984105		433036.8
83	Bi	211	210.987258		128.4
83	Bi	212	211.991272		3633
83	Bi	213	212.994375		2735.4
83	Bi	214	213.998699		1194
83	Bi	215	215.00183		456
83	Bi	216	216.00620		130.2
84	Po	190	189.99511		2.46e-003
84	Po	191	190.99465		2.2e-002
84	Po	192	191.99152		3.22e-002
84	Po	193	192.99110		0.42
84	Po	194	193.98828		0.392
84	Po	195	194.98805		4.64
84	Po	196	195.98551		5.56

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
84	Po	197	196.98557		53.6
84	Po	198	197.98334		106.2
84	Po	199	198.98360		328.8
84	Po	200	199.98174		690
84	Po	201	200.98221		918
84	Po	202	201.98070		2682
84	Po	203	202.981410		2202
84	Po	204	203.980307		12708
84	Po	205	204.981170		5976
84	Po	206	205.980465		760320
84	Po	207	206.981578		20880
84	Po	208	207.981231		91451971.77984
84	Po	209	208.982416		3218806460.16
84	Po	210	209.982857		11955686.4
84	Po	211	210.986637		0.516
84	Po	212	211.988852		2.99e-007
84	Po	213	212.992843		4.2e-006
84	Po	214	213.995186		1.643e-004
84	Po	215	214.999415		1.781e-003
84	Po	216	216.0019052		0.145
84	Po	217	217.00625		1.47
84	Po	218	218.0089658		186
85	At	193	193.00019		0
85	At	194	193.99897		0
85	At	195	194.99655		0.328
85	At	196	195.99570		0.253
85	At	197	196.99329		0.35
85	At	198	197.99275		4.2
85	At	199	198.99063		7.2
85	At	200	199.99029		43.2
85	At	201	200.98849		85
85	At	202	201.98845		184
85	At	203	202.98685		444

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
85	At	204	203.98726		552
85	At	205	204.986040		1572
85	At	206	205.986600		1836
85	At	207	206.985776		6480
85	At	208	207.986583		5868
85	At	209	208.986159		19476
85	At	210	209.987131		29160
85	At	211	210.987481		25970.4
85	At	212	211.990735		0.314
85	At	213	212.992921		1.25e-007
85	At	214	213.996356		5.58e-007
85	At	215	214.998641		1.e-004
85	At	216	216.002409		3.e-004
85	At	217	217.004710		3.23e-002
85	At	218	218.008681		1.5
85	At	219	219.011300		56
85	At	220	220.01530		222.6
85	At	221	221.01814		138
85	At	222	222.02233		54
85	At	223	223.02534		50
86	Rn	196	196.00231		4.7e-003
86	Rn	197	197.00166		6.6e-002
86	Rn	198	197.99878		6.5e-002
86	Rn	199	198.99831		0.62
86	Rn	200	199.99568		1.03
86	Rn	201	200.99554		7.0
86	Rn	202	201.99322		9.94
86	Rn	203	202.99332		43.5
86	Rn	204	203.99137		74.4
86	Rn	205	204.99167		168
86	Rn	206	205.99016		340.2
86	Rn	207	206.990730		555
86	Rn	208	207.989631		1461

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
86	Rn	209	208.990380		1710
86	Rn	210	209.989680		8640
86	Rn	211	210.990585		52560
86	Rn	212	211.990689		1434
86	Rn	213	212.993868		1.95e-002
86	Rn	214	213.995346		2.7e-007
86	Rn	215	214.998729		2.3e-006
86	Rn	216	216.000258		4.5e-005
86	Rn	217	217.003915		5.4e-004
86	Rn	218	218.005586		3.5e-002
86	Rn	219	219.009475		3.96
86	Rn	220	220.0113841		55.6
86	Rn	221	221.01546		1500
86	Rn	222	222.0175705		330350.4
86	Rn	223	223.02179		1458
86	Rn	224	224.02409		6420
86	Rn	225	225.02844		279.6
86	Rn	226	226.03089		444
86	Rn	227	227.03541		20.8
86	Rn	228	228.03808		65
87	Fr	200	200.00650		2.4e-002
87	Fr	201	201.00399		6.1e-002
87	Fr	202	202.00329		0.29
87	Fr	203	203.00105		0.55
87	Fr	204	204.00059		1.7
87	Fr	205	204.99866		3.85
87	Fr	206	205.99849		~16
87	Fr	207	206.99686		14.8
87	Fr	208	207.997130		59.1
87	Fr	209	208.995920		50.0
87	Fr	210	209.996398		190.8
87	Fr	211	210.995529		186
87	Fr	212	211.996195		1200

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
87	Fr	213	212.996175		34.6
87	Fr	214	213.998955		5.e-003
87	Fr	215	215.000326		8.6e-008
87	Fr	216	216.003188		7.e-007
87	Fr	217	217.004616		1.68e-005
87	Fr	218	218.007563		1.e-003
87	Fr	219	219.009241		2.e-002
87	Fr	220	220.012313		27.4
87	Fr	221	221.014246		294
87	Fr	222	222.017544		852
87	Fr	223	223.0197307		1320
87	Fr	224	224.023240		199.8
87	Fr	225	225.025607		240
87	Fr	226	226.02934		49
87	Fr	227	227.03183		148.2
87	Fr	228	228.03572		38
87	Fr	229	229.03843		50.2
87	Fr	230	230.04251		19.1
87	Fr	231	231.04541		17.6
87	Fr	232	232.04965		5
88	Ra	203	203.00921		4.e-003
88	Ra	204	204.00648		6.e-002
88	Ra	205	205.00619		0.22
88	Ra	206	206.00378		0.24
88	Ra	207	207.00373		1.3
88	Ra	208	208.00178		1.3
88	Ra	209	209.00194		4.6
88	Ra	210	210.00045		3.7
88	Ra	211	211.000890		13
88	Ra	212	211.999783		13.0
88	Ra	213	213.000350		164.4
88	Ra	214	214.000091		2.46
88	Ra	215	215.002704		1.55e-003

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
88	Ra	216	216.003518		1.82e-007
88	Ra	217	217.006306		1.63e-006
88	Ra	218	218.007124		2.56e-005
88	Ra	219	219.010069		1.e-002
88	Ra	220	220.011015		1.79e-002
88	Ra	221	221.013908		28
88	Ra	222	222.015362		38.0
88	Ra	223	223.018497		987552
88	Ra	224	224.0202020		316224
88	Ra	225	225.023604		1287360
88	Ra	226	226.0254026		50491081600
88	Ra	227	227.0291707		2532
88	Ra	228	228.0310641		181452324.96
88	Ra	229	229.034820		240
88	Ra	230	230.037080		5580
88	Ra	231	231.04122		103
88	Ra	232	232.04369		250
88	Ra	233	233.04800		30
88	Ra	234	234.05055		30
89	Ac	207	207.01209		3.1e-002
89	Ac	208	208.01149		9.7e-002
89	Ac	209	209.00957		9.2e-002
89	Ac	210	210.00926		0.35
89	Ac	211	211.00765		0.213
89	Ac	212	212.00781		0.92
89	Ac	213	213.006570		0.731
89	Ac	214	214.006890		8.2
89	Ac	215	215.006450		0.17
89	Ac	216	216.008721		4.4e-004
89	Ac	217	217.009333		6.9e-008
89	Ac	218	218.011630		1.08e-006
89	Ac	219	219.012400		1.18e-005
89	Ac	220	220.014750		2.636e-002

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
89	Ac	221	221.015580		5.2e-002
89	Ac	222	222.017829		5.0
89	Ac	223	223.019126		126
89	Ac	224	224.021708		10008
89	Ac	225	225.023221		864000
89	Ac	226	226.026090		105732
89	Ac	227	227.0277470		687057394.61376
89	Ac	228	228.0310148		22140
89	Ac	229	229.032930		3762
89	Ac	230	230.03603		122
89	Ac	231	231.03855		450
89	Ac	232	232.04202		119
89	Ac	233	233.04455		145
89	Ac	234	234.04842		44
89	Ac	235	235.05110		1.e-099
89	Ac	236	236.05518		1.e-099
90	Th	210	210.01503		1.7e-002
90	Th	211	211.01486		4.8e-002
90	Th	212	212.01292		3.6e-002
90	Th	213	213.01296		0.14
90	Th	214	214.01145		0.1
90	Th	215	215.011730		1.2
90	Th	216	216.011051		2.68e-002
90	Th	217	217.013070		2.4e-004
90	Th	218	218.013268		1.09e-007
90	Th	219	219.015520		1.05e-006
90	Th	220	220.015733		9.7e-006
90	Th	221	221.018171		1.68e-003
90	Th	222	222.018454		2.05e-003
90	Th	223	223.020795		0.6
90	Th	224	224.021459		1.05
90	Th	225	225.023941		523.2
90	Th	226	226.024891		1834.2

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
90	Th	227	227.027699		1613952
90	Th	228	228.0287313		60324219.894528
90	Th	229	229.031755		231627836840
90	Th	230	230.0331266		2378761081880
90	Th	231	231.0362971		91872
90	Th	232	232.0380504	100	4.433748103e+017
90	Th	233	233.0415769		1338
90	Th	234	234.043595		2082240
90	Th	235	235.047500		432
90	Th	236	236.04971		2250
90	Th	237	237.05389		288
90	Th	238	238.05624		564
91	Pa	213	213.02118		7.e-003
91	Pa	214	214.02074		1.7e-002
91	Pa	215	215.01910		1.4e-002
91	Pa	216	216.01911		0.105
91	Pa	217	217.018290		3.48e-003
91	Pa	218	218.020010		1.13e-004
91	Pa	219	219.019880		5.3e-008
91	Pa	220	220.021880		7.8e-007
91	Pa	221	221.021860		5.9e-006
91	Pa	222	222.023730		3.2e-003
91	Pa	223	223.023960		5.1e-003
91	Pa	224	224.025610		0.844
91	Pa	225	225.026120		1.7
91	Pa	226	226.027933		108
91	Pa	227	227.028793		2298
91	Pa	228	228.031037		79200
91	Pa	229	229.032089		129600
91	Pa	230	230.034533		1503360
91	Pa	231	231.0358789	100	1033804895760
91	Pa	232	232.038582		113184
91	Pa	233	233.0402402		2329948.8

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
91	Pa	234	234.043302		24120
91	Pa	235	235.045440		1466.4
91	Pa	236	236.04868		546
91	Pa	237	237.05114		522
91	Pa	238	238.054500		136.2
91	Pa	239	239.05713		6480
91	Pa	240	240.06098		1.e-099
92	U	218	218.02349		6.e-003
92	U	219	219.024920		5.5e-005
92	U	220	220.02471		1.e-099
92	U	221	221.02635		1.e-099
92	U	222	222.02607		1.4e-006
92	U	223	223.027720		2.1e-005
92	U	224	224.027590		9.4e-004
92	U	225	225.029380		6.1e-002
92	U	226	226.029340		0.269
92	U	227	227.031140		66
92	U	228	228.031366		546
92	U	229	229.033496		3480
92	U	230	230.033927		1797120
92	U	231	231.036289		362880
92	U	232	232.0371463		2174272206.912
92	U	233	233.039628		5023862619200
92	U	234	234.0409456	0.0055	7747225333000
92	U	235	235.0439231	0.7200	2.2216075904e+016
92	U	236	236.0455619		739063206920000
92	U	237	237.0487240		583200
92	U	238	238.0507826	99.2745	1.40996345368e+017
92	U	239	239.0542878		1407
92	U	240	240.056586		50760
92	U	241	241.06033		1.e-099
92	U	242	242.06293		1008
93	Np	225	225.033900		1.e-099

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
93	Np	226	226.03513		3.5e-002
93	Np	227	227.034960		0.51
93	Np	228	228.03618		61.4
93	Np	229	229.036250		240
93	Np	230	230.037810		276
93	Np	231	231.038230		2928
93	Np	232	232.04010		882
93	Np	233	233.040730		2172
93	Np	234	234.042889		380160
93	Np	235	235.0440559		34223040
93	Np	236	236.046560		4859766604000
93	Np	237	237.0481673		67658049344000
93	Np	238	238.0509405		182908.8
93	Np	239	239.0529314		203558.4
93	Np	240	240.056169		3714
93	Np	241	241.058250		834
93	Np	242	242.06164		132
93	Np	243	243.064270		111
93	Np	244	244.06785		137.4
94	Pu	228	228.038730		1.e-099
94	Pu	229	229.040140		120
94	Pu	230	230.039646		102
94	Pu	231	231.04126		516
94	Pu	232	232.041179		2022
94	Pu	233	233.042990		1254
94	Pu	234	234.043305		31680
94	Pu	235	235.045282		1518
94	Pu	236	236.0460481		90189694.73664
94	Pu	237	237.0484038		3905280
94	Pu	238	238.0495534		2767542417.216
94	Pu	239	239.0521565		760837485860
94	Pu	240	240.0538075		207139662264
94	Pu	241	241.0568453		452841889.248

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
94	Pu	242	242.0587368		11833847250000
94	Pu	243	243.061997		17841.6
94	Pu	244	244.064198		2.52455408e+015
94	Pu	245	245.067739		37800
94	Pu	246	246.070198		936576
94	Pu	247	247.07407		196128
95	Am	231	231.04556		1.e-099
95	Am	232	232.04659		78.6
95	Am	233	233.04647		192
95	Am	234	234.04779		139.2
95	Am	235	235.04803		594
95	Am	236	236.04957		1.e-099
95	Am	237	237.049970		4380
95	Am	238	238.051980		5880
95	Am	239	239.053018		42840
95	Am	240	240.055288		182880
95	Am	241	241.0568229		13638903451.776
95	Am	242	242.0595430		57672
95	Am	243	243.0613727		232574544620
95	Am	244	244.0642794		36360
95	Am	245	245.066445		7380
95	Am	246	246.069768		2340
95	Am	247	247.07209		1380
95	Am	248	248.07575		1.e-099
95	Am	249	249.07848		1.e-099
96	Cm	233	233.05080		1.e-099
96	Cm	234	234.05024		51
96	Cm	235	235.05159		1.e-099
96	Cm	236	236.05141		1.e-099
96	Cm	237	237.05289		1.e-099
96	Cm	238	238.053020		8640
96	Cm	239	239.05495		0
96	Cm	240	240.0555190		2332800

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
96	Cm	241	241.0576467		2833920
96	Cm	242	242.0588293		14065920
96	Cm	243	243.0613822		918306548.928
96	Cm	244	244.0627463		571180362.048
96	Cm	245	245.0654856		268233871000
96	Cm	246	246.0672176		150210967760
96	Cm	247	247.070347		492288045600000
96	Cm	248	248.072342		10981810248000
96	Cm	249	249.075947		3849
96	Cm	250	250.078351		1.e-099
96	Cm	251	251.082278		1008
96	Cm	252	252.08487		86400
97	Bk	235	235.05658		1.e-099
97	Bk	236	236.05733		1.e-099
97	Bk	237	237.05713		1.e-099
97	Bk	238	238.05827		144
97	Bk	239	239.05836		1.e-099
97	Bk	240	240.05975		288
97	Bk	241	241.06022		276
97	Bk	242	242.06205		420
97	Bk	243	243.063002		16200
97	Bk	244	244.065168		15660
97	Bk	245	245.0663554		426816
97	Bk	246	246.068670		155520
97	Bk	247	247.070299		43548557880
97	Bk	248	248.073080		284012334.72
97	Bk	249	249.074980		28512000
97	Bk	250	250.078311		11563.2
97	Bk	251	251.080753		3336
97	Bk	252	252.08430		108
97	Bk	253	253.08688		1.e-099
97	Bk	254	254.09060		1.e-099
98	Cf	237	237.06207		2.1

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
98	Cf	238	238.06141		2.11e-002
98	Cf	239	239.06258		60
98	Cf	240	240.06230		63.6
98	Cf	241	241.06372		228
98	Cf	242	242.063690		209.4
98	Cf	243	243.06542		642
98	Cf	244	244.065990		1164
98	Cf	245	245.06804		2700
98	Cf	246	246.0687988		128520
98	Cf	247	247.070992		11196
98	Cf	248	248.072178		28857600
98	Cf	249	249.074847		11076481054.08
98	Cf	250	250.0764000		412764593.1264
98	Cf	251	251.079580		28401233472
98	Cf	252	252.081620		83468069.4816
98	Cf	253	253.085127		1538784
98	Cf	254	254.087316		5227200
98	Cf	255	255.09104		5100
98	Cf	256	256.09344		738
99	Es	240	240.06892		1.e-099
99	Es	241	241.06866		10
99	Es	242	242.06970		13.5
99	Es	243	243.06963		21
99	Es	244	244.07097		37
99	Es	245	245.07132		66
99	Es	246	246.07297		462
99	Es	247	247.073650		276
99	Es	248	248.075460		1620
99	Es	249	249.076410		6132
99	Es	250	250.07865		30960
99	Es	251	251.079984		118800
99	Es	252	252.082970		40754880
99	Es	253	253.084818		1768608

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
99	Es	254	254.088016		23820480
99	Es	255	255.090266		3438720
99	Es	256	256.09359		1524
99	Es	257	257.09598		665280
100	Fm	242	242.07343		8.e-004
100	Fm	243	243.07451		0.21
100	Fm	244	244.07408		3.3e-003
100	Fm	245	245.07538		4.2
100	Fm	246	246.075280		1.1
100	Fm	247	247.07682		35
100	Fm	248	248.077184		36
100	Fm	249	249.07902		156
100	Fm	250	250.079515		1800
100	Fm	251	251.081566		19080
100	Fm	252	252.082460		91404
100	Fm	253	253.085176		259200
100	Fm	254	254.086848		11664
100	Fm	255	255.089955		72252
100	Fm	256	256.091767		9456
100	Fm	257	257.095099		8683200
100	Fm	258	258.09707		3.7e-004
100	Fm	259	259.10059		1.5
101	Md	245	245.08102		9.e-004
101	Md	246	246.08193		1.0
101	Md	247	247.08180		0.27
101	Md	248	248.08291		7
101	Md	249	249.08300		24
101	Md	250	250.08449		52
101	Md	251	251.08492		240
101	Md	252	252.08663		138
101	Md	253	253.08728		720
101	Md	254	254.08973		600
101	Md	255	255.091075		1620

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
101	Md	256	256.094050		4620
101	Md	257	257.095535		19872
101	Md	258	258.098425		4449600
101	Md	259	259.10050		5760
101	Md	260	260.10365		2401920
102	No	249	249.08782		5.7e-005
102	No	250	250.08749		5.7e-006
102	No	251	251.08896		0.76
102	No	252	252.088966		2.44
102	No	253	253.09065		97.2
102	No	254	254.090949		51
102	No	255	255.093232		186
102	No	256	256.094276		2.91
102	No	257	257.096850		25
102	No	258	258.09820		1.2e-003
102	No	259	259.10102		3480
102	No	260	260.10264		0.106
102	No	261	261.10574		1.e-099
102	No	262	262.10752		0
103	Lr	251	251.09436		1.e-099
103	Lr	252	252.09533		0.39
103	Lr	253	253.09526		0.58
103	Lr	254	254.09659		13
103	Lr	255	255.09677		22
103	Lr	256	256.09876		27
103	Lr	257	257.09961		0.646
103	Lr	258	258.10188		4.1
103	Lr	259	259.102990		6.2
103	Lr	260	260.10557		180
103	Lr	261	261.10694		2340
103	Lr	262	262.10969		0
103	Lr	263	263.11139		1.e-099
104	Rf	253	253.10068		1.3e-002

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
104	Rf	254	254.10017		2.3e-005
104	Rf	255	255.10149		1.64
104	Rf	256	256.101180		6.45e-003
104	Rf	257	257.10307		4.7
104	Rf	258	258.10357		1.2e-002
104	Rf	259	259.105630		2.8
104	Rf	260	260.10643		2.1e-002
104	Rf	261	261.10875		5.5
104	Rf	262	262.10992		2.3
104	Rf	263	263.11254		660
104	Rf	264	264.11398		1.e-099
105	Db	255	255.10740		1.7
105	Db	256	256.10811		1.9
105	Db	257	257.10786		1.53
105	Db	258	258.10944		4.5
105	Db	259	259.10972		0.51
105	Db	260	260.11143		1.52
105	Db	261	261.11211		1.8
105	Db	262	262.11415		35
105	Db	263	263.11508		29
105	Db	264	264.11747		1.e-099
105	Db	265	265.11866		1.e-099
106	Sg	258	258.11315		3.3e-003
106	Sg	259	259.11465		0.58
106	Sg	260	260.114440		3.8e-003
106	Sg	261	261.11620		0.23
106	Sg	262	262.11648		8.e-003
106	Sg	263	263.11831		1.0
106	Sg	264	264.11892		1.e-099
106	Sg	265	265.12107		8
106	Sg	266	266.12193		21
107	Bh	260	260.12180		1.e-099
107	Bh	261	261.12180		1.3e-002

Appendix E: The Direct Default Isotopedata Definition File

AtomicNumber	AtomicSymbol	MassNumber	AtomicMass	IsotopicComposition	HalfLife
107	Bh	262	262.12301		0.29
107	Bh	263	263.12315		1.e-099
107	Bh	264	264.12473		1.3
107	Bh	265	265.12520		1.e-099
107	Bh	266	266.12701		5
107	Bh	267	267.12774		22
108	Hs	263	263.12871		1.e-099
108	Hs	264	264.128410		5.4e-004
108	Hs	265	265.13000		2.1e-003
108	Hs	266	266.13004		2.7e-003
108	Hs	267	267.13177		3.2e-002
108	Hs	268	268.13216		1.e-099
108	Hs	269	269.13411		27
108	Hs	277			2400
109	Mt	265	265.13657		1.e-099
109	Mt	266	266.13794		1.2e-003
109	Mt	267	267.13753		1.e-099
109	Mt	268	268.13882		5.3e-002
109	Mt	269	269.13911		1.e-099
109	Mt	270	270.14072		1.e-099
109	Mt	271	271.14123		1.e-099
110	Ds	267	267.14396		1.e-005
110	Ds	268	268.14353		1.e-099
110	Ds	269	269.14514		2.3e-004
110	Ds	270	270.14463		1.6e-004
110	Ds	271	271.14608		0.21
110	Ds	272	272.14631		1.e-099
110	Ds	273	273.14925		3.6e-004
111	Rg	272	272.15348		2.e-003

Appendix F:

UniProt Converter

The UniProt converter is designed to read a file containing UniProt format records, convert the sequence and most features into an BIOVIA molfile, then register the molfile and selected data from the Uniprot records into an Oracle table.

The converter handles:

- A file containing any number of UniProt records.
- UniProt records that are either in the UniProt (EMBL) flat-file format or in the UniProt XML format

About the UniProt Converter

The converter creates or appends to a table with one or more of the following fields for each record:

- The ID, ID in flat-file, <name> in XML, default column name ID
- The accession number, AC in flat file, <accession> in XML, default column name AC
- The sequence, SQ in flat file, <sequence> in XML, default column name SQ
- The converted binary ctab, or NULL if conversion failed, default column name CTAB
- The full record as text in a CLOB field, default column name UNIPROT
- Conversion error output for the record in a CLOB field, default column name LOG

The conversion default is to write all of these fields, using the column names as shown above. However, the user can specify a file with the `-fields` argument that contains a list of specific fields to be written to the molecule table, with associated column names.

Note: If the table already exists, it must contain the appropriate column names.

In addition, if the `-featuretable` argument is present, the converter writes UniProt feature entries FT in flat-file, <feature> in XML to the specified table. The converter creates or appends to a table one or more of the following fields for each feature entry.

- The ID, ID in flat-file, <name> in XML, default column name ID
- The accession number, AC in flat file, <accession> in XML, default column name AC
- The feature type, part of FT in flat-file, <type> in XML, default column name TYPE
- The feature description, part of FT in flat-file, <description> in XML, default column name DESCRIPTION
- The beginning position, part of FT in flat-file, <location><begin> in XML, default column name BEGIN
- The ending position, part of FT in flat-file, <location><end> in XML, default column name END

If a feature contains only one position entry, that number is written to both the BEGIN and END columns.

The default behavior writes to all fields, using the column names as shown. However, the user can specify a file with the `-fields` argument that contains a list of specific fields that writes to the feature data table, with associated column names.

Note: If the table already exists, it must contain the appropriate column names.

Running the UniProt Converter

The UniProt converter is a command-line utility program. The utility is invoked by running a Windows batch file `UniProtConverter.bat` or a shell script `UniProtConverter`.

Start the UniProt converter by executing the batch file:

On Windows:

```
drive:\BIOVIA\direct\bin<versionNumber>\UniProtConverter.bat
```

On Linux:

```
install_directory/BIOVIA/direct/bin<versionNumber>/UniProtConverter
```

Where `<versionNumber>` refers to the corresponding Oracle version.

Usage

The UniProt converter accepts the following arguments, those in square brackets are optional.

Note: A single argument of `-help` or `-?` prints a brief help message.

```
UniProtConverter [-fields fields-file] [-featuretable feature-data-table [-  
nowait] UniProt-file-name molecule-table-name  
username/password@host:port:SID
```

where:

- `-fields fields-file` specifies an optional file that lists the fields to be written to the molecule table and feature data table.
This can be used to limit the number of fields stored in the tables and to specify the column names to be used. See [Format of Fields File](#) for a description of the fields file. If this option is missing, all fields are written as described in [About the UniProt Converter](#) section.
- `-featuretable feature-data-table` specifies a table name to receive feature data records.
The table is created if it does not exist. If the table does exist it must contain all of the columns that will be written out by the converter. The table name must not include a schema. If this entry is not present, the feature data is not written to a separate table.
- `-nowait` causes the converter to use `COMMIT WRITE NOWAIT BATCH` instead of `COMMIT`.
- `UniProt-file-name` is the name of the input file containing the UniProt records.
- `molecule-table-name` is the name of the molecule table to which molecules are written.
The table is created if it does not exist. If the table does exist, it must contain all of the columns that will be written out by the converter. The table name must not include a schema.
- `username/password@host:port:SID` specifies the Oracle connection string.
Specify the Oracle username and password, followed by JDBC (Java) thin driver connection information. The connection information consists of the host name of the computer on which Oracle is running, the TNS listener port number and the Oracle database service identifier (SID). The host name can be specified as `localhost` to use an Oracle instance on the current computer. The port number is often 1521 but this depends on how many Oracle instances are running on the computer. You can locate the port number and Oracle SID by listing the file `ORACLE_HOME/network/admin/tnsnames.ora`. Contact your Oracle administrator for more information.

Format of Fields File

The optional fields file should contain one entry per line, each entry consists of a keyword followed by a space followed by the desired column name. Blank lines and lines starting with the pound character (#) are ignored. The column name must not contain spaces. The file must contain either an ID entry or an AC entry to identify the record. Valid keywords and their associated default column types for entries that are written to the molecule table are as follows.

Column Name	Valid Keyword	Default Column Type
ID	Name/ID value	VARCHAR2 (80)
AC	Accession number	VARCHAR2 (80)
CTAB	Molecule	BLOB
SQ	Sequence text	CLOB
UNIPROT	Entire UniProt record text	CLOB
LOG	Conversion error information for record	VARCHAR2 (4000)

Valid keywords for fields that are written to the feature data table include ID and AC from, as well as the following formats.

Column Name	Valid Keyword	Default Column Type
BEGIN	Starting position of feature in sequence	NUMBER (9)
END	Ending position of feature in sequence	NUMBER (9)
TYPE	Feature type	VARCHAR2 (80)
DESCRIPTION	Feature description	VARCHAR2 (4000)

Operation

- The UniProt converter determines the type of its input file, XML or flat file, then reads it and converts each record to a BIOVIA molfile plus associated data.
- The converter inserts the molecule record and optional feature data table records.
- The converter commits the transaction after each record is processed.
- The converter prints a status entry every 10 records.
- If a record conversion fails, a message is printed indicating the record that failed. The reason for the failure is written to the LOG column if one is specified in the fields file, or if there is no fields file. Otherwise, the reason for the failure is printed to the terminal. Failures occur because of features that either specify multiple or alternate molecular structures or because a feature is not understood. The conversion errors are generally cryptic but do indicate the feature that was problematic. A record is still written to the molecule table, however, the CTAB column will contain NULL.
- The time it takes to convert a UniProt file to a molfile depends on the size and the complexity of the sequence being converted. The longer the sequence and the more modifications and mutations that it contains, the longer it takes to convert. BIOVIA testing has successfully converted a file that contained 35,000 residues. You should expect conversions of this size to take 10-20 minutes. The conversion time for more modestly sized residues, in the range of 400-1,000 residues takes seconds. Data Sgroups are used to identify undocumented or ambiguous modifications. BIOVIA recommends that you do not routinely search for residues identified with data Sgroups. This type of search is

relatively slow, and some modifications, for example, N-glycosylation, are common in the data set. This can result in slow search times.

Example: Uniprot Conversion

This example covers converting an XML file of lily peptides downloaded from UniProt. Six of the UniProt records fail to process, the molecule table LILIES will contain corresponding structure records, however, the CTAB column value will be NULL. Of these six failures, the first four records code for multiple proteins and the last two contain unknown amino acids that are not converted ("Xaa").

```
$ UniProtConverter xml/liliaceae.xml lilies  
biotest/biotest@localhost:1521:dg102 -fields lily.txt
```

Converting UniProt file

Input file: xml/liliaceae.xml

Molecule table: lilies

Feature data table: (none)

Conversion failed, inserted NULL molstructure for record 8, Q43531

Conversion errors: 8, Accession number Q43531: "ProcessMutation(267, 267, "T", "A", "Loss of calcium/calmodulin-stimulated kinase activity.");"

failed, returned: ProcessMutation attempted to process explicit chemistry.

Processed record 10, Q7M443

Conversion failed, inserted NULL molstructure for record 17, Q402E1

Conversion errors: 17, Accession number Q402E1: "ModifyResidue(10,"N6-methylated lysine");" failed, returned: FALSE

Conversion failed, inserted NULL molstructure for record 18, Q402E2

Conversion errors: 18, Accession number Q402E2: "ModifyResidue(10,"N6-methylated lysine");" failed, returned: FALSE

Conversion failed, inserted NULL molstructure for record 19, Q2Z2F4

Conversion errors: 19, Accession number Q2Z2F4: "ModifyResidue(10,"N6-methylated lysine");" failed, returned: FALSE

Processed record 20, Q9MBF6

Conversion failed, inserted NULL molstructure for record 30, Q9TMC8

Conversion errors: 30, Accession number Q9TMC8: "FinalizeStructure();" failed, returned: invalid templates for AA/Xaa AA/Xaa

Conversion failed, inserted NULL molstructure for record 35, Q9GHC3

Conversion errors: 35, Accession number Q9GHC3: "FinalizeStructure();" failed, returned: invalid templates for AA/Xaa AA/Xaa AA/Xaa AA/Xaa

Processed record 40, O22646

Processed record 50, O24634

Processed record 60, O49078

Processed 60 records, 6 records failed to process

Additional Information on UniProt Structure Conversion

BIOVIAUniProtConverter processes two structural sections of a UniProt record, first the SEQUENCE entry and then the FEATURE TYPE entries.

The SEQUENCE entry in a UniProt record contains the unmodified amino acid sequence. This is processed to convert the sequence into a molecule containing template atoms with one template atom for each amino acid in the original sequence.

After constructing this molecule, the UniProt converter then processes all of the FEATURE TYPE entries. Each FEATURE TYPE entry has one of four outcomes:

- The feature type specifies a modification that is handled in the conversion by replacing the original template atoms with the full structure of the modified amino acids, for example, replacing a serine template atom with the full structure of the phosphoserine molecule. In some cases, a Mod atom is used in lieu of explicit chemistry.
- The feature type does not specify a modification but instead is an annotation to the sequence, and non-structure-differentiating Sgroup data might be added to the molecule to describe this feature.
- The feature type is ignored.
- The feature type specifies a modification that cannot be processed. In this case, the conversion will not process this record, the CTAB column will contain NULL, and the LOG column or program output contains information about the failure.

The UniProt converter recognizes the following FEATURE TYPES. An error occurs if the input file contains a FEATURE TYPE that is not on this list. Most of these FEATURE TYPE are annotations (see the second bullet listed above) and only result in the addition of Sgroup data to the molecule. The remaining types are described in the sections that follow.

FEATURE TYPES

The UniProt Converter recognizes the following FEATURE TYPES. An error occurs if the input file contains a FEATURE TYPE not on the following list.

```
CHAIN
TRANSIT_PEPTIDE
SIGNAL_PEPTIDE
DISULFIDE_BOND
METAL_ION_BINDING_SITE
SEQUENCE_VARIANT
MUTAGENESIS_SITE
SEQUENCE_CONFLICT
HELIX
TURN
STRAND
INITIATOR_METHIONINE
MODIFIED_RESIDUE
SPLICE_VARIANT
REGION_OF_INTEREST
TRANSMEMBRANE_REGION
COMPOSITIONALLY_BIASED_REGION
PROPEPTIDE
BINDING_SITE
DOMAINTOPOLOGICAL_DOMAIN
GLYCOSYLATION_SITE
SHORT_SEQUENCE_MOTIF
REPEAT
SITE
ACTIVE_SITE
NONTERMINAL_RESIDUE
DNA_BINDING_REGIONZINC_FINGER_REGION
COILED_COIL_REGION
NUCLEOTIDE_PHOSPHATE_BINDING_REGION
LIPID_MOIETY_BINDING_REGION
NONCONSECUTIVE_RESIDUES
PEPTIDE
```

CROSSLINK
 CALCIUM_BINDING_REGION
 UNSURE_RESIDUE
 NONSTANDARD_AMINO_ACID

Ignored Types

SEQUENCE_VARIANT
 SEQUENCE_CONFLICT
 SPLICE_VARIANT
 REGION_OF_INTEREST
 COILED_COIL_REGION
 SHORT_SEQUENCE_MOTIF
 COMPOSITIONALLY_BIASED_REGION
 DOMAIN
 TOPOLOGICAL_DOMAIN
 HELIX
 TURN
 STRAND
 METAL_ION_BINDING_SITE
 BINDING_SITE
 SITE
 ACTIVE_SITE
 DNA_BINDING_REGION
 ZINC_FINGER_REGION
 NUCLEOTIDE_PHOSPHATE_BINDING_REGION
 CALCIUM_BINDING_REGION
 UNSURE_RESIDUE
 NONSTANDARD_AMINO_ACID

Structure Modifications

These are the FEATURE TYPES that relate to structural modifications. Besides structural changes to the molecule generated from the original sequence, these FEATURE TYPES are annotated in the molecule using non-structure-differentiating data Sgroups.

FEATURE TYPE	Description
NONTERMINAL_RESIDUE and NONCONSECUTIVE_ RESIDUES	Handled by adding terminal or embedded Mod atoms to the structure to represent the unknown attached amino acids.
CHAIN, TRANSIT_PEPTIDE, SIGNAL_PEPTIDE, PROPEPTIDE, PEPTIDE, and INITIATOR_METHIONINE	Implies breaks in the peptide backbone. The breaks are made, and the annotation is stored as a non-structure-differentiating Data Sgroup.
MUTAGENESIS_SITE	Indicates the change of a residue or range of residues. The substitution is made and the annotation, amended to include the ORIGINAL>MUTANT information, is stored as a non-structure-differentiating Data Sgroup. Currently limited to cases where the original and variant residue strings are of the same length.
MODIFIED_RESIDUE, LIPID_MOIETY_BINDING_	The template atom referenced by the modification is either replaced with a Mod atom or expanded into the structure specified by the

FEATURE TYPE	Description
REGION, and GLYCOSYLATION	modification. In some cases, a Mod atom is added to the structure to represent an addition, as in the case of GLYCOSYLATION described above.
CROSSLINK DISULFIDE_BOND	The two crosslinked template atoms are replaced with expanded structures or Mod atoms and crosslinked.

Sgroups

There are two structure differentiating data Sgroup field names:

- `generic_modification` — labels MOD atoms used by the UniProtConverter for non-coded modifications or crosslinks.
- `Table` — labels MOD atoms used for ill-defined structures.

The non-structure differentiating data Sgroup field names are:

```
chain
crosslink
initiator methionine
modification
modification (glycosylation)
modification (lipid)
mutagenesis site
peptide
propeptide
signal peptide
transit
```

Abbreviation Classes

Abbreviation Sgroups are used to both link explicit atoms to the original residues, and for display. For modified or crosslinked residues the Abbreviation label corresponds to the original residue name. The following Abbreviation classes are used to further differentiate different modifications.

- `AA` — the amino acid class
- `DAA` — the d-amino acid class
- `MODAA` — the modified amino acid class
- `XLINKAA` — the crosslinked amino acid class

Known Limitations

Record Encoding Multiple Structures

As noted in the table under the [Structure Modifications](#), `SEQUENCE_VARIANT`, `SEQUENCE_CONFLICT`, and `SPLICE_VARIANT` result in a UniProt input that corresponds to multiple output protein structures. Because of their ubiquity, these are silently ignored by the conversion utility. However, sometimes the residue specified as a `SEQUENCE_VARIANT` will have another modification. In this case, the record will fail to process.

Multiple Modifications to a Single Residue

Three examples of this limitation are glycosylations of 2-hydroxyproline, 4-hydroxyproline, and 5-hydroxylysine.

Crosslinks Involving 3 Residues

These modifications, which are represented in UniProt format as two separate crosslinks with notation as the third residue, are not converted: Piperidine-2,5-dicarboxylic acid, Pyridine-2,5-dicarboxylic acid, and Tryptophyl-tyrosyl-methioninium.

Creating a Sample Proteins Database

You can create your own database of reviewed liliacea proteins as follows.

1. Connect your Web browser to:
`http://www.uniprot.org/uniprot/?query=taxonomy%3A%22Liliaceae+%5B4677%5D%22+AND+reviewed%3Ayes&sort=score`
2. Click the **Download...** link on the upper right side of the page.
3. On the Download page, right-click the **Download** link to download a file in XML format, which is under this text:
XML
Complete data in XML format.
4. **Save Target As...** (Internet Explorer) or **Save Link As...** (Firefox) to save the contents as a file.
5. Specify a file name and location, and click.
6. Start the Direct UniProt File Conversion Utility and provide as the first argument the name of the XML file you saved in Step 5.

In the following example, the table name is LILY and uses all the default column names.

On Unix:

```
$ /opt/BIOVIA/direct/bin<versionNumber>/UniProtConverter \  
  file-name-saved-in-step-5\  
  lily \  
  oracle-username/password@host:port:SID
```

On Windows:

```
C:\>BIOVIA\direct\bin<versionNumber>\UniProtConverter ^  
  file-name-saved-in-step-5 ^  
  lily ^      oracle-username/password@host:port:SID
```

Specify an Oracle user name and password for a schema that has been enabled for Direct.

Appendix G:

Direct Boost Files

Direct includes functionality that stores often-used data to disk files external to Oracle. These files are referred to as boost files because they boost performance of substructure (SSS, RSS) and similarity searches. Depending on the database hardware configuration, external boost files can provide a significant performance gain for substructure and similarity searches.

The boost files must be located on the same computer as the one running Oracle. On a RAC system, the boost files need to be on a file system that is shared by all of the computers which make up the RAC.

Rather than reading the boost files directly using standard file I/O calls, the boost files are mapped into memory and accessed as if they were part of the Direct address space. This has the advantage that multiple Direct extproc processes can use the same mapped memory.

Storing in Boost Files

Direct 2021 can use boost files to stores the following data.

- Fastsearch — each segment (up to 2 GB) is placed in a separate file.

The SSS performance boost on a computer with fast disks and a large amount of memory allocated to Oracle is moderate. A speed increase of 35% is typical and some queries execute up to 3 times faster. Computers with relatively small amount of memory allocated to Oracle see even larger performance gains.

- Inverted 2D molecule or reaction key data — each chunk, up to approximately 45 MB, is placed in a separate file.

Key data is used for substructure (SSS or RSS) searches when the query contains data or polymer Sgroups, and for all similarity searches. Typically this means one file for each 128000 molecule or 64000 reactions. The performance boost provided by storing key data in a file is large for similarity searches, small for substructure searches. If similarity searching is not important, consider using boost files for connection table data rather than keys.

- Molecule or reaction connection tables are placed into a single file with a second file providing an index into the first file.

Connection tables are fetched for verification during key-based substructure (SSS or RSS) searches when the query contains data or polymer Sgroups. The single file will be as large the molecules or reactions in the table. A typical small molecule is about 1000 to 1500 bytes in size, and a typical reaction is about 4000 to 6000 bytes in size. Bio-molecules and Markush molecules can be much larger than 1000 bytes. Storing connection table data in a boost file provides a large performance gain for substructure (SSS, RSS) searches when the query contains data or polymer Sgroups.

Note: Any search speed increase provided by boost files depends greatly on the hardware and software configuration at your site. The more memory allocated to the Oracle buffer cache, the smaller the gain will be when using boost files.

Adding Boost Files to a Molecule or Reaction Index

When you first create a Direct molecule or reaction index, there will not be any boost files used by the cartridge. All data will be fetched from Oracle as needed. If you want to use boost files to improve search performance, you need to do two things:

1. Create a directory to contain the boost files. This directory must be accessible and editable by the extproc process. Normally the extproc process is started by the Oracle server. However, if you have created a special Oracle listener that starts up from a non-privileged account, the extproc process is started by that non-privileged account. The account starts the extproc process must be able to create, write, and read files in the boost file directory.
2. Use the Direct cartridge procedure `MDLAUX.CREATEBOOSTFILE` to create boost files for any or all of Fastsearch data, inverted keys data or binary molecule or reaction connection tables.

Location of Boost Files

Boost files are stored in a directory that you specify. The directory path can be specified:

- in the `MDLAUX.CREATEBOOSTFILE` command
- as a parameter to the `CREATE INDEX` or `ALTER INDEX` command
- globally using `MDLAUX.SETPROPERTY`

The boost file directory must be accessible and editable by the extproc process. Normally the extproc process is started by the Oracle server. However, if you have created a special Oracle listener that starts up from a non-privileged account, the extproc process will be started by that non-privileged account. The account starts the extproc process must be able to create, write, and read files in the boost file directory.

To specify the location of boost files in the `MDLAUX.CREATEBOOSTFILE` procedure, include `BOOSTDIR=directory-path` in the argument to the procedure, for example:

```
execute mdlaux.createboostfile('moltable',  
'fastsearch boostdir=/opt/BIOVIA/boostfiles');
```

For more information, see [MDLAUX.CREATEBOOSTFILE](#). This directory name overrides any name specified in `CREATE INDEX`, `ALTER INDEX`, or with `MDLAUX.SETPROPERTY`.

To specify the location of boost files when you create an index (or alter the index), add `BOOSTDIR=directory-path` to the `PARAMETERS` clause of the `CREATE INDEX` or `ALTER INDEX` statement, for example:

```
create index moltable_ix on moltable (ctab)  
    indextype is C$DIRECT2021.mxixmdl  
    parameters ('boostdir=/opt/BIOVIA/boostfiles');
```

This directory name allows you to use `MDLAUX.CREATEBOOSTFILE` without specifying a `BOOSTDIR` parameter. This directory name overrides any name specified with `MDLAUX.SETPROPERTY`.

To specify the location of boost files for all Direct users in the Oracle instance, use the `MDLAUX.SETPROPERTY` procedure to set a global value for the `BOOSTDIR` parameter. For example:

```
execute mdlaux.setproperty('boostdir', '/opt/BIOVIA/boostfiles');
```

You must be connected to Oracle as the cartridge owner, `C$DIRECT2021`, to use this command. Setting this property allows users to use `MDLAUX.CREATEBOOSTFILE` without having to be concerned with creating a directory to hold the boost files and without having to specify a `BOOSTDIR` parameter in either `CREATE INDEX` or `MDLAUX.CREATEBOOSTFILE`.

Individual boost files are named `username_indexname_filename[NNNNN][.ext]`

where:

- `username` is the name of the user who owns the domain index
- `indexname` is the name of the domain index

- *filename* is a name specific to the type of the boost file (for example, fastsearch)
- NNNNN specifies which row in the table for Fastsearch and inverted keys data
- *.ext* is a file extension used with the connection table boost files to differentiate the data (*.dat*) from the index (*.idx*).

Size of Boost Files

When using Direct boost files, additional disk space is required outside of any Oracle tablespace.

The amount of additional disk space is the same as the space used within Oracle and can be computed with the following SQL statements.

Replace *indexname* in each statement with the name of the molecule or reaction domain index.

Note: If *indexname* is longer than 24 characters, use only the first 24 characters of *indexname* in the SELECT statements below.

Replace *molecule-table* with the name of the table containing molecules, and *ctab-column* with the name of the molecule column (often this column name is CTAB).

Replace *reaction-table* with the name of the table containing reactions, and *rctab-column* with the name of the reaction column (often this column name is RCTAB).

- For molecule or reaction FASTSEARCH boostfiles the number of bytes required is:
SELECT SUM(DBMS_LOB.GETLENGTH(FSINDEX)) FROM *indexname*_FSIX;
- For molecule KEYS boostfiles the number of bytes required is:
SELECT SUM(DBMS_LOB.GETLENGTH(KEYS)) FROM *indexname*_IKY2;
- For reaction KEYS boostfiles the number of bytes required is:
SELECT SUM(DBMS_LOB.GETLENGTH(KEYS)) FROM *indexname*_IRKY;
- For molecule CTAB boost files the number of bytes required is:
SELECT SUM(8+DBMS_LOB.GETLENGTH(*ctab-column*)) FROM *molecule-table*;
- For reaction CTAB boost files the number of bytes required is:
SELECT SUM(8+DBMS_LOB.GETLENGTH(*rctab-column*)) FROM *reaction-table*;

Boost Files Creation Time

During creation, a file is created in the boost file directory and data are copied from the appropriate Oracle table into the file. Timings to create boost files for a 9.8 million row table on a fast Linux server are:

- Inverted keys – 2 minutes to create 76 files totaling 1.1 Gb
- Molecule ctab – 21 minutes to create two files totaling 8.3 Gb
- Fastsearch – 14 minutes to create 3 files totaling 4.8 Gb

For more information on creating boost files, see [MDLAUX.CREATEBOOSTFILE](#).

Recreating Boost Files

Boost files contain a snapshot of data at the time that the boost files are created. When new records are inserted into the table, or when records are updated in a table, the new or updated records data will not be reflected in the boost files. Searches will retrieve data for these records from Oracle until such time as the boost files are recreated.

BIOVIA recommends running the procedures `MDLAUX.UPDATEPENDINGINVERSIONS` and `MDLAUX.UPDATEPENDINGFASTSEARCH` after every 2000 inserts or updates in a molecule or reaction table. If boost files have been created for inverted keys or Fastsearch data, you must recreate the KEYS boost file after running `MDLAUX.UPDATEPENDINGINVERSIONS`, and you must recreate the FASTSEARCH boost file after running `MDLAUX.UPDATEPENDINGFASTSEARCH`.

Note: If you do not recreate the boost files after an update operation, the boost files will contain invalid data and will not be used for searching.

Tip: Recreating the FASTSEARCH boost file is a lengthy operation on a large database, you might want to do this less often than every 2000 inserts or updates.

In addition, if boost file files have been created for molecule or reaction connection table data, you can recreate the CTAB boost file to add the updated molecules and reactions to it. This operation is lengthy on a large database, and the performance gain is not so great that it needs to be run every 2000 updates or inserts. You might want to wait until a much larger batch of updates or inserts has been processed and then recreate the CTAB boost file.

Some index maintenance operations disable one or all of the boost files in the index. This is done because the maintenance operation invalidated the data in the boost file.

The KEYS boost file is disabled and must be recreated after any of the following operations:

- `MDLAUX.UPDATEPENDINGINVERSIONS`
- `MDLAUX.RECREATEKEYS`
- `ALTER INDEX REBUILD PARAMETERS ('KEYS')`
- `ALTER INDEX REBUILD PARAMETERS ('ALL')`
- `ALTER INDEX REBUILD` (without any PARAMETERS clause)

The FASTSEARCH boost file is disabled and must be recreated after any of the following operations:

- `MDLAUX.UPDATEPENDINGFASTSEARCH`
- `MDLAUX.RECREATEFASTSEARCH`
- `ALTER INDEX REBUILD PARAMETERS ('FASTSEARCH')`
- `ALTER INDEX REBUILD PARAMETERS ('ALL')`
- `ALTER INDEX REBUILD` (without any PARAMETERS clause)

The CTAB boost file is disabled and must be recreated after any of the following operations:

- `ALTER INDEX REBUILD PARAMETERS ('ALL')`
- `ALTER INDEX REBUILD` (without any PARAMETERS clause)

There might be additional operations that disable use of boost files. Use `MDLAUX.SHOWBOOSTFILES` to determine what boost files are currently in use.

See also

[Displaying Boost Files](#)

Displaying Boost Files

The current boost file status for a domain index is obtained with the function `MDLAUX.SHOWBOOSTFILES`, for example:

```
SELECT MDLAUX.SHOWBOOSTFILES('ACD2D_IX') FROM DUAL;
```

For more information on displaying boost files, see [MDLAUX.SHOWBOOSTFILES](#).

Removing Boost Files

Boost files can be removed from a domain index at any time. To remove a boost file from an index, execute the PL/SQL Direct procedure `MDLAUX.REMOVEBOOSTFILE`. For more information, see [MDLAUX.REMOVEBOOSTFILE](#).

Appendix H:

Upgrading Direct 8

The Direct upgrade tool, `directupgrade.exe` on Microsoft Windows or `directupgrade` on Linux, upgrades a table containing Direct 8 binary molecules or reactions so that the structures are compatible with Direct 2021. After the structure conversion, `directupgrade` creates a Direct 2021 molecule or reaction domain index on the table. For information about the chemistry changes made to structures during the upgrade see [Direct 8 to Direct 2021 Chemistry Update](#).

If you are upgrading a Direct 9 or later molecule or reaction table refer to the instructions in [Upgrading Indexes](#).

Use the `directupgrade` utility to convert the Direct periodic table (PTABLE) file format and salt definition (SALTS) file format to the new formats used by Direct 9 or later and Pipeline Pilot 9.5 or later.

Note: The new periodic table format contains additional information used by Pipeline Pilot, and the new salt definition file format uses an SDF file rather than MDL Line Notation strings.

Configuring Direct Global Cartridge Environment

Configure your Direct 2021 cartridge installation so that it contains the environment files used in your Direct 8 installation. For more information, see [Setting the Cartridge C\\$DIRECT2021 Environment](#).

If you have customized the periodic table (PTABLE), salt definitions file (SALTS), or Sgroup field definitions file (SGROUPFIELDS) in your Direct 8 cartridge installation, you need to install those definition files into your new Direct 2021 cartridge installation.

The SGROUPFIELDS definition file format has not changed in Direct 8, and this file can be fetched from the Direct 8 installation with (C\$MDLICHEM80)MDLAUX.GETENVFILE and then installed into the Direct 9 or later installation with (C\$DIRECT2021)MDLAUX.SETENVFILE.

The PTABLE and SALTS definition file formats have changed in Direct 2021 to match the format used in Pipeline Pilot Server. You must:

1. Fetch these files from the Direct 8 installation with (C\$MDLICHEM80)MDLAUX.GETENVFILE.
2. Upgrade the files to the new format using the Direct 2021 upgrade utility. The procedure to upgrade the PTABLE and SALTS definition files is described in [Converting a Previous Direct PTABLE or SALTS File to Direct 2021 Format](#).
3. Install the new upgraded files into the Direct 2021 installation with (C\$DIRECT2021)MDLAUX.SETENVFILE.

In Direct 8, the MOL and RXN operators did not validate pseudoatoms in the molecule or reaction being inserted into the table. Validation was only done by the domain index, using a periodic table of elements (ptable) that was local to that specific domain index.

This meant that it was possible to add invalid structures – that is, those containing pseudoatoms not present in the ptable – to a table that did not have a domain index, and when the index was created it would fail because of the invalid elements.

In more recent versions of Direct validation is performed twice: once by the MOL or RXN operator using the global (cartridge) ptable, and again when the structure is indexed using the ptable local to the domain index. This additional validation prevents users from inserting structures containing invalid pseudoatoms into a table.

When modifying your cartridge ptable so that it is compatible with your customized Direct 8 ptable, ensure that it contains all of the pseudoatoms which are in either of those two ptables.

Upgrading Direct 8 Molecule or Reaction Table

The upgrade procedure is the method whereby tables configured for Direct 8 are updated to tables configured for Direct 2021. This involves some preparatory steps followed by running the `directupgrade` utility. The `directupgrade` utility converts the binary large objects, BLOBs, containing the molecules or reactions into a format compatible with Direct 2021 and Pipeline Pilot Server.

1. Decide how to handle R and X pseudoatoms in your previous Direct table.

If your Direct table that is being upgraded has molecules or reactions which contain R or X atoms, you must decide how to handle those in the Direct 2021 table that will be created.

R and Z pseudoatoms have a special meaning within Pipeline Pilot chemistry: they have been used to provide an alternate method of representing Markush chemical structures. Because of this special meaning, Direct versions 9 through 2017 do not allow R and Z pseudoatoms to be inserted into a table. The `directupgrade` utility automatically converted R and Z to some other pseudoatom, '*Rgp*' and '*Zgp*' by default.

Starting with Direct 2017 R2, both R and Z pseudoatoms can be inserted into a table, and in addition R and Z pseudoatoms act as query atoms in a substructure search query. While an R atom in an SSS query will match an R atom in the table, it will also match any other atom including hydrogen.

Z atoms do not occur in Direct tables, but your table may contain R atoms. If it does, you must decide how to handle it in the converted table:

- Do nothing, in which case `directupgrade` will convert all R pseudoatoms to *Rgp* pseudoatoms. *Rgp* does not have any special meaning, an *Rgp* atom in a substructure query will only match an *Rgp* atom in the database.
- Use the `directupgrade` parameter `RATOMSYMBOL` to convert R pseudoatoms to some other pseudoatom which does not have same special meaning that R does, for example to convert R to '*GRP*' with `RATOMSYMBOL=GRP`. The replacement symbol can be one to three characters and must be present in the periodic table environment file, either in the Direct cartridge environment or in the file specified as an argument to the `PTABLE` parameter of `directupgrade`.
- Use the `directupgrade` parameter `RATOMSYMBOL` to cause the program to leave R pseudoatoms unchanged, you do this with the option `RATOMSYMBOL=R`. If you retrieve a structure containing an R atom from the database and use it as a substructure query, all R atoms in the query will be removed before searching; R in the query matches any atom in the database structure including hydrogen. You can force SSS to ignore this special feature of R query atoms by including a third argument to SSS, '*InterpretRAtomsLiterally*'. The SSS search will then act like it did in Direct 8, where an R atom only matches an R atom. FLEXMATCH searches never treat R atoms as query features, an R in a FLEXMATCH query only matches an R in the database.

Starting with Direct 2017 R2, X pseudoatoms in a substructure query also have a special meaning. An X atom in an SSS query will match an X atom in the database, but it will also match any halogen.

The Direct 2021 `directupgrade` utility does not provide a way to rename X atoms during registration. You can force SSS to ignore this special feature of X query atoms by including a third argument to SSS, '*InterpretXAtomsLiterally*'.

FLEXMATCH searches never treat X atoms as query features, an X in a FLEXMATCH query only matches an X in the database.

2. Obtain the environment files from your previous the Direct table's domain index. There are three environment files: SGROUPFIELDS, PTABLE, and SALTS. You might need to obtain all, some, or none of these files.
3. Prepare a table in Oracle for the upgrade. For example, import the table into a new schema or copy it into a new schema.
4. Drop the existing Direct index.

Note: The upgrade fails if an index from a previous version of Direct is present.

5. Undo the setup for the previous version of Direct, and then set up the Direct 2021 version.
6. Run the directupgrade utility.
7. Check the index.

Upgrading Direct

How you perform the Direct upgrade depends on the amount of customization that occurred in your previous version.

Note: The following procedure requires that you already have the Direct 2021 cartridge installed in Oracle.

1. Obtain the environment files from your previous Direct table's domain index.

There are three environment files: SGROUPFIELDS, PTABLE, and SALTS. If these files have been customized for the table, that is, they are different from your previous Direct cartridge global environment, you need to extract them into files and provide the file names as arguments to the directupgrade program.

Tip: If you are uncertain whether any of the environment files have been customized, see [How-To Determine If Direct Environment Files Are Customized](#).

For example, if SGROUPFIELDS was customized for the table being upgraded:

```
sqlplus <user 6,7,8>/<password>
select mdlaux.getindexenvfile('db_mols', 'sgroupfields',
'/home/users/db_mols_sgroupfields.txt') from dual;
```

2. Prepare a table in Oracle for the upgrade.

Make a copy of the original table in a new schema, this new table will be upgraded for use with Direct 2021. The two most common ways to copy the table are to import it or to use CREATE TABLE AS SELECT to create it.

Note: If you plan to import the table when you export data from the previous Direct schema you should exclude the domain index using the EXCLUDE option with Data Pump Export.

After you have a copy of the table you must:

- Remove a NOT NULL constraint on the molecule or reaction column if one exists

For example:

```
SQL> alter table structures modify (ctab null);
```

- Disable any triggers on the table which call Direct functions

3. Drop the existing Direct index on the table to be upgraded.

This step is only required if the original domain index still exists on the copy of the table, there will not be any domain index if the table was copied using `CREATE TABLE AS SELECT` or if it was imported with the option to exclude the domain index.

- a. Connect to the new schema in Oracle.

- b. Drop the domain index, for example:

```
SQL> drop index structures_molix force;
```

- c. Make sure there are no tables remaining that have names which start with the domain index name followed by underscore, for example:

```
SQL> select table_name from user_tables where table_name like
'Structures_Molix_%';
```

If any tables with names following this pattern exist they must be dropped, for example:

```
SQL> drop table structures_molix_crprm purge;
```

4. Undo the setup of the previous version of Direct, and then set up the Direct 2021 version.

```
SQL> execute mdlaux.unsetup;
```

```
SQL> execute c$direct2021.mdlauxop.setup;
```

Note: Performing these steps is a good idea even if the schema might have already been set up for Direct 2021. Rerunning these steps does no harm. If the setup has not been done, the upgrade will not complete and will stop after performing a substantial amount of work.

5. Run the `directupgrade` utility.

The upgrade utility is a program that resides in the `bin<versionNumber>` subdirectory of your Direct 2021 installation and is named `directupgrade` on Linux servers and `directupgrade.exe` on Windows servers.

The upgrade utility is discussed in detail in [Direct Upgrade Utility Parameters](#).

This chapter documents some common scenarios, but BIOVIA strongly suggests that you consult "*Direct Upgrade Utility Parameters*" before deciding on which parameters to use. Although a parameter file can be used to run the upgrade, all the examples that follow use command-line parameters.

- First, set the path so that runtime libraries required by the upgrade utility can be located.

On Windows, execute `<install_directory>\direct2021\bin<versionNumber>\setpath.bat`

On Linux, source `setpath` (or `setpath.csh`) located in `<install_directory>/direct2021/bin<versionNumber>`

- To upgrade a table in the `testuser` schema with index named `test_mdlix` enter:

```
directupgrade connstr=testuser/testuser table=test_moltable
index=test_mdlix logfile=test.log
```

- To upgrade a table in the `testuser` schema and provide the `SGROUPFIELDS` and `PTABLE` environment file from a previous Direct database for the table `test_moltable` in this database, enter:

```
directupgrade connstr=testuser/testuser table=test_moltable
index=test_mdlix
```

```
sgroupfields=C:\temp\test_sgroups.txt ptable=C:\temp\db_ptable.txt  
logfile=testuser.log
```

Tip: These are files obtained with `getindexenvfile` in Step 1.

Notes:

- The `directupgrade` program can be run on any machine which has a 64-bit Oracle client installation. It does not need to be run on the Oracle database server.
- The upgrade utility might find molecules or reactions that contain errors. If this occurs the program will issue the following error:
MDL-1919: Molecule failed registration check, write the molfile or rxnfile to the log file, insert a NULL in the molecule or reaction table in place of the structure, and continue with the upgrade. You can delete the row or update it with a corrected structure when `directupgrade` finishes processing all rows.
- Use the `ERRORTABLE` option to write information about each failed molecule or reaction to a table as well as the log file. This way you do not have to process the log file to locate failed records, you can immediately find them with a search for `WARNING_OR_ERROR='ERROR'` in the error table. See the discussion of the `ERRORTABLE` parameter at the end of [Direct Upgrade Utility Parameters](#).
- Both the log file and the error table will contain the original molfile or rxnfile for the failed record, you can read this file into BIOVIA Draw and correct the problem. When you have a new molfile or rxnfile update the upgraded table, for example:

```
update molstr set ctab=mol('c:/temp/new_molfile.mol')  
where rowid='AAGTOEAAGAAA01uAAA';  
commit;
```

6. Check the index.

The end of the log file should contain lines that are similar to:

```
Processed 5000001 rows in 10168 seconds  
Creating index MOLSTR_MDLIX on table MOLSTR  
Created index MOLSTR_MDLIX in 13719 seconds  
Running MDLAUX.RECREATEFASTSEARCH on index MOLSTR_MDLIX  
Created FASTSEARCH data for index MOLSTR_MDLIX in 16030 seconds
```

These lines represent:

Repacking and compressing the molecule or reaction BLOBs, creating the domain index, and creating the fastsearch index.

You can also run the following SQL statement. The values of `status` and `opstatus` should both have values of "VALID."

```
select substr(table_name,1,20) table_name,  
       substr(index_name,1,20) index_name,  
       substr(ityp_owner,1,15) ityp_owner,  
       substr(domidx_status,1,10) status,  
       substr(domidx_opstatus,1,10) opstatus from user_indexes  
where domidx_status is not null  
order by table_name,index_name;
```


Note: Molecules and reactions containing unattached X pseudoatoms will compute a different molecular weight and formula in Direct 9 or later than they did in previous Direct versions. Direct 9 or later assigns a valence of one to X, while previous versions of Direct assigned a valence of zero to X. In Direct 9 and later, an unattached X atom has one implicit hydrogen, its formula will be XH and its molecular weight will be about one higher than it was in previous versions of Direct.

7. If your table contains a molecular weight column of values computed using the MOLWT operator, you should recompute the weight after the table has been upgraded. For example:

```
update moltable set molecular_weight = molwt(ctab);
```

The MOLWT operator or MDLAUX.MOLWT function in Direct 9 or later computes the molecular weight of molecules containing isotopes using the measured mass of the isotope. Previous versions of Direct computed the molecular weight using the integral mass number of the isotope. For example, Direct 8 used a mass of 17 for ^{17}O while Direct 9 or later uses a mass of 16.99913150. Thus, molecular weights computed with MOLWT will be different in Direct 9 or later for molecules containing isotopes.

Upgrade Example

- (Optional) Fetch the old PTABLE file if it had been customized:

```
C:\> sqlplus test80/test80
SQL> select mdlaux.getindexenvfile('test_mdlix', 'ptable', 'c:\temp\test_
ptable.txt') from dual;
MDLAUX.GETINDEXENVFILE('TEST_MDlix','PTABLE','C:\TEMP\TEST_PTABLE.TXT')
-----
1
```

- Make a copy of the table to be upgraded, ensure it has a primary key:

```
C:\> sqlplus testuser/testuser
SQL> create table test_moltable as select * from testuser.test_moltable;
Table created.
SQL> alter table test_moltable modify (cdbregno primary key);
Table altered.
```

- Upgrade the copy of the table. If the PTABLE file had not been customized when the original index was created, do not include a PTABLE parameter:

```
C:\> directupgrade connstr=testuser/testuser table=test_moltable
index=test_ix logfile=test.log
```

If the PTABLE file had been customized, provide the original customized PTABLE as a parameter to directupgrade:

```
C:\> directupgrade connstr=testuser/testuser table=test_moltable
index=test_ix ptable=c:\temp\test_ptable.txt logfile=test.log
```

- In either case the program produces output similar to this:

```
Direct 2021 Database Upgrade Utility
Revision 2021 (Microsoft Windows Oracle12) (21.1.0.x.x)
Copyright (c) Dassault Systemes, 1999-2020
```

Converting Direct 8 table "test_moltable" to BIOVIA Direct 2021

Repacking structures in table TEST_MOLTABLE

MDL-5092: warning:Added implicit hydrogens to metal atom

Previous message occurred processing structure at ROWID=AADV7YAAGc6iAAH,

```
primary key value=3338
10000 20000 30000 40000 50000 60000
MDL-2134: Warning: Removed wedge from atom that is not a stereo center
Previous message occurred processing structure at ROWID=AADV7YAAjAAABwoAAE,
primary key value=61177
70000 80000 91000 100000
MDL-5092: Warning: Added implicit hydrogens to metal atom
Previous message occurred processing structure at ROWID=AADV7YAAjAAAsCgAAL,
primary key value=109550
110000
MDL-2134: Warning: Removed wedge from atom that is not a stereo center
Previous message occurred processing structure at ROWID=AADV7YAAkAAJoQwAAT,
primary key value=111285
120000 130000 140000
MDL-2134: Warning: Removed wedge from atom that is not a stereo center
MDL-2012: Warning: Removed invalid atom stereo center(s)
Previous message occurred processing structure at ROWID=AADV7YAAoAAJdOLAAC,
primary key value=145220
150000 160000 170000 180000 191000
Processed 200000 rows in 295 seconds
Creating index TEST_IX on table TEST_MOLTABLE
Created index TEST_IX in 357 seconds
Running MDLAUX.RECREATEFASTSEARCH on index TEST_IX
Created FASTSEARCH data for index TEST_IX in 271 seconds
```

Direct Upgrade Utility Parameters

Parameters can be specified on the command line, in a parameter file, or a combination of the two. Parameters to the upgrade utility consist of a keyword, or a keyword and value pair. The value is separated from the keyword by an equals sign (=). If a value is case dependent, includes blanks, or includes double quote characters it must be enclosed in double quotes with any double quote characters doubled. Keywords are not case dependent and do not contain spaces or quotes.

For example, to run `directupgrade` with command-line parameters:

```
directupgrade connstr=molddb/moldbpw table=molstr index=molstr_mdlix
logfile=molstr_mdlix.log
```

Alternatively, you can create a parameter file. Parameters in the parameter file must each be on a separate line.

- For example, upgrade using a parameter file:

1. Create `upgrade_molstr.txt` with contents:

```
connstr=molddb/moldbpw
table=molstr
index=molstr_mdlix
logfile=molstr_mdlix.log
```

2. Perform the upgrade

```
directupgrade parfile=upgrade_molstr.txt
```

- For example, upgrade using a parameter file with command-line:

1. Enter:

```
upgrade_molstr.txt contains
table=molstr
index=molstr_mdlix
logfile=molstr_mdlix.log
```

2. Perform the upgrade:

```
directupgrade connstr=molddb/moldbpw parfile=upgrade_molstr.txt
```

Required Parameters

Text in brackets is either optional or not always required.

Parameter	Description
CONNSTR=username/password [@tnsname]	Specifies the username, password, and dbalias (optional) to connect to the account containing the table to be upgraded. Specify the argument as either: username/password[@dbalias] or /[@dbalias]
TABLE=tablename	Specifies the name of the table to be upgraded.
INDEX=indexname	Specifies the name of the new Direct 2021 domain index to be created.

Optional Parameters

Parameter	Description
LOGFILE=logfile-name	Specifies the name of the logfile, it contains all output from the program. If the file exists the program output will be appended to the file, otherwise it will be created. If not specified the default name is directupgrade.log.
CTAB=structure-column-name	Only needed if the table contains more than one BLOB column. If that is the case, specify the column name for the column which contains Direct molecules or reactions.
GENERICCS	Forces directupgrade to create a generic (Markush) structure index on the molecule table. Normally this option is not needed, directupgrade automatically determines whether the table contains reactions, non-generic molecules, or generic molecules and creates the correct index type. Use this option only if the previous Direct domain index was a GENERICCS index but the table contains only non-generic structures. The new Direct domain index will be created based on that determination.

Parameter	Description
PTABLE=ptable-filename	Specifies the name of a non-default periodic table file that was used in the previous version to use with the Direct domain index. Use <code>getindexenvfile</code> for this table.
SGROUPFIELDS=sgroupfields-filename	Specifies the name of a non-default data SGROUPFIELDS definition file to use with the Direct domain index.
SALTS=salts-filename	Specifies the name of a non-default FLEXMATCH salt file definition file to use with the Direct domain index.
RATOMSYMBOL=R-atom-symbol	Use this parameter to specify the replacement symbol for the R pseudoatom. To leave R pseudoatoms unchanged specify the value R. Otherwise, specify a replacement symbol of one to three characters that is present in the periodic table environment file, either in the Direct cartridge environment or in the file specified as an argument to the PTABLE parameter of <code>directupgrade</code> . If this parameter is not specified, R is converted to Rgp.
ZATOMSYMBOL=Z-atom-symbol	Use this parameter to specify this replacement symbol, which can be one to three characters and distinct from any other atom symbol. If not specified Z will be converted to Zgp. The replacement symbol must be present in the periodic table environment file, either in the Direct cartridge environment or in the file specified as an argument to the PTABLE parameter. Note: The symbol "Z" was never present in a Direct default periodic table file and is not used in databases sold by BIOVIA.
TABLESPACE=tablespace-name	Option is passed direct to <code>CREATE INDEX</code> to specify the tablespace to use.
TABLE_ TABLESPACE=tablespace-name	Option is passed directly to <code>CREATE INDEX</code> to specify the tablespace to use.
INDEX_ TABLESPACE=tablespace-name	Option is passed direct to <code>CREATE INDEX</code> to specify the tablespace to use.
LOB_TABLESPACE=tablespace-name	Option is passed direct to <code>CREATE INDEX</code> to specify the tablespace to use.
TEMPDIR=directory-name	Option is passed direct to <code>CREATE INDEX</code> to specify the temporary directory to use when creating Fastsearch data.
ERRORTABLE=structure-error-tablename	<i>structure-error-tablename</i> specifies the name of a table which will have error records inserted into it. The table must not exist, if it already exists <i>directupgrade</i> will terminate with error MDL-5100. When this option is provided, if an error occurs upgrading a molecule or reaction then the ROWID, primary key, molecule or reaction, flag 'ERROR', and the error messages for that record will be inserted into the specified table. The molecule or reaction is inserted into the table as a molfile or rxnfile CLOB. If the error occurs while unpacking the binary structure and a molfile or

Parameter	Description
	<p>rxnfile cannot be created the original binary data are written into the table as a BLOB.</p> <p>If a record generates one or more warnings such as "MDL-5092: warning: Added implicit hydrogens to metal atom", an entry is also written to the table specified by the ERRORTABLE parameter. In this case the ROWID, primary key, molecule or reaction, flag 'WARNING', and the warning messages for that record will be inserted into the table. The original molecule or reaction is inserted into the table as a molfile or rxnfile CLOB. This action is in addition to the normal error logging, so the log file will still contain the same information.</p> <p>Using an error table allows you to easily locate, correct and update structures which require manual intervention during the upgrade procedure. If a serious error occurs which causes directupgrade to terminate the automatic rollback to the last commit point will also roll back the error table, thus it only contains entries for those records which have been upgraded. It will not contain an entry for the record which caused the serious error, only the log file will contain that information.</p> <p>The error table also contains entries for records which generate warnings. This may be helpful in locating those records if desired. directupgrade will create the error table. If it already exists directupgrade will terminate with an error. The table is created as follows:</p> <pre>CREATE TABLE structure-error-tablename (STRUCTURE_ROWID VARCHAR2(4000), STRUCTURE_PRIMARYKEY VARCHAR2(4000), STRUCTURE_FILE CLOB, STRUCTURE_DATA BLOB, WARNING_OR_ERROR VARCHAR2(10), MESSAGES VARCHAR2(4000));</pre> <p>Records which cannot be automatically upgraded will contain a NULL for the structure and the error table will contain a row with the following values:</p> <p>STRUCTURE_ROWID contains the ROWID of the record.</p> <p>STRUCTURE_PRIMARYKEY contains the primary key value of the record.</p> <p>STRUCTURE_FILE contains the molfile or rxnfile of the record, or if a molfile or rxnfile cannot be created then STRUCTURE_DATA contains the original binary data.</p> <p>WARNING_OR_ERROR contains the string 'ERROR'.</p> <p>MESSAGES contains the errors indicating why the record cannot be upgraded.</p> <p>Records which generate warnings will also be stored in the error table:</p> <p>STRUCTURE_ROWID contains the ROWID of the record.</p> <p>STRUCTURE_PRIMARYKEY contains the primary key value of the</p>

Parameter	Description
	<p>record.</p> <p>STRUCTURE_FILE contains the original molfile or rxnfile of the record.</p> <p>WARNING_OR_ERROR contains the string 'WARNING'.</p> <p>MESSAGES contains informational messages about the record, for example "MDL-5092: warning: Added implicit hydrogens to metal atom".</p>

How To Determine If Direct Environment Files Are Customized

Customization may be specific to certain tables, across the global cartridge, or both. It is important that the Direct database administrator know what customizations, if any, have been done so that these customizations can be carried forward to the Direct 2021 installation. This section describes how to determine the customization in Direct installations.

- Determine if any domain indexes on molecule or reaction tables have customization that differs from that used across the cartridge.

For each table with a domain index for which the customization is not known, extract the environment files that you want to check, for example, PTABLE, SALTS, and SGROUPFIELDS. Extract these files from the Direct cartridge and compare the index and cartridge files.

For files that are the same, no action specific for the table needs to be taken, and you can proceed to checking whether the environment for the cartridge has been customized. If the files are different, then be sure to use the index-specific file when the table is upgraded. The sample workflow that follows is with a table named *structures*, using the PTABLE as an example and assuming Direct 8 is installed.

```
$ sqlplus chemdata/password
--
-- Fetch cartridge PTABLE environment. Be sure to specify the full path.
SQL> select mdlaux.getenvfile('ptable', '/var/tmp/ptable_80_
cartridge.txt') SQL> from dual;
--
-- Fetch PTABLE environment for STRUCTURES table
SQL> select mdlaux.getindexenvfile('structures', 'ptable',
'/var/tmp/ptable_80_structures.txt')
SQL> from dual;
--
SQL> quit
```

Now compare the two files using a file comparison utility (for example, `diff` on Linux or `fc` on Windows):

```
$ diff /var/tmp/ptable_80_cartridge.txt /var/tmp/ptable_80_structures.txt
```

If there are differences other than comments or whitespace, the environment in the domain index has been customized. To use this customization in the Direct 2021 cartridge for this database, use the PTABLE parameter when upgrading the table and its domain index. For more information on parameters, see [Direct Upgrade Utility Parameters](#).

Note: The Sgroupfields environment file is empty if it has not been customized.

- Determine if there has been any customization of the Direct cartridge environment.

This step requires extracting and examining the environment files from the current Direct installation as described below, again using Direct 8 as an example.

```
$ sqlplus 'c$mdlchem80/password'
--
-- Fetch cartridge Ptable environment file. Be sure to use the full
path.
SQL> select mdlaux.getenvfile('ptable', '/var/d80env/ptable_80_
cartridge.txt')
SQL> from dual;
```

```
--
-- Fetch cartridge Salts environment file. Be sure to use the full path.
SQL> select mdlaux.getenvfile('salts', '/var/d80env/salts_80_
cartridge.txt')
SQL> from dual;
--
-- Fetch cartridge Sgroupfields environment file. Be sure to use the
full path.
SQL> select mdlaux.getenvfile('sgroupfields', '/var/d80env/Sgroupfields_
80_cartridge.txt')
SQL> from dual;
--
SQL> quit
```

Examine the files on a case-by-case basis.

PTABLE

Compare the contents of the file with the portion of the default file that is listed in *BIOVIA Chemical Representation > Customizing the BIOVIA Ptable*. If the file contains pseudoatoms not listed in the default, or is missing some of the pseudoatoms listed in the default, the file has been customized. This customized file should be converted and loaded into the Direct cartridge environment so that Direct pseudoatoms will be available.

Note: Converting this file also means that the atomic weights in the customized version will be used unless the converted Ptable file is edited to remove entries that override the default Direct atomic weights.

If the file contains only those pseudoatoms listed in the *BIOVIA Chemical Representation Guide*, your PTABLE is compatible with Direct 2021, except that a few of your atoms will have slightly different atomic weights. If this is acceptable, you can consider that your PTABLE has not been customized. If you require your original atomic weights, assume that the file has been customized.

SALTS

Compare the contents of the file with the default that is listed in "*Customizing the BIOVIA Salts Definition*" in the *BIOVIA Chemical Representation Guide*. If they are the same, excluding comments or white space, the cartridge environment is the default. Otherwise, the file has been customized.

SGROUPFIELDS

The SGROUPS file is empty by default. A table that is not empty has been customized.

Customized files must be installed after the Direct 2021 cartridge is installed. For example,

```
DOS> sqlplus c$direct2021/password
SQL> select mdlaux.setenvfile('ptable', 'C:\d80env\ptable_80_cartridge.txt')
SQL> from dual;
SQL> quit
```


Converting a Previous Direct PTABLE or SALTS File to Direct 2021 Format

- Direct 2021 and Pipeline Pilot share a common format for customizations to the built-in Pipeline PilotChemistry periodic table (PTABLE) and the salt definition file (SALTS). The PTABLE and SALTS files are different in Direct 2021 than they were in Direct versions prior to 9.0. Therefore, if a customized PTABLE or SALTS file from a Direct 8 cartridge is to be installed into the Direct 2021 cartridge, the upgrade utility must first be employed to convert the previous format into the Direct 2021 format. First, set the path so that runtime libraries required by the upgrade utility can be located.

- On Windows, execute `<install_folder>\direct2021\bin<versionNumber>\setpath.bat`
- On Linux, source `setpath` (or `setpath.csh`) located in `<install_directory>/direct2021/bin<versionNumber>`

Then convert the file. For example:

To convert a PTABLE environment file (Linux)

```
directupgrade ptable=/var/d80env/ptable_80_cartridge.txt
outputfile=/var/d2021env/ptable_2021_cartridge.txt
```

To convert a SALTS file (Windows)

```
directupgrade ptable=C:\d80env\salts_80_cartridge.txt
outputfile=C:\d2021env\salts_2021_cartridge.txt
```

- The R pseudo-atom was not allowed in Direct 9 through 2017 and has different searching semantics in Direct 2017 R2 through 2021 from what they were in Direct 8. By default the Direct 2021 upgrade utility automatically converts R pseudo-atoms to the symbol Rgp so that the searching semantics are the same.

If you want to keep the symbol R you only need to specify `RATOMSYMBOL=R` when upgrading the table, no changes to the environment are necessary.

If you want to use a symbol name other than R or Rgp as the replacement for the R pseudo-atom you must configure the Direct 2021 environment with your symbol name:

- Fetch the default Direct 2021 PTABLE file with `(C$DIRECT2021) MDLAUX.GETENVFILE`.
- Edit the file and add the new pseudo-atom symbol. Use the existing line in the file that starts with Rgp as a template to add the new line at the end of the file. For example to change the symbol to Grp add this line:

```
Grp R 1.80 0.79 1 2 3 4 5 0.000 000.0000000
```

- Install the modified file into the Direct 2021 installation `(C$DIRECT2021)` with `MDLAUX.SETENVFILE`.

- If you are installing your previous customized PTABLE file into the Direct 2021 installation: When converting your previous PTABLE file using the procedure in Converting a Previous Direct PTABLE or SALTS file to Direct Format, specify the desired pseudo-atom symbol name with the `RATOMSYMBOL` parameter, for example:

```
directupgrade ptable=old_ptable.txt ratomsymbol=Grp outputfile=new_
ptable.txt
```

Tip: Be sure to specify your new pseudo-atom symbol name with the `RATOMSYMBOL` parameter when upgrading molecule and reaction tables. If you do not do this, the default symbol Rgp will be used.

Note: The Direct 2021 PTABLE file only lists non-default atom symbols or atoms with non-default atomic weights. Direct 8 and earlier versions of the PTABLE file explicitly listed every atom symbol including standard elements. Therefore, your Direct 2021 converted PTABLE file will likely have many fewer entries than in previous revisions.

Common Warnings Issued by the directupgrade Utility

You might see the following warnings when structures are being upgraded. If you want to examine the structures, the primary key value and ROWID will be included with each warning. No changes to the structures should be needed.

```
MDL-2010: warning: Removed invalid Chiral flag
MDL-2011: warning: Removed invalid non-tetrahedral stereo center
(s)
MDL-2012: warning: Removed invalid atom stereo center(s)
MDL-2141: warning: Removed invalid double-either bond(s)
```

These four warnings occur when the stereochemistry perception determines that an atom marked as a stereocenter is not actually a valid stereocenter. This might occur because a trivalent nitrogen stereocenter was removed, which caused a remaining stereocenter to be invalid because of symmetry. directupgrade removes the invalid stereochemical information from the structure.

Tip: To preserve the original molecule or reaction, before you upgrade your table add a column to the table which stores the original the original molfile, rxnfile, or Chimestring in it.

```
MDL-2129: warning: Flattened 3D molecule to 2D
```

This occurs when you upgrade a Direct 8 table containing 3D models. The models are flattened by setting the Z coordinate to zero and adding up or down (wedge) marks to stereocenters.

Note: If you want to preserve the original 3D coordinates, before you upgrade your table add a column to the table which stores the original molfile. (Force the use of the previous version of the MOLFILE operator by prefixing it with the Direct schema name.) For example, to add molfiles to a Direct 8 model table:

```
ALTER tablename ADD (original_molfile CLOB);
UPDATE tablename SET original_molfile = C$MDLICHEM80.MOLFILE
(ctab);
COMMIT;
```

```
MDL-2134: warning: Removed wedge from atom that is not a stereo
center
```

This occurs whenever your table contains a structure with an atom which was considered to be a tetrahedral stereocenter in Direct 8 but is not considered to be a stereocenter in Direct 2021. Trivalent nitrogen and phosphorus are the most common atoms where this will occur. directupgrade removes the up or down (wedge) bond from the atom.

Note: To preserve the original molecule or reaction, before you upgrade your table add a column to the table which stores the original the original molfile, rxnfile, or Chimestring in it.

```
MDL-5092: warning: Added implicit hydrogens to metal atom
```

This occurs whenever your table contains a structure with a metal atom which had a certain number of implicit hydrogens in Direct 8 but has no implicit hydrogens in Direct 2021 directupgrade adds implicit hydrogens (a valence designation such as (I)) to the atom to force the same number of implicit hydrogens that it had in the previous version of Direct.

Appendix I:

Direct 8 to Direct 2021 Chemistry Update

Direct 2021 can refer to either Direct 2017, 2017R2, 2018, 2019, 2020, or 2021. For purposes of the topics covered here, they are equivalent unless explicitly mentioned otherwise.

Audience

This document is intended for administrators and curators of Direct installations who have a thorough understanding of the chemical representation capabilities of Direct 8 and prior versions.

The focus of the Direct 2021 release is to synchronize the chemistry representation and handling between the Direct cartridge and Pipeline Pilot. Without synchronization, there would be unacceptable deficiencies and discrepancies between the search results returned from a cartridge database and those obtained from protocols run against Pipeline Pilot Servers. Due to these core chemistry synchronization changes, the number and types of hits for searches running in Direct 8 cannot be compared directly to the hits for the same searches run in Direct 2021. This document seeks to identify chemistry areas that will cause expected search differences in the two cartridge versions to allow administrators and curators to assess the validity of their updated and synchronized search results from a Direct 2021 installation or migration.

List of Changes

The following areas have changed:

- BIOVIA Ptable definition
- Salts definition
- Valence Differences
- Aromaticity
- Tautomers

See also

- "flexmatch" in the Direct Reference Guide.
- [Tautomers](#)
- [Stereochemistry of Allenes and Biaryls](#)
- [Tetrahedral Stereochemistry](#)
- [Geometric Stereoisomers](#)
- [A-Line and Nonstandard Type Behavior](#)
- [Molecular Weight Differences](#)

For details on changes to the chemistry perception for Pipeline Pilot Server, refer to the Appendix A of the *Chemistry Collection Basic Chemistry Guide*.

Valence Differences

The following elements have different implied valence rules than they had previously. Implied valences are relevant only for atoms that do not have an explicit abnormal valence property applied to them. Refer to "Default Valences" in the *BIOVIA Chemical Representation Guide*.

1 H	The Periodic Table of the Elements																2 He
3 Li	4 Be											5 B	6 C	7 N	8 O	9 F	10 Ne
11 Na	12 Mg											13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
19 K	20 Ca	21 Sc	22 Ti	23 V	24 Cr	25 Mn	26 Fe	27 Co	28 Ni	29 Cu	30 Zn	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
37 Rb	38 Sr	39 Y	40 Zr	41 Nb	42 Mo	43 Tc	44 Ru	45 Rh	46 Pd	47 Ag	48 Cd	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe
55 Cs	56 Ba	*	72 Hf	73 Ta	74 W	75 Re	76 Os	77 Ir	78 Pt	79 Au	80 Hg	81 Tl	82 Pb	83 Bi	84 Po	85 At	86 Rn
87 Fr	88 Ra	**	104 Rf	105 Db	106 Sg	107 Bh	108 Hs	109 Mt									
			57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb	71 Lu
			89 Ac	90 Th	91 Pa	92 U	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No	103 Lr

Changes to implied valences can impact molecular weight and formulas for Direct 8 versus Direct 2021 structures containing these elements.

Additionally, for Direct 2021, implicit hydrogens are never added to metal atoms or ions (implied valence is zero), with the exception of Al(-1) which has a default valence of 4.

The metal ions for Direct 2021 are:

1 H	The Periodic Table of the Elements																2 He
3 Li	4 Be											5 B	6 C	7 N	8 O	9 F	10 Ne
11 Na	12 Mg											13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
19 K	20 Ca	21 Sc	22 Ti	23 V	24 Cr	25 Mn	26 Fe	27 Co	28 Ni	29 Cu	30 Zn	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
37 Rb	38 Sr	39 Y	40 Zr	41 Nb	42 Mo	43 Tc	44 Ru	45 Rh	46 Pd	47 Ag	48 Cd	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe
55 Cs	56 Ba	*	72 Hf	73 Ta	74 W	75 Re	76 Os	77 Ir	78 Pt	79 Au	80 Hg	81 Tl	82 Pb	83 Bi	84 Po	85 At	86 Rn
87 Fr	88 Ra	**	104 Rf	105 Db	106 Sg	107 Bh	108 Hs	109 Mt									
			57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb	71 Lu
			89 Ac	90 Th	91 Pa	92 U	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No	103 Lr

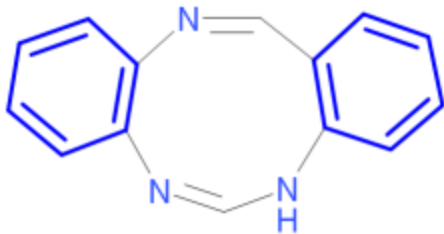
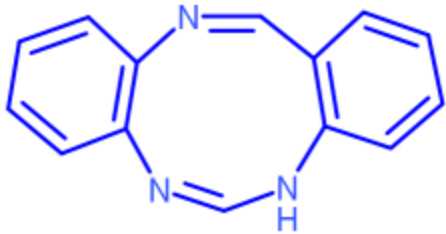
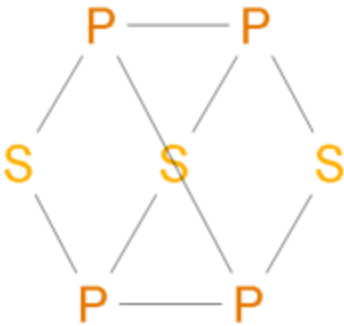
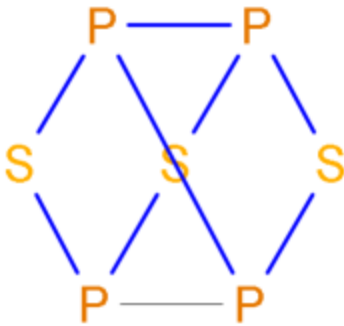
Aromaticity

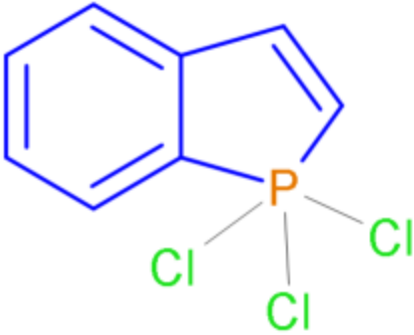
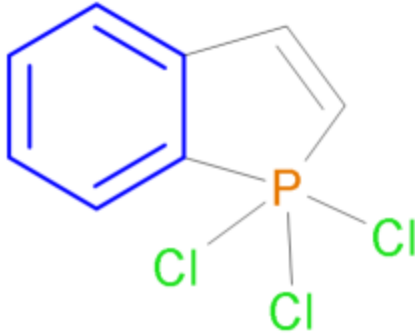
The definition of aromaticity has been significantly changed from the Direct 8 definition, which considered ring sizes (6-membered and 5,7-azulene envelope rings) and the alternate bonding pattern in the ring. Aromaticity perception now uses a more rigorous $4n+2$ π -electron validation of the alternating bonding pattern in rings of all sizes. Where the Direct 8 definition never considered the atom types in

Kekule aromatic rings, in Direct 2021 the atom types are considered to determine if there are π -electrons to contribute to aromatic delocalization. Query atom types now affect whether a ring is considered to be a perceived aromatic ring.

Direct 8 considered extended aromatics (5-ring hetero or C(-1) rings) as having a dual aromatic and single/double character. The Direct 2021 aromatic perception treats these rings as aromatic bonds, and so they are no longer hit by queries unless they are aromatic, S/A or D/A or 'any' query bond types, or the query contains the full ring to allow determination of the ring aromaticity. In Direct 8, the extended aromatic perception results did not propagate to affect the aromatic status of neighboring rings. In Direct 2021, all rings identified as aromatic can propagate the results to neighboring rings.

Examples of aromatic bond differences are shown below. The blue highlighted bonds indicate the respective perceived aromatic rings.

Direct 8	Direct 2021
	
	

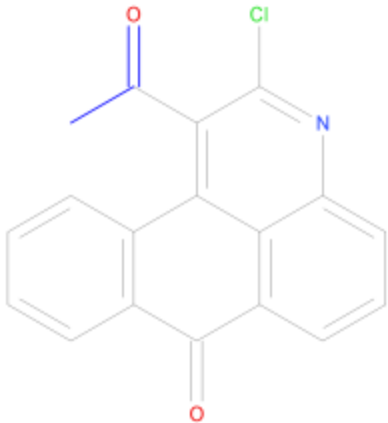
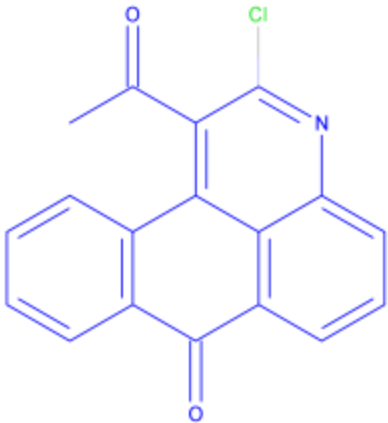

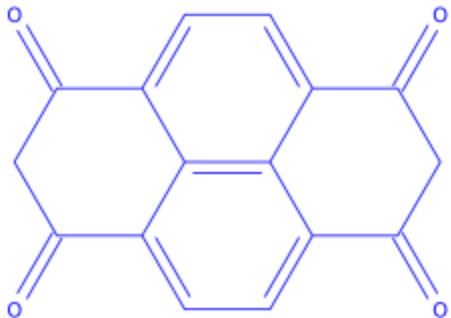
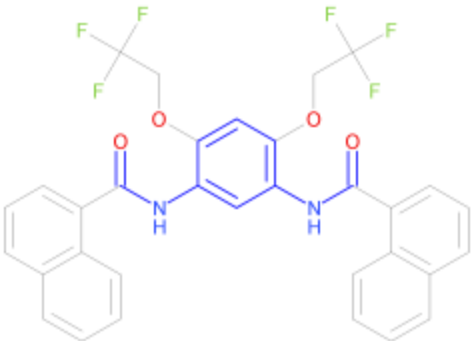
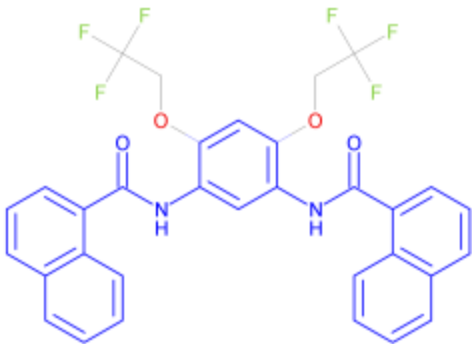
Direct 8	Direct 2021
	

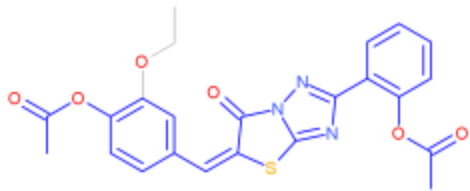
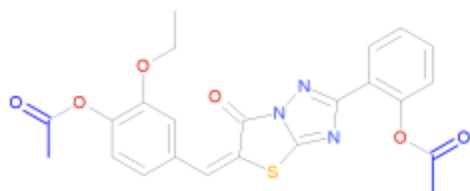
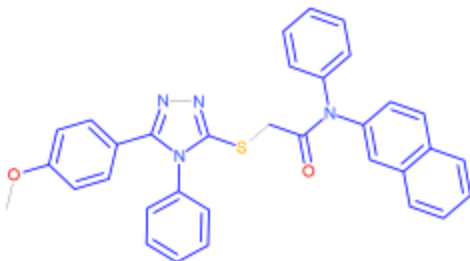
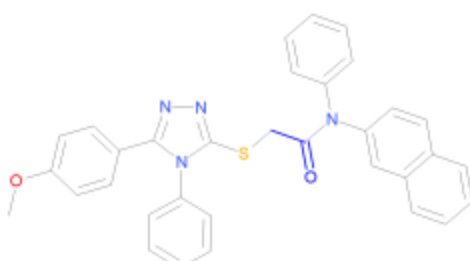
Tautomers

For Flexmatch/TAU searches, the Direct 8 perception of the tautomer region identified the areas of the query and target used to validate differences in the hydrogen count between the structures. This algorithmic perception of the tautomer region is now based strictly on the Pipeline Pilot Client definition of tautomer regions as documented in "Canonicalization and Enumeration of Tautomers", Sayle and Delany, EuroMUG99, 28-29 October 1999, Cambridge, UK.

A key difference between the respective tautomer region perception algorithms is largely based on whether aromatic ring assemblies are included in the identified region or not based on the proximity and location of nearby tautomer groups. These tautomer region differences should have only small effects on tautomer search results for normal chemical structures.

Examples of tautomer differences are shown below. The blue highlighted bonds indicate the respective perceived tautomer regions.

Direct 8	Direct 2021
	
	
	

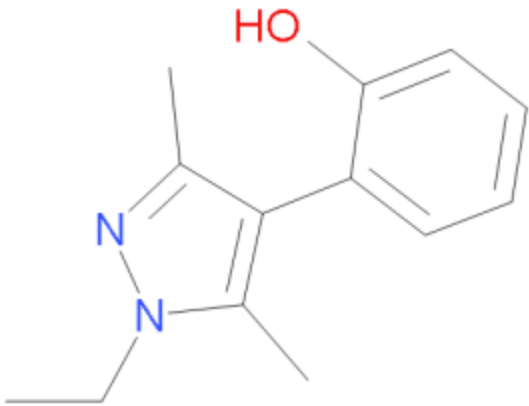
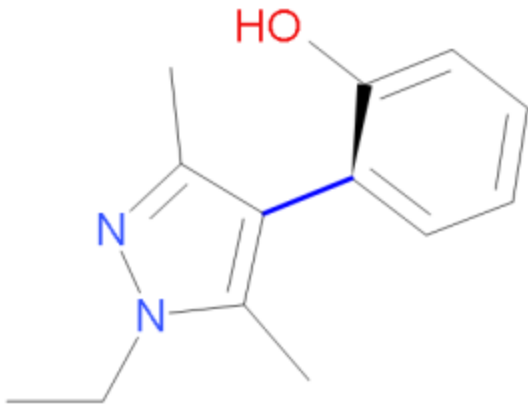
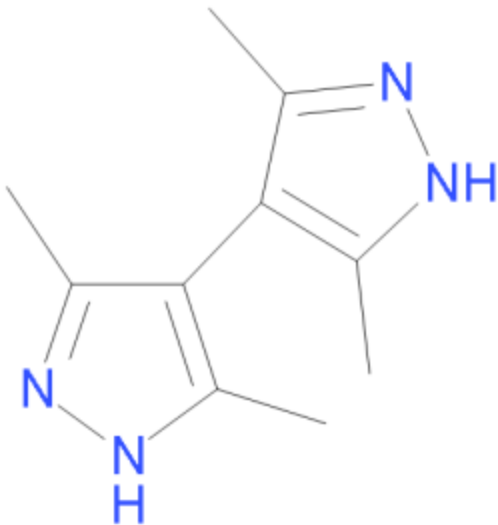
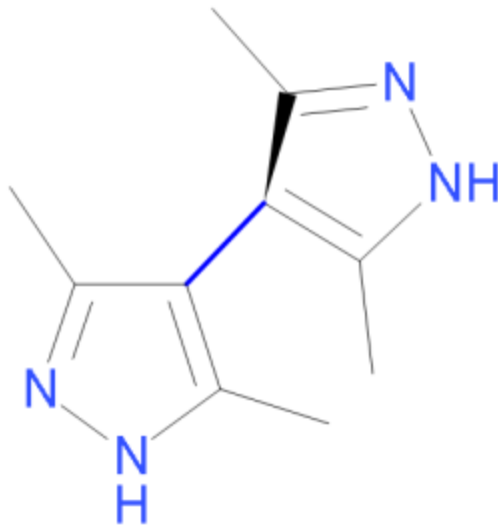
Direct 8	Direct 2021
	
	

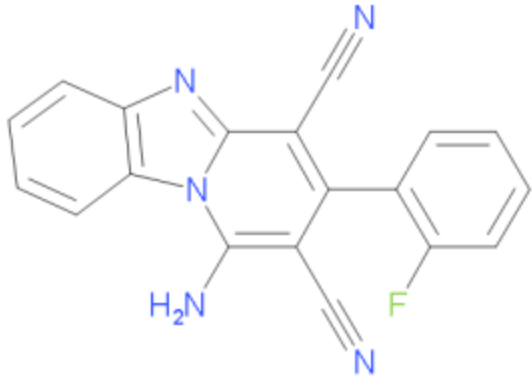
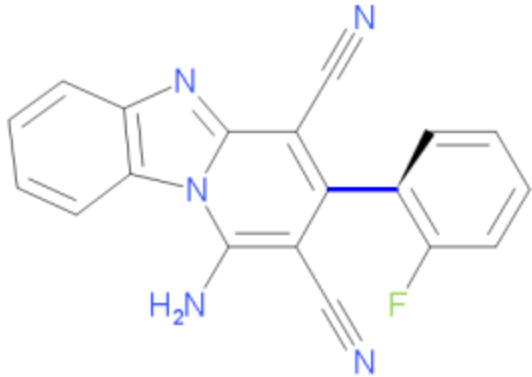
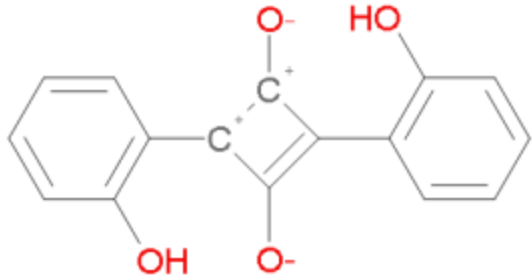
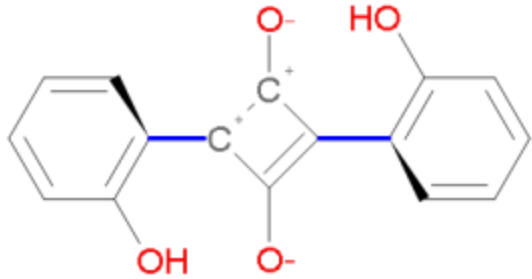
Stereochemistry of Allenes and Biaryls

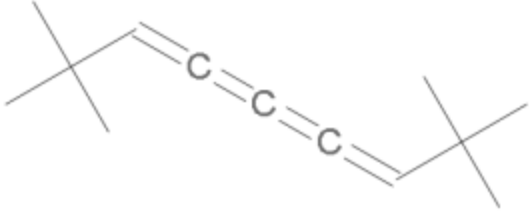
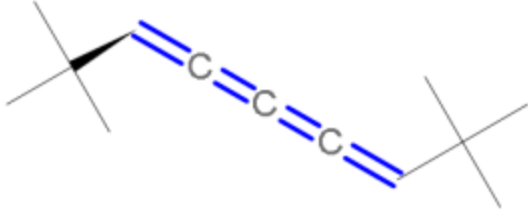
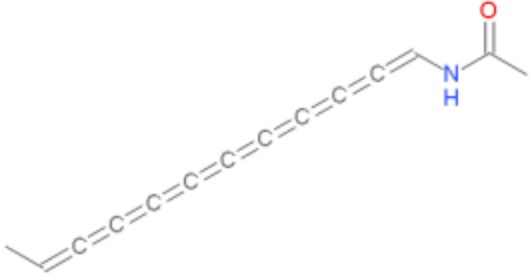
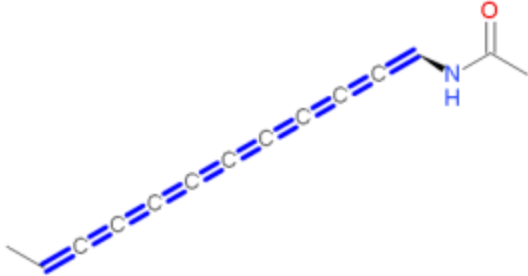
The Direct 8 definition of allenes that could exhibit a parity was limited to $[C,Si]=[C,Si]=[C,Si]$ chemical units. The new allene perception routine supports any even cumulene lengths of $[C,Si]$ atoms as potential nontetrahedral stereochemistry units.

The Direct 8 definition of restricted rotation biaryls that could define an axis of chirality has been enlarged to include bonds between any two aromatic rings using the Direct 2021 aromaticity definition.

Examples of allenes and biaryls are show below. The blue highlighted bonds indicate the new stereogenic units perceived in Direct 2021 that can now be marked to indicate a specific configuration.

Direct 8	Direct 2021
	
	

Direct 8	Direct 2021
	
	

Direct 8	Direct 2021
	
	

Tetrahedral Stereochemistry

Trivalent nitrogen or isoelectronic equivalents are only considered to be valid tetrahedral stereocenters if:

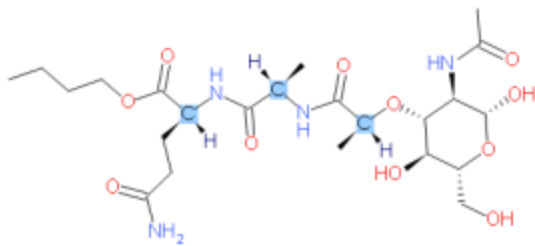
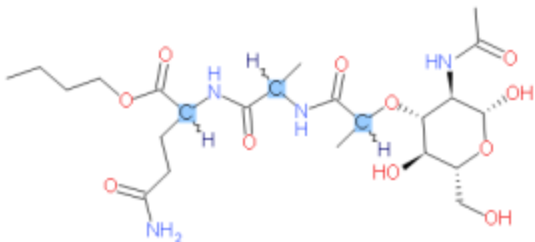
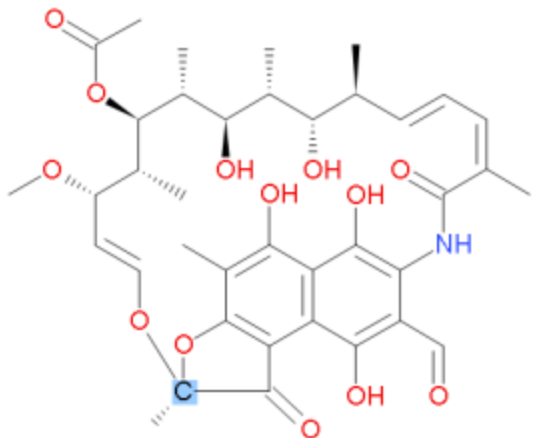
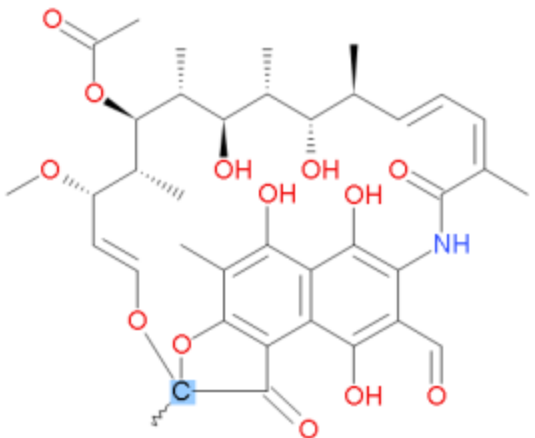
- All three bonds are single and in rings
- None of the three atoms adjacent to nitrogen is part of a double bond

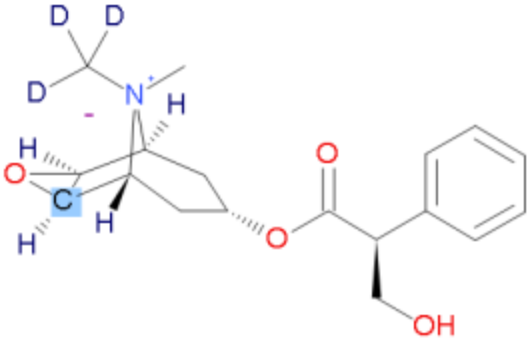
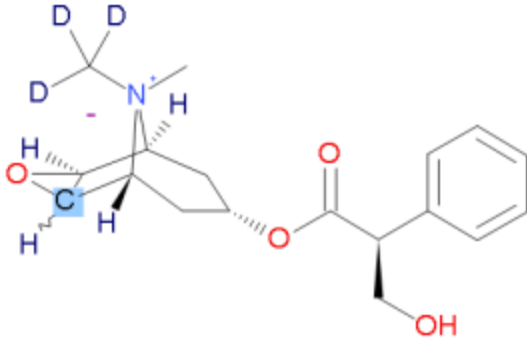
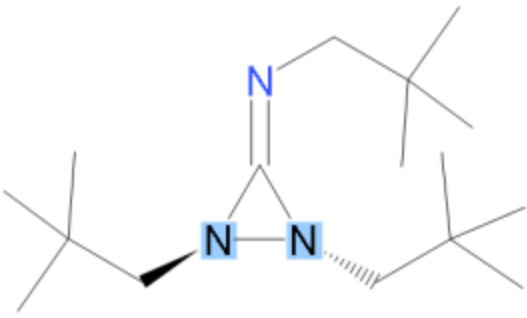
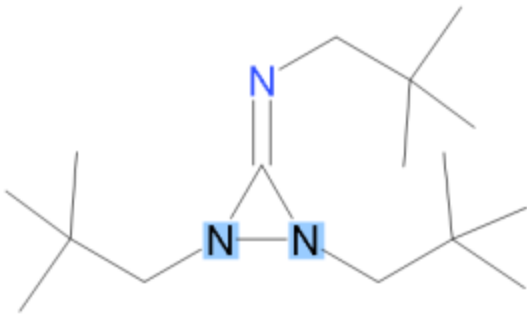
Nitrogen or isoelectronic equivalent atoms that do not meet these criteria will have marked bonds removed on registration and ignored for stereo-specific search queries.

For marked tetrahedral stereocenters that violate the previously documented rules about the type and number of bond marks (see the *BIOVIA Chemical Representation Guide*), the stereoconfigurations of these atoms might be interpreted differently than they were previously. In general, unambiguously marked stereocenters will maintain the same stereoconfiguration before and after the structure migration.

One significant difference is that stereocenters which are so badly marked as to generate an ambiguous parity determination are now internally considered as *unknown stereo*. An unknown stereocenter acts equivalently to an unmarked stereocenter for registration or for queries, unmarked or unknown stereo will hit unmarked, unknown or marked parities. Previously, these ambiguously marked centers would have been considered to have an arbitrary, and typically highly coordinate sensitive, parity.

Examples of tetrahedral stereochemistry differences are shown below. The blue highlighted atoms indicate atoms where the perceived stereochemical parity has changed.

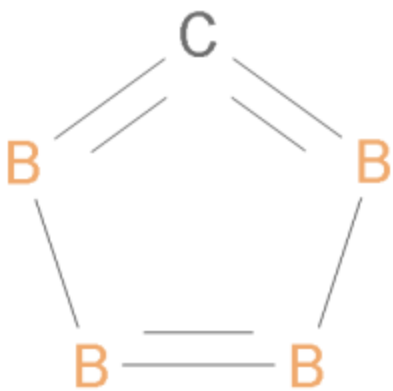
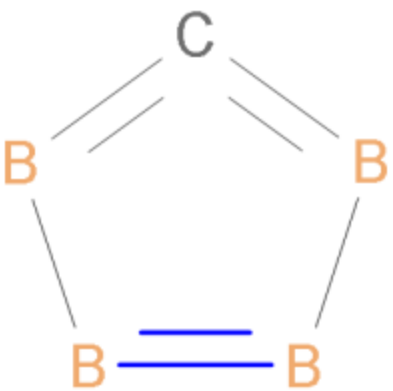
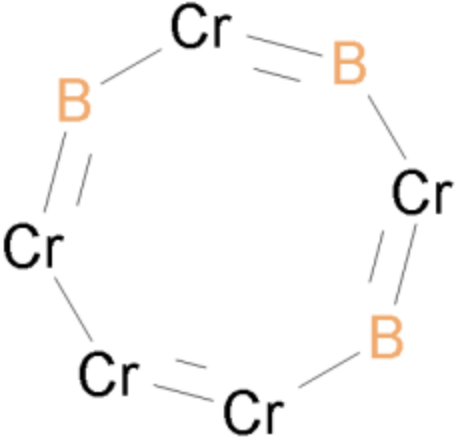
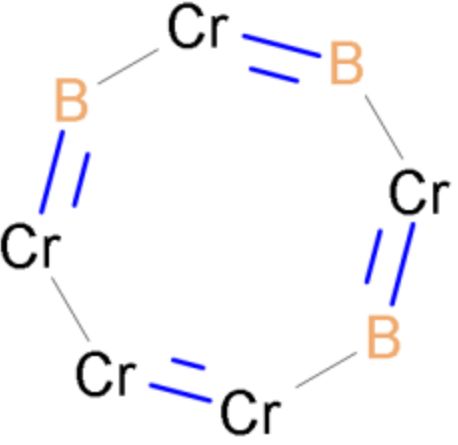
Direct 8	Direct 2021
	
	

Direct 8	Direct 2021
	
	

Geometric Stereoisomers

Previously only C, N, O, Si, P, S atom types were allowed as nonaromatic double bond endpoints to define a cis/trans double bond geometry. For Direct 2021, no atom type restrictions apply, so a larger number of cis/trans bonds can define a geometric parity provided each endpoint atom has two or three attached neighbors. The Direct 2021 geometric perception algorithm also explicitly disallows cis/trans double bonds in rings less than 8 members, while Direct 8 did not consider the effects of small rings for cis/trans double bonds. As a result, cis/trans double bond queries can return more hits in Direct 2021, especially if the geometric bond is in a small ring system, and thus ignored for parity validations during matching.

Examples of geometric stereoisomer differences are shown below. The blue highlighted bonds indicate a bond that can be cis or trans in Direct 2021 but not in Direct 8.

Direct 8	Direct 2021
	
	

Direct 8	Direct 2021

A-Line / Nonstandard Type Behavior

Direct 8 and prior versions ignored all legacy ISIS/Desktop features on import. Direct 2021 adopts the Pipeline Pilot default behavior which converts Atom Alias (A-line) information into atoms of the specified alias type. If the alias symbol is not present in either the default or user-customized periodic

table, those structures cannot be registered in Direct 2021. Such structures should be corrected to remove the atom alias information or to convert it into Sgroup abbreviations.

Molecular Weight Differences

The MOLWT operator or MDLAUX.MOLWT function in Direct computes the molecular weight of molecules containing isotopes using the measured mass of the isotope. Previous versions of Direct computed the molecular weight using the integral mass number of the isotope. For example, Direct 8 would use a mass of 17 for ^{17}O and Direct 2021 uses a mass of 16.99913150. Thus molecular weights computed with MOLWT will be different in Direct 2021 for molecules containing isotopes.

The table of isotope mass values in Direct is the same as that used in Pipeline Pilot. These mass values are slightly different than those used with the MONOISOTOPICMASS operator or MDLAUX.MONOISOTOPICMASS function in previous versions of Direct. Thus, MONOISOTOPICMASS will return a slightly different value in Direct 2021 than it did in previous versions of Direct. On average the difference is very small, +/- 0.00002 for 640 isotopically substituted molecules obtained from a 1M structure chemicals database.