

DataFlux dfPower Studio

This page is intentionally blank



DataFlux dfPower Studio

Getting Started Guide

Version 8.2 Service Pack 1

February 4, 2010

This page is intentionally blank

Contact DataFlux

Corporate Headquarters

DataFlux Corporation
940 NW Cary Parkway, Suit 201
Cary, NC 27513-2792
Toll Free Phone: 877-846-FLUX (3589)
Toll Free Fax: 877-769-FLUX (3589)
Local Phone: 919-447-3000
Local Fax: 919-447-3100
Web: <http://www.dataflux.com>

European Headquarters

DataFlux UK Limited
59-60 Thames Street
WINDSOR
Berkshire
SL4 ITX
United Kingdom
UK (EMEA): +44(0) 1753 272 020

Technical Support

Phone: 919-531-9000
Email: techsupport@dataflux.com
Web: <http://www.dataflux.com/Resources/DataFlux-Resources/Customer-Care-Portal/Technical-Support.aspx>

Legal Information

Copyright © 1997 - 2009 DataFlux Corporation LLC, Cary, NC, USA. All Rights Reserved.

DataFlux and all other DataFlux Corporation LLC product or service names are registered trademarks or trademarks of, or licensed to, DataFlux Corporation LLC in the USA and other countries. ® indicates USA registration.

Apache Portable Runtime License Disclosure

Copyright © 2008 DataFlux Corporation LLC, Cary, NC USA.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Apache/Xerces Copyright Disclosure

The Apache Software License, Version 1.1

Copyright (c) 1999-2003 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:

"This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
4. The names "Xerces" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.
5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright (c) 1999, International Business Machines, Inc., <http://www.ibm.com>. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

DataDirect Copyright Disclosure

Portions of this software are copyrighted by DataDirect Technologies Corp., 1991 - 2008.

Expat Copyright Disclosure

Part of the software embedded in this product is Expat software.

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT

OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

gSOAP Copyright Disclosure

Part of the software embedded in this product is gSOAP software.

Portions created by gSOAP are Copyright (C) 2001-2004 Robert A. van Engelen, Genivia inc. All Rights Reserved.

THE SOFTWARE IN THIS PRODUCT WAS IN PART PROVIDED BY GENIVIA INC AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

IBM Copyright Disclosure

ICU License - ICU 1.8.1 and later [used in dfPower Studio, DataFlux Integration Server, Profile, and Monitor]

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1995-2005 International Business Machines Corporation and others. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

Microsoft Copyright Disclosure

Microsoft®, Windows, NT, SQL Server, and Access, are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Oracle Copyright Disclosure

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates.

PCRE Copyright Disclosure

A modified version of the open source software PCRE library package, written by Philip Hazel and copyrighted by the University of Cambridge, England, has been used by DataFlux for regular expression support. More information on this library can be found at:
<ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/>.

Copyright (c) 1997-2005 University of Cambridge. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Red Hat Copyright Disclosure

Red Hat® Enterprise Linux®, and Red Hat Fedora™ are registered trademarks of Red Hat, Inc. in the United States and other countries.

SAS Copyright Disclosure

Portions of this software and documentation are copyrighted by SAS® Institute Inc., Cary, NC, USA, 2009. All Rights Reserved.

SQLite Copyright Disclosure

The original author of SQLite has dedicated the code to the public domain. Anyone is free to copy, modify, publish, use, compile, sell, or distribute the original SQLite code, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

Sun Microsystems Copyright Disclosure

Java™ is a trademark of Sun Microsystems, Inc. in the U.S. or other countries.

Tele Atlas North American Copyright Disclosure

Portions © 2006 Tele Atlas North American, Inc. All rights reserved. This material is proprietary and the subject of copyright protection and other intellectual property rights owned by or licensed to Tele Atlas North America, Inc. The use of this material is subject to the terms of a license agreement. You will be held liable for any unauthorized copying or disclosure of this material.

USPS Copyright Disclosure

National ZIP®, ZIP+4®, Delivery Point Barcode Information, DPV, RDI. © United States Postal Service 2005. ZIP Code® and ZIP+4® are registered trademarks of the U.S. Postal Service.

DataFlux holds a non-exclusive license from the United States Postal Service to publish and sell USPS CASS, DPV, and RDI information. This information is confidential and proprietary to the United States Postal Service. The price of these products is neither established, controlled, or approved by the United States Postal Service.

Table of Contents

Introduction	1
Conventions Used In This Document	1
DataFlux Reference Publications	1
dfPower Studio Getting Started Guide	3
Installing dfPower Studio	4
System Requirements	4
Supported Databases	4
Supported Databases for Creating Repositories	5
Before You Upgrade	6
Installation Wizard	7
Command-Line Switches	9
Installing Enrichment Data	10
Installing Data Packs	10
Downloading and Installing Data Packs	10
Installing Supplemental Language Support	12
Configuring dfPower Studio to Use the Java Plugin	12
Configuring dfPower Studio	14
Licensing dfPower Studio	14
Annual Licensing Notification	15
Fine-Tuning Performance	16
dfPower Studio Settings	16
Database and Connectivity Issues	20
Quality Knowledge Base Issues	21
Introduction to dfPower Studio	22
Overview	22

Navigator	22
AIC Process and Five Steps to More Valuable Enterprise Data	23
dfPower Studio – Base	24
dfPower Studio - Profile.....	25
dfPower Studio - Integration	26
dfPower Studio - Design	27
dfPower Studio - Other Tools	28
Sample Data Management Scenario	30
Profiling	30
Quality	38
Integration.....	42
Enrichment.....	47
Monitoring.....	50
Technical Support	53
Frequently Asked Questions.....	53
Error Messages	54
Appendix.....	55
Appendix A: dfPower Architect Configuration Directives	55
Appendix B: dfPower Data Access Component Directives.....	58
Glossary	62

Introduction

This section provides basic information about the dfPower Studio product and documentation.

[Conventions Used in this Document](#)

[DataFlux References](#)

Conventions Used In This Document

This document uses several conventions for special terms and actions.

Typographical Conventions

The following typographical conventions are used in this document:

Convention	Description
Bold	Text in bold signifies a button or action
<i>italic</i>	Identifies document and topic titles
monospace	Typeface used to indicate examples of code

Syntax Conventions

The following syntax conventions are used in this document:

Syntax	Description
[]	Brackets [] are used to indicate variable text, such as version numbers
#	The pound # sign at the beginning of example code indicates a comment that is not part of the code

DataFlux Reference Publications

This document may reference other DataFlux publications, including:

- DataFlux dfPower Studio Online Help
- DataFlux Integration Server Administrator's Guide
- DataFlux Integration Server User's Guide

- [DataFlux Expression Language Reference Guide](#)
- [DataFlux Quality Knowledge Base Online Help](#)

dfPower Studio Getting Started Guide

DataFlux dfPower® Studio (dfPower) is a powerful, easy-to-use suite of data cleansing and data integration software applications. dfPower connects to virtually any data source. Any dfPower Studio user, in any department can use dfPower to profile, cleanse, integrate, enrich, monitor, and otherwise improve data quality throughout the enterprise. dfPower Architect, an innovative job flow builder, allows users to build complex management workflows quickly and logically.

This capability allows frontline staff—not IT or development resources—to discover and address data problems, merge customer and prospect databases, verify and complete address information, transform and standardize product codes, and perform other data management processes required by your organization.

Installing dfPower Studio

This chapter describes how to install dfPower® Studio.

Topics in this chapter include:

[Before You Upgrade](#)

[System Requirements](#)

[Supported Databases](#)

[Installation Wizard](#)

[Command-Line Switches](#)

System Requirements

Requirement	Minimum	Recommended
Platforms	Microsoft Windows 2003; Windows XP; Windows Vista; or Windows 7	Microsoft Windows XP Professional
Processor	Intel® Pentium® 4 - 1.2 GHz or higher	Intel Pentium 4 - 2.2 GHz or higher
Memory (RAM)	512 MB	2+ GB
Disk Space	5 GB	10+ GB

Supported Databases

The following databases are supported with dfPower Studio.

Database	Driver
ASCII Text Files	TextFile
Pervasive® Btrieve® 6.15	Btrieve
Clipper	dBASE File
DB2 Universal Database (UDB) v7.x, v8.1, and v8.2 for Linux, UNIX, and Windows	DB2 Wire Protocol
DB2 UDB v7.x and v8.1 for z/OS	DB2 Wire Protocol
DB2 UDB V5R1, V5R2, and V5R3 for iSeries	DB2 Wire Protocol
dBASE® IV, V	dBASE
Microsoft Excel® Workbook 5.1, 7.0	Excel
FoxPro 2.5, 2.6, 3.0	dBase
FoxPro 6.0 (with 3.0 functionality only)	dBase
FoxPro 3.0 Database Container	dBase

Database	Driver
IBM Informix® Dynamic Server 9.2x, 9.3x, and 9.4x	Informix
IBM Informix Dynamic Server 9.2x, 9.3x, and 9.4x	Informix Wire Protocol
Microsoft SQL Server 6.5	SQL Server
Microsoft SQL Server 7.0	SQL Server Wire Protocol
Microsoft SQL Server 2000 (including SP 1, 2, 3 and 3a)	SQL Server Wire Protocol
Microsoft SQL Server 2000 Desktop Engine (MSDE 2000)	SQL Server Wire Protocol
Microsoft SQL Server 2000 Enterprise (64-bit)	SQL Server Wire Protocol
Microsoft SQL Server 2005	SQL Server Wire Protocol
Oracle® 8.0.5+	Oracle
Oracle 8i R1, R2, R3 (8.1.5, 8.1.6, 8.1.7)	Oracle
Oracle 9i R1, R2 (9.0.1, 9.2)	Oracle
Oracle 10g R1 (10.1)	Oracle
Oracle 8i R2, R3 (8.1.6, 8.1.7)	Oracle Wire Protocol
Oracle 9i R1 and R2 (9.0.1 and 9.2)	Oracle Wire Protocol
Oracle 10g R1 (10.1)	Oracle Wire Protocol
Corel® Paradox® 4, 5, 7, 8, 9, and 10	ParadoxFile
Pervasive PSQL® 7.0, 2000	Btrieve
Progress® OpenEdge® Release 10.0B	Progress OpenEdge
Progress 9.1D, 9.1 E	Progress SQL92
Sybase® Adaptive Server® 11.5 and higher	Sybase Wire Protocol
Sybase Adaptive Server Enterprise 12.0, 12.5, 12.5.1, 12.5.2 and 12.5.3	Sybase Wire Protocol
XML	XML



Note: This is a consolidated list of the drivers available for Windows, Linux, and various UNIX platforms. Please consult with DataFlux® for the complete and updated database version and platform support list prior to initiation of your project.

There is a separate list of databases that support the creation of DataFlux Repositories. Refer to the *dfPower Studio Online Help* for a detailed list of databases and versions.

Supported Databases for Creating Repositories

The following databases support the creation of DataFlux repositories:

Database
DB2 Universal Database (UDB) v7.x, v8.1, and v8.2 for Linux, UNIX, and Windows
DB2 UDB v7.x and v8.1 for z/OS

Database
DB2 UDB V5R1, V5R2, and V5R3 for iSeries
IBM Informix® Dynamic Server 9.2x, 9.3x, and 9.4x
IBM Informix Dynamic Server 9.2x, 9.3x, and 9.4x
Microsoft® SQL Server 6.5
Microsoft SQL Server 7.0
Microsoft SQL Server 2000 (including SP 1, 2, 3 and 3a)
Microsoft SQL Server 2000 Desktop Engine (MSDE 2000)
Microsoft SQL Server 2000 Enterprise (64-bit)
Microsoft SQL Server 2005
Oracle® 8.0.5+ Oracle 8i R1, R2, R3 (8.1.5, 8.1.6, 8.1.7)
Oracle 9i R1, R2 (9.0.1, 9.2)
Oracle 10g R1 (10.1)
Oracle 8i R2, R3 (8.1.6, 8.1.7)
Oracle 9i R1 and R2 (9.0.1 and 9.2)
Oracle 10g R1 (10.1)
Sybase® Adaptive Server® 11.5 and higher
Sybase Adaptive Server Enterprise 12.0, 12.5, 12.5.1, 12.5.2 and 12.5.3
Teradata® 12.0



Note: Due to locking issues, file-based and Microsoft® Access based repositories are not recommended.

Before You Upgrade

You do not need to uninstall a previous version of dfPower Studio before you install a new version. By default, new versions install into a separate directory indicated by the version number. Install the new version into a new directory, allowing both the older and the newer versions to operate side by side. Uninstalling dfPower Studio first might delete some of the ODBC drivers you are currently using but have been discontinued in the upgraded software installation.

Action: Create backup copies of your personal resource files, such as jobs and reports that are displayed in the dfPower Studio Navigator.



Note: These files are stored by default in the \Program Files\DataFlux\dfPower Studio\8.2.1\mgmtrsrc directory.

Action: Make a backup copy of your Quality Knowledge Base (QKB) prior to installing a new QKB.



Note: By default, the QKB location is \Program Files\DataFlux\QltyKB. If you choose to install the latest QKB, you can install it into a new directory. Click **Options > QKB Directory** in dfPower Studio and enter the new

location. Alternatively, you can install it into a new directory and use dfPower Customize and the import features of dfPower Studio to selectively import portions of the updated QKB. A third option is to install the new QKB directly into an existing QKB location. The installation has merge functionality that incorporates the updates from DataFlux® into your existing QKB while keeping any changes you have made.

Starting with version 7, the QKB format for storing metadata changed. This change could cause problems during your installation unless a few steps are taken to accommodate the change. Here are a few scenarios:

1. If you install dfPower Studio version 8.2 Service Pack 1 on a machine that has version 6.2 or earlier installed and QKB version 2004D or earlier already installed, the installation process converts the version 6.2 metadata file so it is compatible with version 8.2.1.
2. If you install dfPower Studio version 8.2.1 on a new machine, and then install QKB version 2004D or earlier, you must run a conversion program that converts QKB version 2004D so it can use the new metadata file format. To do this, use the vault_merge.exe application in the root directory of your QKB. Launch it from the command line and use the following syntax:

```
vault_merge --convert <new file> <old file>
```

For example:

```
vault_merge --convert "C:\Program Files\DataFlux\QltyKB\2004D\qltykb.db"  
"C:\Program Files\DataFlux\QltyKB\2004D\qltykb.xml"
```



Note: You will be required to update your repositories to 8.2.1 in order to use dfPower Explorer.

A QKB is not delivered in the same installation as dfPower Studio. You can use an existing Quality Knowledge Base or you can install the most recent version of the Quality Knowledge Base, available at www.dataflux.com/qkb/. If you do choose to download the latest QKB version, install it after you install dfPower Studio.



Note: If you have an earlier version of dfPower Studio, you will be able to load the new QKB files, but will not be able to save changes without upgrading. You will be warned if you attempt to save an older QKB file. A pop-up explains that the save operation will convert the file to a new format that will not be readable in earlier versions of dfPower Studio.

Installation Wizard

Use the following procedure to install your copy of dfPower Studio using the installation wizard.



Note: This procedure is designed for Windows users. Non-Windows users should refer to dfPower Studio Installation command-line switches.

1. Insert the dfPower Studio CD into your computer's CD drive. A screen appears that offers several options. Select **Install dfPower Studio 8.2.1**.
2. From the Windows task bar, click **Start > Run**. The **Run** dialog appears.
3. In the **Open** dialog, type **[your_drive]:\dfPowerStudio.exe**, replacing [your_drive] with the letter for your CD drive. For example, if your CD drive letter is E, type **e:\dfPowerStudio.exe**.
4. Click **Enter**. The setup wizard begins. The first screen offers a general welcome. Click **Next** to continue.



Note: You can also start dfPower Studio by navigating to and double-clicking the **dfPowerStudio.exe** file on the dfPower Studio CD.

5. The **Choose Destination Location** dialog appears. Specify the directory where you want to install dfPower Studio. We suggest you use the default directory. Do not install a new version of dfPower Studio into an older version's directory. For example, do not install version 8.2.1 into an existing 7.1 directory. However, if you need to, you can re-install the same version of dfPower Studio into the same directory. Click **Next** to continue.
6. The **Copy Resources** dialog appears, as shown in the following illustration. If you have a previously installed version of dfPower Studio on your computer, you can make all of your existing reports and jobs available to the new version of dfPower Studio.





Note: Reports created with dfPower Match version 7.0.3 or earlier are not included in this process.

Click **Next** to continue.

7. The dfPower Studio **License Agreement** dialog appears. Use this control to review and accept the software license agreement. Click **Accept** to continue.
8. The **Select Program Manager Group** dialog appears. Specify the name for the Program Manager group that is added to the Windows Start menu. By default, the group is named DataFlux dfPower Studio 8.2.1. Click **Next** to continue.
9. Select your licensing method and location. You may accept the default settings and enter your licensing information later using the License Manager application. Click **Next** to continue and **Next** again to begin the install. More information on licensing is available under [Configuring dfPower Studio - Licensing dfPower Studio](#).
10. Click the checkbox marked **View Release Notes** if you would like to review the release notes when the installation has completed. Click **Finish** when the installation wizard completes. Check for any available dfPower Studio patches or updates, available at <http://www.dataflux.com/Resources/DataFlux-Resources/Customer-Care-Portal/Downloads.aspx>.

Command-Line Switches

The dfPower Studio installation command-line switches allow you to modify the way dfPower Studio installs:

Command	Description
/S	Install dfPower Studio in silent mode. Use this switch in conjunction with the /M switch described below.
/M=<filename>	Use installation variables from an external text file. The available variables are:
MAINDIR=<path>	Specify the dfPower Studio installation location.
COMPONENTS=[A][B][C][D]	Specify the dfPower Studio applications components to install. A= dfPower Verify B= dfPower Match C= dfPower Customize D= dfPower Profile
RUNMDAC=[YES/NO]	Specify if you want to install MDAC (Microsoft Data Access Components).
RUNCONVWIZ=[YES/NO]	Specify if you want to run the Conversion Wizard Utility. The conversion wizard changes dfPower Studio version 4.x-style jobs into version 5.x/6.x-style jobs.  Note: Specifying YES brings up a related dialog during installation.
RUNSASODBC=[YES/NO]	Specify if you want to install the SAS ODBC driver.
RUNODBC=[YES/NO]	Specify if you want to install DataFlux ODBC Drivers.  Note: This variable also specifies if you want to install the sample DataFlux database. If you do not install this database, none of the sample jobs set up in dfPower Studio will work correctly.

Sample installation command line:

```
dfPowerStudio.exe /S /M=dfinst.txt
```

where dfinst.txt contains the following text:

```
MAINDIR=C:\Program Files\DataFlux\dfPowerStudio\8.2.1
COMPONENTS=ABCD
RUNMDAC=NO
RUNCONVWIZ=NO
RUNSASODBC=NO
```

Installing Enrichment Data

This section provides information about how to install enrichment data (data packs) for dfPower Studio.

[Enrichment Data](#)

[Supplemental Language Support](#)

[Java Pluggin](#)

Installing Data Packs

You can use external data from the United States Postal Service (USPS), Canada Post (SERP), QuickAddress Software (QAS), AddressDoctor, Tele Atlas, and more.

If you use external data, install the data using the instructions provided by the appropriate vendor but make a note of the path to each data source. You will need this information to properly configure dfPower Studio.



Important: Do not attempt to install USPS or SERP data older than June 2007 or Geo+Phone data older than Summer 2007 on a machine where dfPower Studio 8.2 SP 1 is installed. Earlier versions attempt to update the architect.cfg file directly, which causes the installation to hang because the architect.cfg file is in Unicode format. The latest installations call out to a program that updates the architect.cfg file and can handle the Unicode format.

- [Data Packs](#)
- [USPS](#)
- [SERP](#)
- [QAS](#)
- [AddressDoctor](#)
- [Geocode](#)

Downloading and Installing Data Packs

Data Packs for data enrichment are available for download on the DataFlux Customer Care Portal. To download data packs, follow these steps:

1. Obtain a user name and password from your DataFlux representative.
2. Log in to the DataFlux Customer Portal at <http://www.dataflux.com/Resources/DataFlux-Resources/Customer-Care-Portal.aspx>.

3. Click **Downloads > Data Updates**.
4. Select the installation file corresponding to your data pack and operating system to download.
5. Close all other Microsoft® Windows® applications.
6. Browse to and double-click the installation file to begin the installation wizard.

If you are installing QAS data, you must enter a license key. When the wizard prompts you for a license key, enter your key for the locale you are installing.



Note: Download links are also available from the dfPower Navigator Customer Portal link in dfPower Studio.

USPS

The USPS Coding Accuracy Support System (CASS) data is available for download from the DataFlux Customer Care Portal.

Select the appropriate platform and data type for your enterprise. You can choose from Verify Data, Verify DPV Data, LACS Data, or eLOT Data. Follow the instructions provided throughout the download process.



Note: All USPS data must be installed in the same directory. Make sure you make a note of the path to your USPS data source.

SERP

Canada Post's Software Evaluation and Recognition Program (SERP) data is available for download from the DataFlux Customer Care Portal.

Select the appropriate platform for your enterprise and follow the instructions provided throughout the download process.



Note: Make sure you make a note of the path to your SERP data source.

QAS

If you are licensed to use QAS, you must acquire the postal reference databases directly from QAS for the countries they support. For more information, contact your DataFlux representative.

Also, refer to the *QuickAddress Batch API Guide* under Documentation on the DataFlux Customer Care Portal.

AddressDoctor (World Address Verification)

For World address verification, you can select one or more country data to download. Go to the DataFlux Customer Care Portal to view the list of countries.

Select the country data you want to download and follow the instructions provided throughout the download process.



Note: Make sure you make a note of the path to your World data source.

Geocode

Geocode data includes US and Canada Geocode and Phone data, Tele Atlas Zip +6 (roof top data), and Tele Atlas Zip +4.

Select the appropriate data and platform for your enterprise and follow the instructions provided throughout the download process.



Note: Make sure you make a note of the path to your data source.

Installing Supplemental Language Support

If you plan to use dfPower Studio for data that includes East Asian languages or right-to-left languages, you must install additional language support. To install these packages:

1. Click **Start > Settings > Control Panel**.
2. Double-click **Regional and Language Options**.
3. In the Regional and Language Options dialog, select the **Languages** tab. Under **Supplemental Language Support**, check the boxes marked **Install files for complex script and right-to-left languages (including Thai)** and **Install files for East Asian languages**.
4. The Windows installer guides you through the installation of these languages packages.

Configuring dfPower Studio to Use the Java Plugin

dfPower Studio or DataFlux Integration Server (DIS) must be properly configured to run jobs containing the dfPower Architect Java™ Plugin node. The following sections explain the requirements for configuring the setup.

Java Runtime Environment

The primary requirement is that the Java runtime environment must be installed on the machine. The Java Plugin currently supports the Sun™ Java runtime environment (JRE™) version 1.4.2 or later. The actual location of install installation is not important as long as the dfPower Architect or DIS process can read the files in the installation. The architect.cfg file should contain a setting called JAVA/VMLIB that references the location of the Java Virtual Machine JVM™ DLL (or shared library on UNIX variants). In the Sun JRE, the location of this file is typically:

```
[JRE install directory]/bin/server/jvm.dll
```

A typical setting in the architect.cfg file on a Microsoft Windows machine might be:

```
JAVA/VMLIB=C:\jre1.4.2\bin\server\jvm.dll
```

If this setting is not configured properly when a job using the Java Plugin runs, you will receive an error that the JVM could not be loaded. Also your Java code must be compiled using a Java Development Kit (JDK™) of the same version or earlier than the JRE version you plan to use to run your job. For example, compiling your code using JDK 1.5 or later and running the code in the Java Plugin using JRE 1.4.2 generates an error that the class file format is incorrect.

For more information, see *DataFlux dfPower Studio Online Help*, Architect - Java Plugin.

Configuring dfPower Studio

This chapter describes how to configure your new dfPower® Studio installation. Topics include:

[Licensing dfPower Studio](#)

[Downloading and Installing Data Packs](#)

[Installing Supplemental Language Support](#)

[Configuring dfPower Studio to Use the Java™ Plugin](#)

Licensing dfPower Studio

The dfPower Studio licensing model uses a License Manager to manage specific licenses over concurrent dfPower instances.

The following is a list of supported platforms for a license server installation:

Platform
AIX® 64-bit - Power PC™ RS/6000®
HP-UX® 64-bit - HP 64-bit
HP-UX 64-bit - Intel® Itanium®
Microsoft® Windows® 32-bit - x86
Red Hat® Enterprise Linux 32-bit - x86 / AMD Opteron™
Red Hat Enterprise Linux 64-bit - Intel Xeon™ / AMD Opteron
Solaris™ 64-bit - SPARC® 64-bit
Solaris 64-bit - AMD Opteron
SUSE® Linux Enterprise Server 32-bit - x86 / AMD Opteron
SUSE Linux Enterprise Server 64-bit - Intel Xeon / AMD Opteron

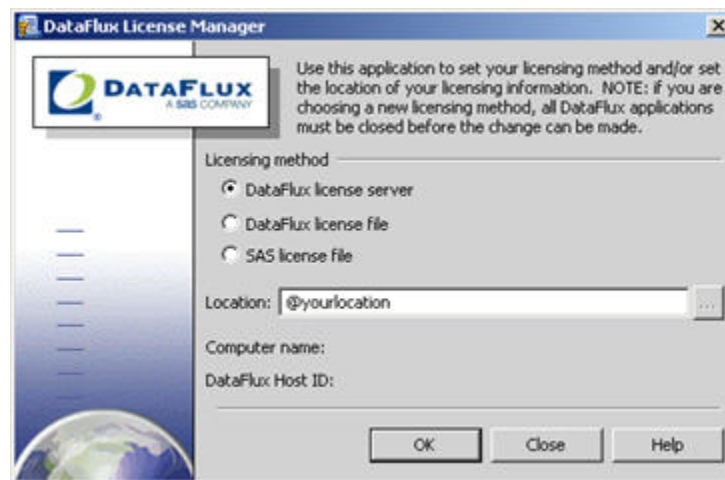
To install the License Server Manager:

1. Download the **License Manager** from <http://www.dataflux.com/Resources/DataFlux-Resources/Custom-Care-Portal/Downloads.aspx>.
2. Install the **License Manager** on your license server by double-clicking the installation package and following the instructions.
3. Run the **lmhostid** command, which generates a machine code.
4. Email the machine code to your DataFlux® representative.

5. Obtain the license file from your DataFlux representative. In Windows, save the license file to your dfPower license directory. In UNIX, save the file to the etc directory.
6. Start the license server.

You can specify the licensing file or server by using the DataFlux License Manager during the dfPower Studio installation or by launching the License Manager after installation is complete.

To specify licensing location using the License Manager, click **Start > Programs > DataFlux dfPower Studio 8.2 > License Manager**. In the License Manager dialog, select the **Licensing Method** and enter the **Location** of your license server or file.



DataFlux License Manager

Annual Licensing Notification

DataFlux products have an annual license model that allow users to access services and run jobs. The system keeps track of the expiration dates for each feature, and a notice alerts users to the impending license expiration using the following process:

1. Sixty days prior to license expiration, a dialog begins appearing daily in dfPower Navigator. The dialog contains a list of the licensed features set to expire, as well as the number of days left for each license. You can select **Do not display this warning again** to disable the warning after it appears at least one time.
2. When the first licensed feature reaches the expiration date, another dialog displays daily, alerting the user that one or more features have expired and these features are now operating within a thirty-day grace period. The list displays the number of days remaining for each feature (or if the license has expired and no longer accesses the product). This notice cannot be disabled.
3. After the thirty-day grace period, services or jobs requested through dfPower Navigator, but have expired, no longer run.

The server log keeps records of all notification warnings generated.

Contact your DataFlux sales representative to renew your DataFlux product license(s).

Fine-Tuning Performance

There are many settings, environment variables, and associated database and system settings that can significantly impact the performance of dfPower® Studio. Aside from normal issues regarding data processing that stem from user hardware and operating-system capabilities, dfPower Studio's performance can benefit either from directly modifying settings within dfPower Studio or from modifying the data source on which dfPower Studio is working.

These modifications can range from the amount of memory used to sort records, to the complexity of the parse definition selected for data processing, to the number and size of columns in a database table. This chapter describes some of the more common ways that you can optimize dfPower Studio for performance.

Topics in this chapter include:

[dfPower Studio Settings](#)

[Database and Connectivity Issues](#)

[Quality Knowledge Base Issues](#)

dfPower Studio Settings

Several dfPower Studio environment settings permit access through the various dfPower Studio applications you use to process your data. Currently, a few of the settings can only be changed by directly editing the HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE keys in the registry. The registry contains many of the user-defined options and system path relationships used by dfPower Studio.



Caution: Edit the registry with care. Unwarranted changes can corrupt your dfPower Studio installation, causing it to perform unexpectedly or fail to initialize.

dfPower Studio (All Applications)

Working Directory Settings: Most dfPower Studio applications use a working directory to create temporary files and save miscellaneous log files. Some processes, such as multi-condition matching, can create very large temporary files depending on the number of records and conditions being processed. To speed processing, the working directory should be a local drive with plenty of physical space to handle large temporary files. Normal hard drive performance issues apply, so a faster hard drive is advantageous, as is keeping the drive defragmented. To change the working directory from the dfPower Studio main screen, double-click **dfPower Settings** on the left-hand panel, and change the path in the **Working Directory** dialog.

dfPower Base – Architect

- **Memory Allocated for Sorting and Joining:** Architect has an option—**Amount of memory to use during sorting operations**—for allocating the amount of memory

used for sorting. Access this option from the Architect main screen by clicking **Tools > Options** then selecting the **Step Specific** tab. The number for this option indicates the amount of memory Architect uses for sorting and joining operations. This number is represented in megabytes, and the default is 64MB. Increasing this number allows more memory to be used for these types of procedures, and can improve performance substantially if your computer has the RAM available. As a rule, if you only have one join or sort step in a Architect job flow, you should not to set this number greater than half the total available memory. If you have multiple uses of sorts and joins, divide this number by the number of sorts or joins in the job flow.

- **Caching the USPS Reference Database:** Architect has an option—**Percentage of cached USPS data indexes**—for how much United States Postal Service data your computer should store in memory. You can access this option from the Architect main screen by clicking **Tools > Options** and selecting the **Step Specific** tab. The number for this option indicates an approximate percentage of how much of the USPS reference data set will be cached in memory prior to an address verification procedure. The default setting is 20. The range is from 0, which indicates that only some index information is cached into memory, to 100, which directs that all of the indices and normalization information (approximately 200MB) be cached in memory. You may also choose to cache all of the USPS reference data by selecting the **Load all USPS data into memory (requires approximately 1 GB of free memory)** check.
- **Sort by ZIP Codes:** For up to about 100,000 records, sorting records containing address data by postal code prior to the address verification process will enhance performance. This has to do with the way address information is searched for in the reference database. If you use Architect to verify addresses, this step can be done inside the application; otherwise, it must be done in the database's system. For over 100,000 records, it might be faster to use the SQL Query step in Architect, write a SQL statement that sorts records by ZIP, and let the database do the work.
- **New Table Creation vs. Table Updates:** When using Architect, it is faster to create a new table as an output step rather than attempting to update existing tables. If you choose to update an existing table, set your commit interval on the output step to an explicit value (for example, 500) instead of choosing **Commit every row**.
- **Clustering Performance in Version 7.0 and Higher:** To take advantage of performance enhancements to the clustering engine that provides the functionality for the Clustering job step in Architect:
 - Try to have as much RAM as possible on your computer, preferably 4GB. If more than 3GB of RAM is installed, the Microsoft® Windows® boot.ini system file needs a special switch to instruct Windows to allow a user process to take up to 3GB. Otherwise, Windows automatically reserves 2GB of RAM for itself. To add this special switch, add /3GB to the Windows boot.ini file.



Note: This setting is not supported on all Windows versions. For more information on making this change and for supported versions, see <http://www.microsoft.com/whdc/system/platform/server/PAE/PAEmem.mspx>.

- Terminate all non-essential processes to free up memory resources.

- Set the **Sort Bytes** memory allocation parameter close to 75–80% of total physical RAM. If the data set is rather small, this might not be as important, but when clustering millions of records, using more memory dramatically improves performance. Do not exceed 80–85% because higher memory allocation might result in memory thrashing, which dramatically decreases clustering performance.
- Defragment the hard drive used for temporary cluster engine files. This is the dfPower Studio working directory described earlier.
- Use the fastest hard drive for cluster engine temporary files. If possible, set the dfPower Studio working directory to be on the fastest physical drive that is not the drive for the operating system or page file.
- Defragment the page file disk. Do this by setting the Windows Virtual Memory size to 0, then rebooting the system and defragmenting the drive.
- Manually set both minimum and the maximum values of the Windows Virtual Memory file size to the same large value. Preferably, this is 2GB or more, depending on disk space available. This prevents the operating system from running out of virtual memory, and prevents the operating system from needing to resize the file dynamically.
- Disable Fast User Switching in Windows XP. This frees up space in the page file.
- **Limit Data Passed Through Each Step:** While it might be convenient to use the Architect setting that passes every output field by default to the next step, this creates a large amount of overhead when you are only processing a few fields. Passing only 10 fields through each step might not be that memory intensive, but when it is 110 fields, performance can suffer. After you create your job flow with the Output Fields setting on **All**, go through each job step and delete the additional output fields you do not need.
- **Large Decimal Support in the Cluster Update Node:** If you are connected only to Oracle, dfPower treats NUMBER(38) columns as an INTEGER. To override this functionality and treat the columns as REAL, you must set the Oracle NUMBER (38) handling in the registry (on Windows) and the dsn configuration file (on UNIX).
 - **Windows** - For Windows, create a key in the HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE section of the registry under SOFTWARE\DataFlux Corporation\dac\[version] with the exact name as your DSN. Under this key, create a DWORD value called or anum38real and set it to 1.
 - **UNIX** - For UNIX, open (or create) the file called dsn.cfg in a .dfpower sub-directory of your home directory (\$HOME/.dfpower/dsn.cfg). Create a line with the exact name of your DSN followed by = or anum38real. If the DSN line already exists, add or anum38real as a new word at the end of the line.

dfPower Integration – Match

- **Multiple Condition Matching:** The number of match conditions, in addition to the number of fields you choose to use as match criteria, will impact performance significantly. A match job of a 100,000 records that uses four conditions with four fields for each condition might take hours to complete, while a match job of 100,000 records with a single condition and six fields might complete in minutes.
- **Generate Cluster Data Mode:** See *Clustering Performance in Version 7.0 and Higher* in *DataFlux® dfPower Studio Online Help*.

dfPower Profile – Configurator

- **Metric Calculation Options:** On the dfPower Profile – Configurator main screen, choose **Job > Options** to access settings that can directly affect the performance of a Profile job process. Two of these settings are **Count all rows for frequency distribution** and **Count all rows for pattern distribution**. By default, all values are examined to determine metrics that are garnered from frequency and pattern distributions. You can decrease processing time by setting a limit on the number of accumulated values. The tradeoff is that you might not get a complete assessment of the true nature of your data.
- **Subset the Data:** You can explicitly specify which metrics to run for each field in a table. If certain metrics do not apply to certain kinds of data, or if you only want to examine select fields of large tables, be sure to manually change the list of metrics to be run for each field. Incidentally, the biggest performance loss is generating a frequency distribution on unique or almost-unique columns. Thus, if you know a field to be mostly unique, you might choose not to perform metrics that use frequency distribution. For numeric fields, these metrics are Percentile and Median. For all fields, these metrics are Primary Key Candidate, Mode, Unique Count, and Unique Percentage. You can also subset the data by creating a business rule or SQL query that can pare down the data to only the elements you want to profile.
- **Sample the Data:** dfPower Profile allows you to specify a sample interval on tables that you want to profile. This feature improves performance at the expense of accuracy, but for very large tables with mostly similar data elements, this might be a good option.
- **Memory Allocated for Frequency Distribution:** dfPower Profile allows you to manually configure the amount of memory being allocated per table column to the Frequency Distribution Engine (FRED). By default FRED allocates 256 KB (512 KB for 64-bit) per column being profiled. This amount (the number of columns * the amount specified per column) is subtracted from the total amount configured by the user in the **Job > Options** menu of dfPower Profile Configurator (Frequency Distribution memory cache size). The amount of memory remaining in the available pool is used for other data operations.

For performance reasons the amount of table memory should always be a power of 2. Setting this value to 1 MB (NOTE: 1 MB = 1024 * 1024 bytes, not 1000 * 1000 bytes) yields optimal performance. Setting it to a value larger than 1 MB (always a power of 2) may help slightly with processing very large data sets (dozens of millions), but could actually reduce performance for data sets with just a few million rows or less. If

you set the amount of table memory too high you may not be able to run your job because it will not be able to initialize enough memory from the available pool.

Database and Connectivity Issues

There are certain issues related to the way databases are physically described and built that can impact the overall performance of dfPower Studio applications. These issues are generally common to all relational database systems. With just a few exceptions, database changes to enhance performance take place internal to dfPower Studio applications.

dfPower Architect

- **Commit Interval:** Architect commit settings are controlled on a per-job basis. The **Data Target (Update)** and **Data Target (Insert)** job steps both have commit options. The default value is **Commit every row**, which commits every change to the database table one record at a time. You can instead select **Commit every N rows** and set this number to somewhere between 10 and 500, or **Commit all rows in a single transaction**. This should increase performance, but on the remote chance there is a problem during processing, all the data changes made by Architect ultimately might not be saved to the source table.

dfPower Profile – Configurator

- **Commit Interval:** Profile commit settings are also controlled on a per-job basis. After selecting your job, click **Tools > Options** and click on the **General** tab. At the bottom, you will find the Commit section. The default value is **Commit all rows in a single transaction**, which helps Profile jobs to run quickly. Note that if there is a problem during processing, data changes made by Profile might not be saved. You can also choose **Commit every N rows** and set this number to somewhere between 10 and 500, or **Commit every row**.

Database

- **ODBC Logging:** Use the Microsoft Windows ODBC Administrator to turn off ODBC Tracing. ODBC Tracing can dramatically decrease performance.
- **Using Primary Keys:** For many dfPower Studio processes, using primary keys will enhance performance only if constructed correctly. Primary keys should be numeric, unique, and indexed. You cannot currently use composite keys to enhance performance.
- **Database Reads and Writes:** All dfPower Studio processes read data from databases, but only some of them write data back. Processes that only read from a database, such as match reports, run quickly. However, if you choose to flag duplicate records in the same table using the same criteria as the match report, the processing time will greatly increase. Using correctly constructed primary keys will help. In addition, as a general rule, smaller field sizes will enhance performance. A table where all fields are set to a length of 255 by default will be processed more slowly than a table that has more reasonable lengths for each field, such as 20 for a name field and 40 for an address field.

- **Turn Off Virus Protection Software:** Some virus-scanning applications such as McAfee® NetShield 4.5 cause processing times to increase substantially. While you might not want to turn off your virus-scanning software completely, you might be able to change some settings to ensure that the software is not harming performance by doing things such as scanning for local database changes.

Quality Knowledge Base Issues

The Quality Knowledge Base (QKB) is the set of file and file relationships that dictates how all DataFlux applications parse, standardize, match, and otherwise process data. The QKB uses several file types and libraries that work with each other to provide expected outputs. A few of the file types are used in ways that do not simply employ direct data look-ups, and it is these types of files that require a certain degree of optimization to ensure that DataFlux applications are processing data as efficiently as possible.

- **Complex Parsing:** Processes using definition types that incorporate parsing functionality—parse, match, standardization, and gender definitions can all use parse functionality—are directly affected by the way parse definitions are constructed. A parse definition built to process email addresses is much less complex than a parse definition that processes address information. This has to do with the specifics of the data type itself. Data types that have more inherent variability most likely have parse definitions that are more complex, and processes using these more complex definitions perform more slowly. When creating custom parse definitions, it is easy to accidentally create an algorithm that is not optimized for performance. Training materials are available from DataFlux that teach the proper way to design a parse definition.
- **Complex Regular Expression Libraries:** Many definitions use regular expression files to do some of the required processing work. Sometimes regular expression libraries are used to normalize data, while other times they are used to categorize data. Incorrectly constructed regular expressions are notorious for being resource intensive. This means you might design a perfectly valid regular expression that takes an extremely long time to accomplish a seemingly simple task. DataFlux-designed definitions have been optimized for performance. If you create custom definitions, be sure to learn how to create efficient regular expressions. Training materials are available from DataFlux that teach Quality Knowledge Base customization.

Introduction to dfPower Studio

This chapter describes dfPower® Studio and its component applications and application bundles. Topics in this chapter include:

[dfPower Studio Overview](#)

[dfPower Studio Navigator](#)

[The AIC Process and the Five Steps to More Valuable Enterprise Data](#)

Overview

You can access dfPower Studio functionality through a graphical user interface (GUI) known as dfPower Studio Navigator, using the command line, or in batch operation mode, providing you flexibility in addressing your data-quality issues. The dfPower Studio main screen is a centralized location from which to launch the dfPower Studio applications. All of the dfPower Studio applications can be accessed through the Tools menu of dfPower Studio Navigator.

dfPower Studio consists of these nodes and application bundles:

- [dfPower Studio — Base](#)
- [dfPower Studio — Profile](#)
- [dfPower Studio — Integration](#)
- [dfPower Studio — Design](#)
- [dfPower Studio — Other Tools](#)

These application bundles are described in greater detail later in this guide.



Note: Some applications will not be available for launching if you have not licensed them.

Navigator

When you launch dfPower Studio, the Navigator window appears.

The Navigator is an essential part of the dfPower solution. The Navigator was designed to help you collect, store, examine, and otherwise manage the various data quality and integration logic and business rules that are generated and created during dfPower Studio use.

Quality Knowledge Bases, Management Resources, and Reference Sources are all containers for metadata that provide you a complete view of your data quality assets and help you build complicated data management processes with ease.

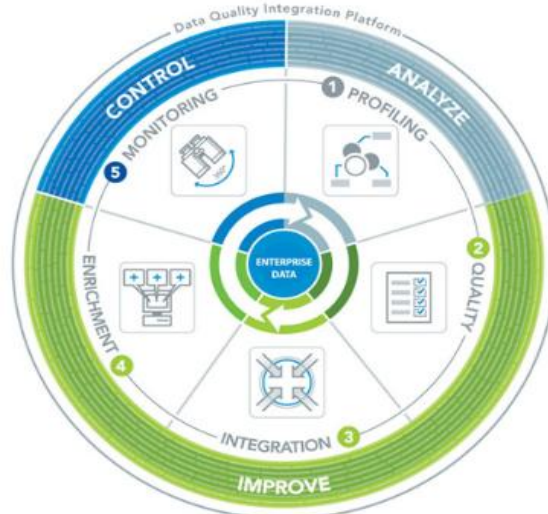
Use dfPower Studio to incorporate data quality business rules that are used across an entire enterprise within many different aspects of business intelligence and overall data hygiene. These business rules are stored as objects that are independent of the underlying data sources used to create them. Thus, there is no limit to the application of these business rules to other sources of data throughout the enterprise, including internal data sets, external data sets, data entry, internal applications, the web, and so on. From the smallest spreadsheets buried in the corners of the enterprise, all the way up to corporate operational systems on mainframes, you can use the same business rules to achieve consistent data integrity and usability across the entire enterprise.

More specifically, the Navigator:

- Stores all business rules generated during any of the various dfPower Studio processes
- Allows metadata to be maintained for each of these data quality objects
- Allows reuse of these data quality jobs and objects across the various applications
- Facilitates quick launches of various stored data management jobs with a few clicks of the mouse
- Maintains all reports that are generated during data quality processing
- Manages various configurations of the various data quality processes routinely implemented by the organization
- Maintains information about batch jobs and the various schedules of these jobs

AIC Process and Five Steps to More Valuable Enterprise Data

The dfPower Studio bundles have been designed to directly support the Analyze, Improve, and Control (AIC) process and the building blocks of data management. AIC is a method of finding data problems, building reusable transformations, and strategically applying those transformations to improve the usability and reliability of an organization's data.



DataFlux Methodology

The five steps to more valuable enterprise data of AIC data management are:

- data profiling
- data quality
- data integration
- data enrichment
- data monitoring

As you will see, these steps closely mirror the structure of dfPower Studio's main menu. For more information, see the DataFlux white paper entitled, *Five Steps to More Valuable Enterprise Data*.

dfPower Studio – Base

At the heart of dfPower® Studio is the bundle of applications and functions collectively known as dfPower Studio - Base. dfPower Studio - Base is the infrastructure that ties all the dfPower Studio applications together, allowing them to interact, process data simultaneously at scheduled times, and share information where necessary. To access these applications using the main dfPower Studio menu, click **Tools > Base**.

dfPower Studio - Base consists of the following components:

- [Architect](#) — Design a workflow for processing your data
- [DB Viewer](#) — View and retrieve records from your data sources
- [Business Rule Manager](#) — Create and manage business rules
- [Monitor Viewer](#) — View task data in a repository

Architect

dfPower Architect brings much of the functionality of the other dfPower Studio applications (such as dfPower Base - Batch and dfPower Profile - Viewer), as well as some unique functionality, into a single, intuitive user interface. In Architect, you define a sequence of operations—for example, selecting a source data, parsing that data, verifying address data, and outputting that data into a new table—and then run the operations at once. This functionality not only saves you the time and trouble of using multiple dfPower Studio applications, but also helps to ensure consistency in your data-processing work.

To use Architect, you specify operations by selecting job flow steps and then configuring those steps, using straightforward settings, to meet your specific data needs. The steps you choose are displayed on dfPower Architect's main screen as node icons, together forming a visual job flow.

Architect can read data from virtually any source, including data processed and output to a text file, HTML reports, database tables, and a host of other formats.

DBViewer

dfPower DBViewer is a record viewer that permits you to view and retrieve records from your various data sources.

Business Rule Manager

Use the Rule Manager to create and manage business rules and custom metrics to add them to a job to monitor the data. You can use business rules and custom metrics to analyze data to identify problems. The Rule Manager provides tools for building expressions for custom metrics and rules. You can create custom metrics to supplement the standard metrics available in dfPower Profile or to be used in rules and tasks with dfPower Architect. Use the Rule Manager to create tasks that implement one or more rules. You can then execute these tasks at various points in a dfPower Architect job flow and trigger events when the conditions of a rule are met.

Monitor Viewer

When you run a job in dfPower Architect, the job executes tasks and rules that you created using the Business Rule Manager. Information about these executed tasks and their associated rules are stored in the default repository. The Monitor Viewer lets you view the information stored in the repository about the tasks and rules executed.

dfPower Studio - Profile

Start any data quality initiative with a complete assessment of your organizations information assets. Use dfPower® Profile to examine the structure, completeness, suitability and relationships of your data. Through built-in metrics and analysis functionality, you can find out what problems you have, and determine the best way to address them.

Using dfPower Profile, you can:

- Select and connect to multiple databases of your choice without worrying about your sources being local, over a network, on a different platform, or at a remote location.
- Create virtual tables using business rules from your data sources in order to scrutinize and filter your data.
- Run multiple data metrics operations on different data sources at the same time.
- Run primary/foreign key and redundant data analyses to maintain the referential integrity of your data.
- Monitor the structure of your data as you change and update your content.

The following tools are available within Profile.

Configurator

The Configurator screen appears when you first start Profile. Use the Configurator to set up jobs to profile your data sources.

Viewer

Allows you to view the results of a dfPower Profile job.

Explorer

dfPower Explorer is a relationship diagramming tool that allows you to display matching tables and the relationships between them. The relationship arrows can be removed to invalidate a relationship. To add a new relationship, the user can drag an arrow from one table column to the matching table's column.

Scheme Builder

Use the Scheme Builder to build a standardization scheme from an analysis report, edit existing schemes, or to create new schemes.

dfPower Studio - Integration

As data integration power tools, Match and Merge help to ensure database consistency and reduces costs associated with duplicate records.

Match

Match uses record merging along with clustering algorithms to develop a match code that identifies duplicate or near-duplicate records. The match sensitivity, which determines how precisely the match codes are calculated, is flexible and can be adjusted in the tool itself. The Match tool begins the process by parsing the data into tokens using grammars. Once tokens are available, it standardizes them, removing ambiguities. Based on sensitivity settings, it then creates match codes from tokens that later are used in the final merging and clustering step. Match handles this process internally, without user interaction. Regardless of the data source, users simply set within the tool the criteria to be used for

matching. For instance, in the beginning the user may set a simple matching criterion to match on name; later this can be elaborated to name and address, or name or address.

Merge

Merge is used in conjunction with Match to identify and combine duplicate records. After employing Match to identify duplicate records, use dfPower® Merge to manually review and edit the surviving record for each cluster of duplicate records.

dfPower Studio - Design

Designed for advanced users, dfPower® Design is a development and testing bundle of applications for creating and modifying data-management algorithms (such as algorithms for matching and parsing) that are surfaced in other DataFlux® products. dfPower Design provides several GUI components to facilitate this process, as well as extensive testing and reporting features to help you fine tune your data quality routines and transformations. Some of the dfPower Design tasks you can perform include:

- Creating data types and processing definitions based directly on your own data.
- Modifying DataFlux-designed processing definitions to meet the needs of your data.
- Creating and editing regular expressions (regex library) to format and otherwise clean inconsistent data.
- Creating and editing phonetics library files, which provide a means for better data matching.
- Creating and modifying extensive look-up tables (vocabularies) and parsing rule libraries (grammars) used for complex parsing routines.
- Creating and modifying character-level chop tables used to split apart strings of text according to the structure of the data.
- Creating and modifying transformation tables (schemes) that you can apply on a phrase or element-level basis.
- Testing parsing, standardization, and matching rules and definitions in an interactive or batch mode before you use them to process live data.
- Generating extensive character-level and token-level reporting information that allows you to see exactly how a string is parsed, transformed, cleaned, matched, or reformatted.
- Creating data types and processing definitions specific to your locale.

Customize

Customize is used to modify or create data quality algorithms used by DataFlux products and SAS Data Quality Server.

Parse Definition Quick Editor

The dfPower Customize Parse Definition Quick Editor uses customer data to automatically build chop tables, vocabularies, and grammars.

Vocabulary Editor

This tool allows you to create and modify collections of words used to compare data. Each word in the vocabulary is defined as belonging to one or more categories, which are defined in an associated grammar.

Grammar Editor

The Grammar Editor allows you to set up rules that define patterns of words.

Regex Editor

The Regex Editor is a tool for creating and modifying regular expressions. In the context of parse definitions, standardization definitions, and match definitions, regular expressions are primarily intended for character-level cleansing and transformations. For word- and phrase-level transformations, you should instead use standardization data libraries.

Phonetics Editor

The Phonetics Editor allows you to create rules to identify similar-sounding data strings.

Chop Table Editor

Chop tables create an ordered word list from an input string through the use of character-level rules.

For more detailed information on available Customize functions, refer to *DataFlux dfPower Studio Online Help*.

dfPower Studio - Other Tools

License Manager and Copy QAS Configuration are two additional tools to assist you in configuring dfPower® Studio.

License Manager

Use this application to set your licensing method or change the location of your licensing server. If you make changes here, you must restart all DataFlux® applications before changes will take effect. Options for licensing method are:

- DataFlux license server
- DataFlux license file
- SAS® license file

Copy QAS Configuration

Select **Copy QAS Configuration** on the **Other** menu if you wish to copy or replace your QAS configuration.

Sample Data Management Scenario

This chapter takes you through a sample data management scenario and provides an overview of dfPower® Studio. The sample data management scenario highlights the typical procedures in a data-management project, following the DataFlux® five-building-block methodology: Profiling, Quality, Integration, Enrichment, and Monitoring.

Other Sources

- This chapter highlights some typical procedures in a data-improvement project. For step-by-step details of the procedures, please refer to the DataFlux *dfPower Studio Online Help*.
- The first scenario in this chapter follows a straight-line path through the DataFlux methodology. In practice, however, this methodology is highly iterative; as you find problems, you will dig deeper to find more problems, and as you fix problems, you will find more problems to fix.
- Throughout the data management scenario, we mention the dfPower Studio applications we use to accomplish each task. In most cases, a task can be performed by more than one application, although the available options and specific steps to accomplish that task might differ. For example, you can profile data with both dfPower Profile and dfPower Architect's Basic Statistics and Frequency Distribution job steps.

Data Management Scenario Background

An organization has a Contacts database and a Purchase database that have been maintained independently. The organization needs to integrate the customer records from both sources. Our job is to:

- ensure that all records follow the same data standards,
- identify and merge duplicate records, and
- prepare the records for an upcoming mailing to all customers

First, we will perform data discovery known as [Profiling](#).

Profiling

Profiling is a proactive approach to understanding your data. Also called data discovery or data auditing, data profiling helps you discover the major features of your data landscape: what data is available in your organization and the characteristics of those data.

In preparing for an upcoming mailing, we know that invalid and non-standard addresses will cause a high rate of returned mail. By eventually standardizing and validating the data, we will lower our risks of not reaching customers and incurring unnecessary mailing costs. Also, by understanding the data structure of customer records, we will be better able to join, merge, and de-duplicate those records.

Where are your organization's data? How do data in one database or table relate to data in another? Are your data consistent within and across databases and tables? Are your data complete and up to date? Do you have multiple records for the same customer, vendor, or product? Good data profiling serves as the foundation for successful data-management projects by answering these questions up front. After all, if you do not know the condition of your data, how can you effectively address your data problems?


Profiling helps you determine what data and types of data you need to change to make your data usable.

Data Profiling Discovery Techniques

The many techniques and processes used for data profiling fall into three major categories:

- [Structure Discovery](#)
- [Data Discovery](#)
- [Relationship Discovery](#)

The next three sections address each of these major categories in our data management scenario.

 **Caution:** As you profile your own data, resist the urge to correct data on the spot. Not only can this become a labor-intensive project, but you might be changing valid data that only appears invalid. Only after you profile your data will you be fully equipped to start correcting data efficiently.

Structure Discovery

The first step in profiling the data is to examine the structure of that data. In structure discovery, we determine if:

- our data match the corresponding metadata
- data patterns match expected patterns
- data adhere to appropriate uniqueness and null-value rules

Discovering structure issues early in the process can save much work later on, and establishes a solid foundation for all other data-management tasks. To profile the organization in this example, do the following:

1. To launch the Configurator, click **Tools > Profile > Configurator**.
2. In dfPower® Profile - Configurator, select the **DataFlux® Sample** database and click **Contacts**.



Note: By default, the DataFlux Sample database is located at
C:\Program Files\DataFlux\dfPower Studio\8.2.1\sample\dfDemo.mdb.

3. To select all of the fields, click the checkbox next to each.
4. From the main menu, click **Job > Select Metrics**.

5. Select **Frequency distribution**, **Pattern frequency distribution**, **Percentiles**, and **Outliers**. Click **Select/unselect all** to select all Column profiling metrics.
6. Click **Job > Run Job**. Save job as **ContactsProfile1**.

Here is some of the information we can see about each data field:

- Column profiling: For each field, a set of metrics such as data type, minimum and maximum values, null and blank counts, and data length.
- Frequency distribution: How often the same value occurs in that field, presented both as text and in a chart.
- Pattern distribution: How often the same pattern occurs in that field, presented both as text and in a chart.

The information we can glean from each metric depends on the field. For example, we look at some column profiling metrics for the Contacts field that specifies the customer's ID:

Metric Name	Metric Value
Data Type	VARCHAR
Unique Count	3276
Pattern Count	5
Minimum Value	1
Maximum Value	999
Maximum Length	5
Null Count	0
Blank Count	0
Actual Type	integer
Data Length	9 chars

These metrics highlight some issues:

- The official type set for the field is VARCHAR, but the actual values are all integers. While this is not a major data problem, it can slow processing.
- The maximum length of the actual data values is 5, but the data length reserved for the field is 9. That is 4 characters of reserved but unused space for each record.



Note: We will look at the Unique Count, Minimum Value, and Maximum Values later in Data Discovery.

Next, we look at the pattern frequency distribution for this field:

Pattern	Alternate	Count	Percentage
9999	9(4)	2276	69.47
999	9(3)	900	27.47
99	9(2)	90	2.75

Pattern	Alternate	Count	Percentage
9	9	9	0.27
99999	9(5)	1	0.03

The table above shows those IDs expressed with different numbers of digits. There does appear to be one ID that is longer than needed to have uniqueness with this number of records.



Note: dfPower Studio expresses patterns using "9" for a digit, "A" for an uppercase letter, "a" for a lowercase letter, and spaces and punctuation marks for spaces and punctuation marks. In addition, dfPower Studio also expresses alternate shorthand patterns, where each "9," "A," and "a" is followed by a number in parentheses that provides a count of each character type at that location in the value, unless that count is 1.

Value	Pattern	Alternate
1	9	9
1997	9999	9(4)
1-919-555-1212	1-999-999-9999	9-9(3)-9(3)-9(4)
(919) 555-1212	(999) 999-9999	(9(3)) 9(3)-9(4)
Mary	Aaaa	Aa(3)
Mary Smith-Johnson	Aaaa Aaaaa-Aaaaaaa	Aa(3) Aa(4)-Aa(6)

We continue to review these metrics for every field in this table, making notes on each data problem we find. We also review metrics for the fields in our example, where we find additional data problems. For example, the PHONE field uses two different patterns.

Data Discovery

Our second data profiling step, data discovery, helps us determine whether our data values are complete, accurate, and unambiguous. If they are not, this can prevent us from adequately recognizing the customer needs, and can make it difficult to tie these records to other data.

We take a look at a set of metrics for the STATE field of the Purchase table. This time, we focus on the Unique Count, Minimum Value, and Maximum Value metrics, and we discover some discrepancies:

- The Minimum Length and Maximum Length are each 2, but 255 characters are dedicated to this field.
- There are 64 unique values, but there are only 50 states. Even including all Canadian provinces and territories would only bring the possible total up to 63. It is likely that some of these are incorrect.

We do this analysis for every field for both tables.

Next, we look at part of the frequency distribution for each field. The following table shows frequency distribution for The Company's Purchase table's STATE field.

Value	Count	Percentage
(null value)	3596	71.89
CA	131	2.62
TX	111	2.22
MA	61	1.22
FL	60	1.20
NY	54	1.08
ON	54	1.08
GA	50	1.00
NJ	48	0.96
IL	46	0.92
PA	45	0.90
TN	42	0.84
...

We already determined that there are 64 unique values for the STATE field. Now we can see the actual values, and they are all valid. They include US states, the District of Columbia, Canadian provinces, Puerto Rico, Virgin Islands, and Guam.

Next, we look at these same count metrics for both tables across several fields:

	ID	COMPANY	CONTACT	ADDRESS	CITY	STATE	PHONE
Count	3276	3276	3276	3276	3276	3276	3276
Null Count	0	0	0	124	0	37	11
Unique Count	3276	1407	3243	2389	2804	61	3220

	ID	COMPANY	CONTACT	ADDRESS	CITY	STATE	PHONE
Count	5002	5002	5002	5002	5002	5002	5002
Null Count	0	1726	1726	1850	4311	3596	681
Unique Count	5002	1293	3243	2389	407	64	2890

The two preceding tables clearly show that while the Contacts table is nearly complete, there is a large amount of data missing from the Purchase table. In particular, null counts greater than 0 show that data is missing for COMPANY, CONTACT, ADDRESS, CITY, STATE, and PHONE.

Looking at the Count and Unique Count values, we discover that both values are equal for both tables' ID fields. This means we can use this field to uniquely identify records.

To find other issues, we review data length and type metrics for both tables across several fields:

	ID	COMPANY	ADDRESS	CITY	STATE	DATE
Data Length	9	50	100	30	15	19
Maximum Length	6	40	30	28	14	19
Data Type	VARCHAR	VARCHAR	VARCHAR	VARCHAR	VARCHAR	DATETIME

	ID	COMPANY	ADDRESS	ADDRESS2	CITY	STATE	ORDER DATE
Data Length	10	255	255	255	255	255	255
Maximum Length	10	38	34	37	19	2	10
Data Type	VARCHAR	VARCHAR	VARCHAR	VARCHAR	VARCHAR	VARCHAR	VARCHAR

These metrics highlight differences in data length between the tables:

- The Contacts table stores customers' addresses in a single ADDRESS field, while the Purchase table uses separate ADDRESS and ADDRESS2 fields.
- While many of the other fields in both tables have the same name, the fields are of different lengths. For example, the CITY Data Length is 30 for Contacts and 255 for Purchase. Also, the STATE Maximum Length is 14 for Contacts, which is larger than the STATE field in the Purchase table, which has a Maximum Length of 2.
- Similar fields also have different data types between the two tables. For example, the Contacts DATE field is set to DATETIME, while the Purchase ORDER DATE field is set to VARCHAR.

Next, take a look at the actual records. To do this from the report in dfPower Profile's Viewer component, we select a field with questionable metrics, and then double-click on that metric. For example, to view records with "CA." as the STATE value, in the Viewer we select the STATE field, show the frequency distribution for that field, then double-click on the "CA." value. The following table shows selected field values for five customer records in the Contacts table where the STATE value is questionable:

ID	Company	Contact	Address	City	State	Phone
320	Gates Rubber Co	W. Nickels	3402 Falcon Ridge Rd	Round Lake	CA.	390-766-5114
316	Gates Rubber Company	Kelly Tullman	29 Amaranth Dr	Mogote	California	237-201-0357
1168	Federal Express	Amanda Oliveira	152 Kenton St J223	North Fillmore	North Car.	265-567-3614
1301	Ernest W Hahn Inc.	Janet Wenner	530 E Patriot Blvd	Frederiksted	Hawaai	240-484-1695
1270	Epter Norton Computing	Y Mc Hugh	3983 S McCarran Blvd	Green Bay	ohioo	352-193-4453

By looking through a sampling of records with suspect data, we discover some problems metrics did not show:

- The STATE values for ID=1168, 1301, and 1270 are invalid.
- The STATE values for ID=320 and ID=316 are valid but inconsistent. Most records in the Contacts and Purchase tables contain a two-letter abbreviation in the STATE field.
- The area code for ID=320, 316, and 1168 are invalid because 390, 237, and 265 are not used as area codes.
- The area code for ID=1301 and ID=1270 are invalid because 240 is a code for Maryland and 352 is a code for Florida.
- The COMPANY name is inconsistent. For example, ID=320 uses Gates Rubber Co while ID=316 uses Gates Rubber Company.
- The CONTACT for ID=1270 has a space between the Mc and Hugh in the CONTACT field's last name.

Relationship Discovery

Our final data profiling step, relationship discovery, can help us answer these questions: Does the data adhere to specified required key relationships across columns and tables? Are there inferred relationships across columns, tables, or databases? Are there redundant data?

Here are some other problems that can result from the wrong relationships:

- A product ID exists in your invoice register, but no corresponding product is available in your product database. According to your systems, you have sold a product that does not exist.
- A customer ID exists on a sales order, but no corresponding customer is in your customer database. In effect, you have sold something to a customer with no possibility of delivering the product or billing the customer.
- You run out of a product in your warehouse with a particular UPC number. Your purchasing database has no corresponding UPC number. You have no way to restock the product.

For the purposes of demonstrating relationship discovery in our scenario, let us say we know from earlier profiling that while Contacts contains unique COMPANY values, some values for COMPANY in the Purchase table are invalid because Purchase was not populated with data from the Contacts table. To help determine how much work might be involved in identifying and fixing the invalid COMPANY values in Purchase table records, we will run two analyses:

- A redundant data analysis determines how many values are common between the Contacts COMPANY field records and the Purchase COMPANY field records.
- A primary key/foreign key analysis shows us which specific values in the customer records do not match, and thus are likely to be the invalid codes.

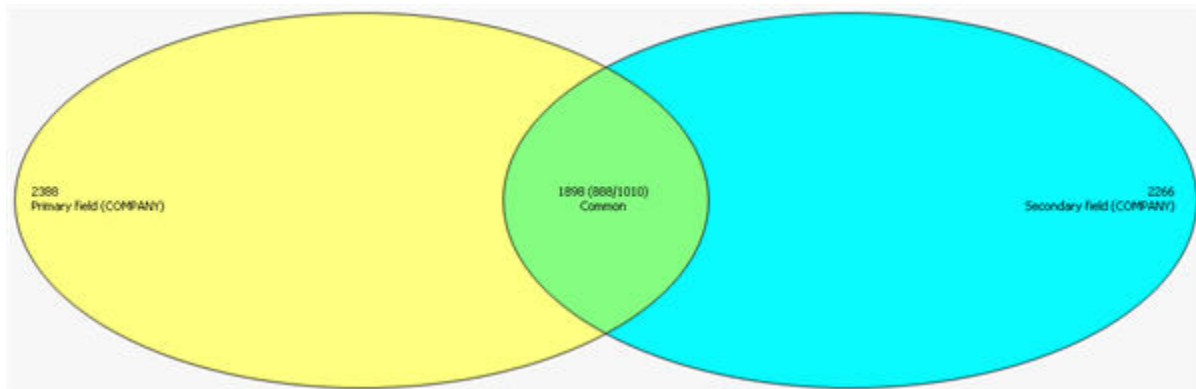
To run our redundant analysis, we start with dfPower Profile's Configurator component. On the **Configurator** main dialog, right-click on the COMPANY field in the Contacts table and

choose **Redundant Data Analysis**. In the screen that appears, select the COMPANY field in the Purchase table. Then, run the job, and the results appear in dfPower Profile's Viewer component. In the Viewer, select the Contacts table and COMPANY field and display the Redundant Data Analysis results:

Field Name	Table Name	Primary Count	Common Count	Secondary Count
COMPANY	Purchase	2388	1898	2266

The results tell us that in the Contacts table 1898 records have COMPANY values that are also in the Purchase table's COMPANY field, the Purchase table contains 2388 codes that are not used by the Contacts table, and the Contacts table has 2266 occurrences of COMPANY values that do not occur in the Purchase table. This means we need to update 2266 records in the Purchase customer table.

Click on the COMPANY field on the Redundant Data Analysis tab to reveal a Venn diagram showing an alternate view of the data:



Redundant Data Analysis Venn Diagram

Now let us look at the same tables using primary key/foreign key analysis. To do this, we again start with dfPower Profile's Configurator component. On the **Configurator** main dialog, we right-click on the COMPANY field in Contacts table and choose **Primary Key/Foreign Key Analysis**. In the screen that appears, we select the COMPANY field in the Purchase table. Then, run the job, and the results appear in dfPower Profile's Viewer component. In the Viewer, select the Contacts table's COMPANY field and display the Primary Key/Foreign Key Analysis results:

Field Name	Table Name	Match Percentage
COMPANY	Purchase	20.19

Outliers for field: COMPANY	Count	Percentage
Northrop Grumman Corp	25	0.63
Loma Linda Univ Medical Ctr	16	0.40
Taco Bell Corp	11	0.28
Hunt-Wesson Inc	8	0.20
Metropolitan Water Dist	8	0.20

Outliers for field: COMPANY	Count	Percentage
Naval Medical Ctr	7	0.18
Good Samaritan Regional Med Ctr	7	0.18

The Match Percentage confirms that of all the COMPANY VALUES in the Purchase table's customer records, 20.19% are also in the Contacts table. Much more valuable for our purposes, though, is the list of outliers for the COMPANY field. We can now see that although 79.81% of the COMPANY values in the Purchase table's customer records are invalid, many values are repeated with slight variations in spelling. This could be pretty good news if the invalid values are consistent; if so, we only need to figure out how the invalid values map to valid ones.

With all that we have learned about our data in data profiling—through structure discovery, data discovery, and relationship discovery—we are now ready to move on to the next data-management building block, [Quality](#).

For more information on Profiling, look for "Profile" in the *DataFlux dfPower Studio Online Help*.

Quality

Using the next data management building block—Quality—we start to correct the problems we found through profiling. When we correct problems, we can choose to correct the source data, write the corrected data to a different field or table, or even write the proposed corrections to a report or audit file for team review. For example, we might leave the Contacts customer table untouched, writing all the corrected data to an entirely new table.

Typical operations to improve data quality include:

- [Parsing](#)
- [Standardization](#)
- [Matching](#)

The goal of parsing and standardization is to improve overall consistency, validity, and usability of the data. Quality can also include creating match codes in preparation for joining and merging in the [Integration](#) building block.

Parsing

Parsing refers to breaking a data string into its constituent parts based on the data's type. Parsing is usually done in preparation for later data management work, such as verifying addresses, determining gender, and integrating data from multiple sources. Parsing can also improve database searches as well as increase their speed.

For our data, parsing the Contacts table's CONTACT field into separate First Name and Last Name fields helps us in two ways:

- It makes integrating the Contacts and Purchase name data easier.
- For an upcoming customer mailing, it allows us to address customers in a more personal way. For example, we can open a letter with Dear Edward or Dear Mr. Smith instead of Dear Edward Smith.

Parsing Techniques

Using the dfPower® Architect application, we parse the name of our customers. To parse:

1. Launch dfPower Architect from the dfPower Base menu.
2. Use the Data Source job step to add the Contacts table to the dfPower Architect job flow.
3. Add a Parsing step.
4. Indicate that dfPower Architect should look in the CONTACT field for both first and last names.
5. Output the parsed values into separate FIRST NAME and LAST NAME fields.

The following table shows CONTACT values after parsing:

CONTACT	FIRST NAME	LAST NAME
James E. Briggs	James	Briggs
Bob Brauer	Bob	Brauer
LUTHER BAKER	LUTHER	BAKER
Irene Greaves	Irene	Greaves
Rob Drain	Rob	Drain
...
G. Weston	G.	Weston
...
Shannon Osterloh	Shannon	Osterloh

Notice that we did not remove the CONTACT values. We just added fields and values for FIRST NAME and LAST NAME. Also, note that dfPower Architect disregarded the "E." in James E. Briggs since we did not include Middle Name in our Parsing step.

With FIRST NAME in its own field, we can now use dfPower Architect for gender analysis. To do this, we add a Gender Analysis (Parsed) step and indicate that dfPower Architect should look in the new FIRST NAME field to determine gender and output updated gender values back to a new GENDER field. Our data now looks like this:

CONTACT	GENDER	FIRST NAME	LAST NAME
James E. Briggs	M	James	Briggs
Bob Brauer	M	Bob	Brauer
LUTHER BAKER	M	LUTHER	BAKER
Irene Greaves	F	Irene	Greaves

CONTACT	GENDER	FIRST NAME	LAST NAME
Rob Drain	M	Rob	Drain
...
G. Weston	U	G.	Weston
...
Shannon Osterloh	U	Shannon	Osterloh

This time, the GENDER field was updated. Note that Shannon Osterloh is marked as a U because Shannon is a name given to both men and women. Also note that the Gender value for G. Weston is unknown because an initial is not enough information to determine gender.

We could have used the dfPower Architect Gender Analysis step both to parse the CONTACT field and identify gender in the same step. However, we would not then been able to use the FIRST NAME and LAST NAME fields, which we know we will need later for our mailing. We also could have used the Gender Analysis step if the CONTACT field contained first, middle, and last names; this approach would allow Architect to identify gender even better by using both first and middle names.

Standardization

While profiling our data, we discovered several inconsistencies in the CONTACTS, STATE, and PHONE fields of the Contacts table.

Let us look at a sampling of data, focusing on the STATE field:

FIRST NAME	LAST NAME	ADDRESS	STATE
James	Briggs	19 East Broad Street	Missouri
Bob	Brauer	6512 Six Forks Road - 404B	North Carolina
LUTHER	BAKER	3560 E 116TH ST	CA
Irene	Greaves	555 W 5th St	OH
Rob	Drain	7718 Elder Way	OH
...
Nancy	Weinstock	2134 Estrado Cir	

To standardize these designations, we again use the dfPower Architect application. Continuing the job flow we created for parsing and gender identification, we add a Standardization job step. For this job step, we instruct dfPower Architect to use the DataFlux®-designed Address standardization definition for the Street Address field and the State (Two Letter) standardization definition for the State field. Both standardization definitions change the addresses to meet US Postal Service standards.


After standardizing these two fields, here is how our records look:

FIRST NAME	LAST NAME	ADDRESS	STATE
James	Briggs	19 E Broad St	MO
Bob	Brauer	6512 Six Forks Rd 404B	NC
LUTHER	BAKER	3560 E 116Th St	CA

FIRST NAME	LAST NAME	ADDRESS	STATE
Irene	Greaves	555 W Fifth St	OH
Rob	Drain	7718 Elder Way	OH
...
Nancy	Weinstock	2134 Estrado Cir	

Notice that one of the STATE fields is still blank. Standardization confirms existing values to a standard, but does not fill in missing values.

One last thing before we leave Standardization: As you might recall, in Data Discovery, we saw that the Contacts table's PHONE field uses a 999-9999-999 pattern, while the Purchase table's PHONE field is inconsistent. We will skip the details here, but to transform one of the patterns to another, we use dfPower Customize to create a new standardization definition, and then use that new definition in a dfPower Architect Standardization job step. For more information, see *dfPower Studio Online Help - Customize*.

 **Note:** Assuming we had all the required standardization definition schemes and definitions at the outset, the most efficient way to standardize all three fields—ADDRESS, STATE, and GENDER—is to use one Standardization job step in dfPower Architect, and assign the appropriate definition or scheme to each field.

Matching

Matching seeks to uniquely identify records within and across data sources. Record uniqueness is a precursor to Integration activities.

While profiling, we discovered some potentially duplicate records. As an example, we have identified three Contacts records that might be all for the same person, as shown below:

	Record 1	Record 2	Record 3
FIRST NAME	John	John	JOHN
LAST NAME	Doe	Doe	DOE
ADDRESS	75 Columbus Avenue	75 Columbus Ave	3684 TRAVER RD
STATE	MI	Mich	CA

To help determine if these records are indeed for the same customer, we launch the dfPower Integration – Match application; select the Contacts table; assign the appropriate DataFlux-designed and custom-created match definitions to the fields we want considered during the match process, set match sensitivities and conditions, and run an Append Match Codes job. The result is a new field containing match codes, as shown below.

	Record 1	Record 2	Record 3
FIRST NAME	John	John	JOHN
LAST NAME	Doe	Doe	DOE
ADDRESS	75 Columbus Avenue	75 Columbus Ave	3684 TRAVER RD

	Record 1	Record 2	Record 3
STATE	MI	Mich	CA
MATCH CODE	GHWS\$\$EWT\$	GHWS\$\$EWT\$	GHWS\$\$WWI\$

Based on our match criteria, dfPower Integration – Match considers Record 1 and Record 2 to be for the same person. Record 3 is very similar—note how the first four characters of the match code () are the same as Records 1 and 2—but the ADDRESS is different.

Of course, we could have figured this out manually, but it gets much more difficult when you need to make matches across thousands of records. With this in mind, we create match codes for both tables.

Now that we have finished parsing, standardizing, and creating match codes for both tables, we are ready to move on to the [Integration](#) building block.

For more information on the Quality building block, search for Quality in the *DataFlux dfPower Studio Online Help*.

Integration

Data integration encompasses a variety of techniques for joining, merging/de-duplicating, and householding data from a variety of data sources. Data integration can help identify potentially duplicate records, merge dissimilar data sets, purge redundant data, and link data sets across the enterprise through:

- [Joining](#)
- [Merging/De-duplicating](#)
- [Householding](#)

Joining

Joining is the process of bringing data from multiple tables into one. For our data, we have three options for joining:

- Combine the Contacts table into the Purchase table
- Combine Purchase table into the Contacts table
- Combine both tables into a new table

We decide to combine both tables into a new table. As we start this process, we recall that the data types and lengths for similar fields differ between the two tables, and that some data types and lengths should be adjusted for increased efficiency. For example, currently:

- The Data Lengths for many of the fields vary between the two tables.
- Similar fields also have different data types between the two tables. For example, the Contacts DATE field is set to DATETIME, while the Purchase ORDER DATE field is set to VARCHAR.

- The Contacts table stores customers' addresses in a single ADDRESS field, while the Purchase table uses separate ADDRESS and ADDRESS2 fields.

We could address all these issues by opening each table using the appropriate database tool (for example, dBase or Oracle) and changing field names, lengths, and types as needed, making sure data lengths for any given field is at least as large as the actual maximum length of those fields in both tables. However, we can also change data types and lengths and join the two tables into a new table using dfPower® Architect.

To begin, start a new job flow in dfPower Architect. Click **Data Inputs > Data Source** and add the Contacts table to the flow. On the Output Fields tab for that job step, click **Override**, which lists each field and its length. To set a new length for a field, specify that length in the Override column. To set a new type for a field, specify the new type in the Override column. To set a new length and type, specify the length followed by the type in parentheses. For example, to change the field from a data length of 20 and a type of VARCHAR to a data length of 4 and a type of INTEGER, specify **4(INTEGER)**. We take similar steps for the Purchase table, adding a second Data Source job step to the flow and using the Override Defaults dialog to change data lengths and types.

Now that both tables are at the top of our job flow and similar fields have the same data lengths and types, click **Utilities > Data Union**, and then click **Add** to create one-to-one mappings between similar fields in each table and to the combined field in the new table. The Data Union job step does not create records with combined data, but rather it adds records from one table as new records to the other without making any changes to the original records. For example, we map the Contacts FIRST NAME field and Purchase FIRST NAME field to a combined Customer FIRST NAME field.

Finally, click **Data Outputs > Data Target (Insert)** to create a job step that specifies a database and new table name. Then run the job flow to create a new table that contains all the customer records for both tables.

If there was a common field in both tables (ideally primary key/foreign key fields), we could have used a Data Joining job step in dfPower Architect to combine matching records as we combined the two tables. For example, if we had one table that contained unique customer IDs and customer name information and another table that had unique customer IDs and customer phone numbers, we could use a Data Joining job step to create a new table with customer IDs, names, and phone numbers.

To illustrate this approach, we will start with these two hypothetical sets of records:

Customer ID	First Name	Last Name
100034	Robert	Smith
100035	Jane	Kennedy
100036	Karen	Hyder
100037	Moses	Jones

Customer ID	Phone
100034	(919) 484-0423
100035	(919) 479-4871

Customer ID	Phone
100036	(919) 585-2391
100037	(919) 452-8472

Using the Data Joining job step, we could identify Customer ID as the primary key/foreign key fields and create the following records:

Customer ID	First Name	Last Name	Phone
100034	Robert	Smith	(919) 484-0423
100035	Jane	Kennedy	(919) 479-4871
100036	Karen	Hyder	(919) 585-2391
100037	Moses	Jones	(919) 452-8472

Merging/De-duplicating

Now that we have all the customer records in one table, we can look for records that appear to be for the same customer and merge the data from all records for a single customer into a single surviving record.

Let us look at selected fields in three records that might be for the same customer:

	Record 1	Record 2	Record 3
First Name	Robert	Bob	Rob
Last Name	Smith	Smith	Smith
Street Address		100 Main	100 Main St
City	Carrboro		Carrboro
State	NC		
ZIP	27510	27510	
Match Code	GHWS\$\$EWT\$	GHWS\$\$EWT\$	GHWS\$\$WWI\$

Notice that each record contains a match code, which we generated in the Matching section of the Integration building block. We can use these codes as keys to find and merge duplicate records. To do this, click **Tools > Integration > Match** on the dfPower Studio main menu. Select our combined customer records table, choose **Outputs > Eliminate Duplicates**, then set a match definition of Exact for the Match Code field.



Note: Because we already generated and appended match codes to our records in the Quality building block, we only need to match on the Match Code field. If we did not already have match codes, however, we could specify match definitions and sensitivities for selected fields in Integration – Match and the application would use match codes behind the scenes to find multiple records for the same customer. We could even set OR conditions such as If FIRST NAME, LAST NAME, and ID are similar OR if FIRST NAME, LAST NAME, and ADDRESS are similar.

We then choose **Settings** from the Eliminate Duplicates output mode and specify several options for our duplicate elimination job, including **Manually Review Duplicate Records**,

Physically Delete Remaining Duplicates, and those surviving records should be written back to the Current Source Table. When the options are all set, run the job, and after processing, the **Duplicate Elimination File Editor** dialog appears, showing our first surviving records from a cluster of records that might be for the same customer. The results look something like this:

	FIRST NAME	LAST NAME	ADDRESS	CITY	STATE	PHONE
X	Robert	Smith		Carrboro	NC	GHWS\$\$EWT\$

	FIRST NAME	LAST NAME	ADDRESS	CITY	STATE	PHONE
X	Robert	Smith	100 Main St	Carrboro	NC	GHWS\$\$EWT\$
	Bob	Smith	100 Main			GHWS\$\$EWT\$
	Rob	Smith	100 Main St	Carrboro		GHWS\$\$WWI\$

Notice that the surviving record and first record in the cluster are both checked and contain the same data. The Duplicate Elimination File Editor has selected the Robert Smith record as the surviving record, probably because it is the most complete record. However, the ADDRESS for that record is still blank. To fix this, we double-click on the ADDRESS value for the Rob Smith record to copy 100 Main St from the Rob Smith record to the Robert Smith record. The results now look something like this:

	FIRST NAME	LAST NAME	ADDRESS	CITY	STATE	PHONE
X	Robert	Smith	100 Main St	Carrboro	NC	GHWS\$\$EWT\$

The ADDRESS is highlighted in red in the surviving record to indicate that we manually added the value.

To review and edit the surviving record and records cluster for the next customer, click **Next** and start the process again, repeating as necessary until duplicate records have been merged and the excess records deleted.



Note: If you find patterns of missing information in surviving records, especially if you have thousands of potentially duplicate records, consider returning to dfPower Integration – Match and setting field rules, or record rules (or both) to help the Duplicate Elimination File Editor make better choices about surviving records.


Householding

Another use for match codes is householding. Householding entails using a special match code, called a household ID or HID, to link customer records that share a physical or other relationship. Householding provides a straightforward way to use data about your customers' relationships with each other to gain insight into their spending patterns, job mobility, family moves and additions, and more.

Consider the following pair of records:

First Name	Last Name	Street Address	Phone
Joe	Mead	159 Milton St	(410) 569-7893
Julie	Swift	159 Milton St	(410) 569-7893

Given that Joe and Julie have the same address and phone number, it is highly likely that they are married or at least share some living expenses. Because the organization in our example prefers to send marketing pieces to entire households rather than just individuals, and because it is considering offering special account packages to households that have combined deposits of more than \$50,000, we need to know what customers share a household.

 **Note:** Although a household is typically thought of in the residential context, similar concepts apply to organizations. For example, householding can be used to group together members of a marketing department, even if those members work at different locations.

To create household IDs, we use dfPower Integration – Match to select our customer records table, and then set up the following match criteria as OR conditions:

- Match Code 1 (MC1): LAST NAME and ADDRESS
- Match Code 2 (MC2): ADDRESS and PHONE
- Match Code 3 (MC2): LAST NAME and PHONE (MC3)

Behind the scenes, dfPower Integration – Match generates three match codes, as shown below. If any one of the codes match across those records, the application assigns the same persistent HID to all those records.

First Name	Last Name	Street Address	Phone	MC1	MC2	MC3	HID
Joe	Mead	159 Milton St	(410) 569-7893	\$MN	#L1	%RQ	1
Julie	Swift	159 Milton St	(410) 569-7893	\$RN	#L1	%LQ	1
Michael	Becker	1530 Hidden Cove Dr	(919) 688-2856	\$BH	#H6	%B6	2
Jason	Green	1530 Hidden Cove Dr	(919) 688-2856	\$GH	#H6	%G6	2
Becker	Ruth	1530 Hidden Cove Dr	(919) 688-2856	\$RH	#H6	%R6	2
Courtney	Benson	841 B Millwood Ln	(919) 231-2611	\$BM	#M2	%B2	3
Courtney	Myers	841 B Millwood Lane	(919) 231-2611	\$MM	#M2	%M2	3
David	Jordan	4460 Hampton Ridge	(919) 278-8848	\$JH	#H2	%J2	5
Carol	Jordan	4460 Hampton Ridge	(919) 806-9920	\$JH	#H2	%J8	5
Robin	Klein	5574 Waterside Drive	(919) 562-7448	\$KW	#W5	%K5	7
Sharon	Romano	5574 Waterside Drive	(919) 562-7448	\$RW	#W5	%R5	7

First Name	Last Name	Street Address	Phone	MC1	MC2	MC3	HID
Carol	Romano	5574 Waterside Drive	(919) 239-7436	\$RW	#W2	%R2	7
Melissa	Vegas	PO Box 873	(919) 239-2600	\$VB	#B2	%V2	14
Melissa	Vegas	12808 Flanders Ln	(919) 239-2600	\$VF	#F2	%V2	14

Now that our data is integrated, including joining, merging/de-duplicating, and householding, we move on to the next building block: [Enrichment](#).

For more information about Integration, refer to the *DataFlux® dfPower Studio Online Help*.

Enrichment

In the Enrichment building block, we add to our existing data by verifying and completing incomplete fields, and adding new data such as geocodes.

For our data, we are going to perform three types of enrichment:

- [Address Verification](#)
- [Phone Validation](#)
- [Geocoding](#)

Address Verification

Address verification is the process of verifying and completing address data based on existing address data. As examples:

- You can use an existing ZIP code to determine the city and state.
- You can use an existing street address, city, and state to determine the ZIP or ZIP+4 code.
- You can use an existing ZIP code to determine that the street address actually exists in that ZIP code.

Currently, a sampling of our data looks like this:

FIRST NAME	LAST NAME	ADDRESS	CITY	STATE	PHONE
Bill	Jones	1324 New Rd		NC	716-479-4990
Sam	Smith	253 Forest Rd	Cary	NC	919-452-8253
Julie	Swift	159 Merle St		NC	716-662-5301
Mary	Wise	115 Dublin Woods Dr			614-484-0555

To verify and help complete some of these records, we first launch dfPower® Architect on the dfPower **Base** menu, use a Data Source job step to specify our combined customer records table, and then add an Address Verification job step. In the Address Verification job step, we assign the following address types to each field:

- ADDRESS: Address Line 1
- CITY: City
- STATE: State

This tells dfPower Architect what type of data to expect in each field. (It is only coincidence that the City and State field and address types use the same term. We could just as easily have had a CtyTwn field to which we would assign a City address type.)

For the output fields, we specify the following:

- Output Type: Address Line 1, Output Name: ADDRESS
- Output Type: City, Output Name: CITY
- Output Type: State, Output Name: STATE
- Output Type: ZIP/Postal Code, Output Name: ZIP
- Output Type: US County Name, Output Name: COUNTY

Notice that except for ZIP and COUNTY, all the fields get updated with new data. For ZIP and COUNTY, Architect creates a new field. Additional Output Fields:

- FIRST NAME
- LAST NAME
- PHONE

After running the job flow, our data now looks like this:

FIRST NAME	LAST NAME	ADDRESS	CITY	STATE	ZIP	COUNTY	PHONE
Bill	Jones	1324 E New Rd	Durham	NC	27712-1525	Durham	716-479-4990
Sam	Smith	253 Forest Rd	Cary	NC	27513-1627	Wake	919-452-8253
Julie	Swift	159 Merle St	Orchard Park	NY	14127-5283	Erie	716-662-5301
Mary	Wise	115 Dublin Woods Dr S	Cadiz	OH	43953-1524	Harrison	614-484-0555

Notice that the ADDRESS, CITY, STATE, and ZIP fields are now completely populated, a couple of addresses have new directional designations (for example, 1324 E New Rd), all the ZIP codes are now ZIP+4 codes, and the COUNTY field has been created and populated. Also note that Julie Swift's State was changed from NC to NY because 14127 is a ZIP code for western New York.



Note: To change these values, dfPower Architect used data from the US Postal Service. If you have licensed dfPower Verify, you should have access to this same data, as well as data for phone validation and geocoding.

Phone Validation

Phone validation is the process of checking that the phone numbers in your data are valid, working numbers. dfPower Studio accomplishes this by comparing your phone data to its own reference database of valid area code and exchange information and returning several pieces of information about that phone data, including a verified area code, phone type, and MSA/PSA codes. In addition, you can add one of the following result codes to each record:

- **FOUND FULL** – The full telephone number appears to be valid.
- **FOUND AREA CODE** – The area code appears to be valid, but the full phone number does not.
- **NOT FOUND** – Neither the area code or phone number appear to be valid.

Ignoring the address fields, our data currently looks like this:

FIRST NAME	LAST NAME	PHONE
Bill	Jones	716-479-4990
Sam	Smith	919-452-8253
Julie	Swift	716-662-5301
Mary	Wise	614-484-0555

To validate the phone numbers, we continue our dfPower Architect job flow from address verification, and add a Phone job step. In the Phone job step, we identify PHONE as the field that contains phone numbers, select PHONE TYPE, AREA CODE, and RESULT as outputs, and add FIRST NAME and LAST NAME as additional output fields.

The data now looks like this:

FIRST NAME	LAST NAME	PHONE	AREA CODE	PHONE TYPE	RESULTS
Bill	Jones	716-479-4990	716	Standard	FOUND FULL
Sam	Smith	919-452-8253			NOT FOUND
Julie	Swift	716-662-5301	716	Standard	FOUND AREA CODE
Mary	Wise	614-484-0555	740	Cell	FOUND FULL

Geocoding

Geocoding is the process of using ZIP code data to add geographical information to your records. This information will be useful because it allows us to determine where customers live in relation to each other and to the organization in question. We might use this information to plan new offices or inform new customers of their closest office. Geocode data can also indicate certain demographic features of our customers, such as average household income.

To add geocode data to our records, we add a Geocoding job step to our dfPower Architect job flow, identify our ZIP field, and indicate what kind of geocode data we want. Options include latitude, longitude, census tract, FIPS (the Federal Information Processing Standard code assigned to a given county or parish within a state), and census block.



Note: Geocode data generally needs to be used with a lookup table that maps the codes to more meaningful data.

For more information on the Enrichment building block, start with the Architect topic in the *DataFlux® dfPower Studio Online Help*.

Our data is now ready for our customer mailing. However, one data-management building block remains: [Monitoring](#).

For more information on Enrichment, search for Enrichment in the *DataFlux dfPower Studio Online Help*.

Monitoring

Because data is a fluid, dynamic, ever-evolving resource, building quality data is not a one-time activity. The integrity of data degrades over time as incorrect, nonstandard, and invalid data is introduced from various sources. For some data, such as customer records, existing data becomes incorrect as people move and change jobs.

In the fifth and final building block, Monitoring, we set up techniques and processes to help us understand when data gets out of limits and to identify ways to correct data over time. Monitoring helps ensure that once data is consistent, accurate, and reliable, we have the information we need to keep the data that way. For our data, we use two types of monitoring techniques and processes:

- [Auditing](#)
- [Alerts](#)

Auditing

Auditing involves periodic reviews of your data to help ensure you can quickly identify and correct inaccurate data. Auditing requires you to set a baseline for the acceptable characteristics of your data. For example, it might be that certain fields should:

- Never contain null values
- Contain only unique values
- Contain only values within a specified range

For our customer records, we know there are instances where the CITY field is blank, so we plan to regularly audit this field to see if the number of blank fields is increasing or decreasing over time. We will generate a trend chart that shows these changes graphically.

To prepare for auditing, we create a Profile job just as we did in the Profiling building block: launch dfPower® Profile **Configurator** component from the dfPower **Profile** menu, connect to our customer records database and table, and run the resulting Profile job. This creates a profile report that appears in dfPower Profile Viewer component and establishes our auditing baseline.



Note: For your own data, you might want to select all fields and all metrics. This gives you the greatest flexibility in auditing data on the fly.

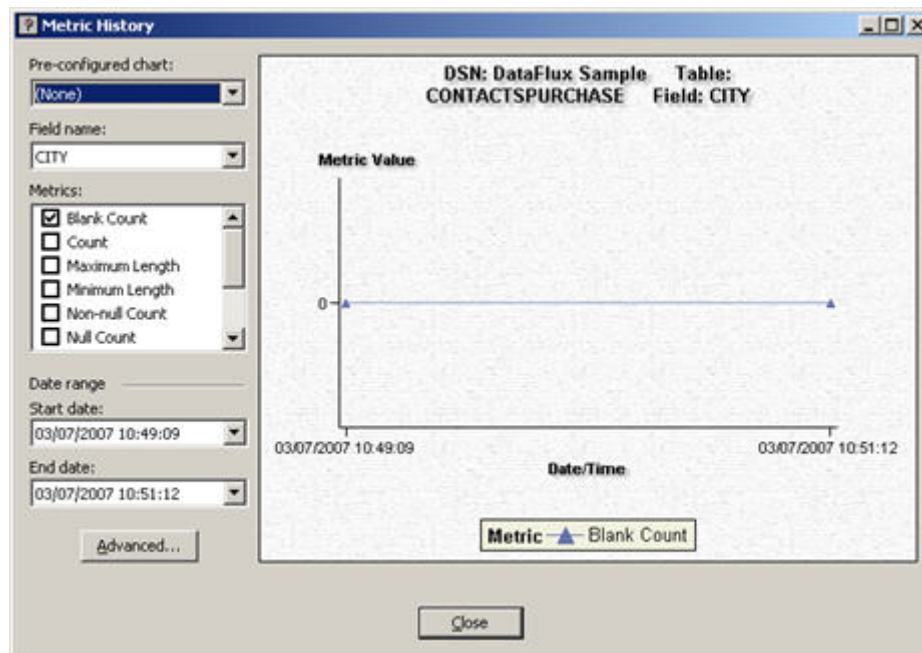
Now, each day, we open and run that same job from the Configurator, making sure to keep **Append to Report (If It Already Exists)** checked on the **Run Job** dialog.



Note: If you plan to do a lot of data auditing, consider using dfPower Studio's Base – Batch component to schedule and automatically run your Profile job or jobs.

When the dfPower Profile - Viewer main screen appears, select **Tools > Trend Analysis > Historical Analysis**. In the **Metric History** dialog, select **CITY** as the field name and **Blank Count** as the metric. If we want to audit data for a certain time period, select start and end dates and times; a date and time is available for each time we ran the Profile job.

The Metric History screen displays a chart like this:



Metric History dialog

Alerts

Alerts are messages generated by dfPower Studio when data does not meet criteria you set. For our data, we know that the LAST NAME field is sometimes left blank. To be alerted whenever this happens, we will set up an alert.

To do this, we launch dfPower Profile **Configurator** component from the dfPower **Profile** menu, choose **Tools > Trend Analysis > Alert Configuration**, and select our customer records database and table. Then, select the **LAST NAME** field and set the following:

- Metric: Blank Count
- Comparison: Metric is Greater Than
- Value: 0

This automatically generates a description of Blank Count is greater than 0.

We also indicate we want to receive alerts by email and specify the email address of a group of people responsible for addressing these alerts. Now, whenever dfPower Profile finds a blank LAST NAME field, it sends out an email message.

We can also review alerts through dfPower Profile Viewer component. To do this, in the Viewer, click **Tools > Trend Analysis > Alerts**. The **Alert** dialog appears, listing the current alerts, similar to what is shown in the following table:

Data Source Name	Table Name	Field Name	Description
Bank Records	Customer Records	LAST NAME	Blank Count is greater than 0

For more information on the Monitoring building block, see the Profile and Batch topics in *DataFlux® dfPower Online Help*.

Summary

We hope this scenario has provided you with a solid foundation for starting to use dfPower Studio through all five data-management building blocks: Profiling, Quality, Integration, Enrichment, and Monitoring.

As we mentioned at the beginning of this chapter, this scenario covers a sample of dfPower Studio. For more information on what you can do with dfPower Studio, see *DataFlux dfPower Online Help*.

Technical Support

This section addresses questions and issues related to DataFlux® dfPower Studio.

[Frequently Asked Questions](#)

[Error Messages](#)

If you do not find your answer, please contact [DataFlux Technical Support](#).

Frequently Asked Questions

Below is a list of frequently asked questions related to getting started with dfPower Studio.

Licensing

Can I access dfPower Studio using my laptop?

Licensing of dfPower Studio can be implemented with different mobile licensing options. Depending on how your network is configured, you may be able to access dfPower while off of your network using checked out or "borrowed" licenses. Typically a borrowed license specifies certain features enabled for a specified number of days.

How can I access dfPower Studio if my license server goes down?

Licensing can be set up using three redundant servers. With this architecture, only two need to be running at any given time for the license server to verify licenses. By maintaining an identical copy of the license file on each server, you avoid having a single point of failure. Contact your DataFlux representative for more information on configuring three-server redundancy.

Can I access dfPower Studio through Remote Desktop, Citrix® Metaframe®, or other Windows® Terminal Service?

Yes, your dfPower Studio license can be configured to check out a license while running in a Windows Terminal Services guest session. This is a feature that can be enabled at the time your license is created.

Why do I receive a notice that a license is about to expire when I open dfPower Navigator?

This annual licensing notification appears when one or more of your DataFlux product licenses is about to expire. DataFlux products have an annual license model to allow users to access services and run jobs. The system keeps track of the expiration dates for each feature, and a notice alerts users to the impending license expiration using the following process:

1. Sixty days prior to license expiration, a dialog begins appearing daily in dfPower Navigator. The dialog DataFlux contains a list of the licensed features set to expire, as well as the number of days left for each license. You can select **Do not display this warning again** to disable the warning after it appears at least one time.

2. When the first licensed feature reaches the expiration date, another dialog displays daily, alerting the user that one or more features have expired and these features are now operating within a thirty-day grace period. The list displays the number of days remaining for each feature (or if the license has expired and no longer accesses the product). This notice cannot be disabled.
3. After the thirty-day grace period, services or jobs requested through dfPower Navigator, but have expired, no longer run.

The server log keeps records of all notification warnings generated.

Contact your DataFlux sales representative to renew your DataFlux product license(s).

Requirements

Why is TKTS failing to read SAS® Data Sets on AIX®?

In order to access SAS Data Sets on AIX, you must have AIX 5.3 with patch level 6 installed on your system.

Error Messages

Below is a possible error message related to getting started with dfPower Studio.

Why am I getting the following error messages?

```
[time of error] (DATAFLUX) UNSUPPORTED: "DFLLDIAG2" (PORT_AT_HOST_PLUS )
phamj4@UTIL0H4GHXD1 (License server system does not support this feature. (-
18,327))
```

```
[time of error] (DATAFLUX) UNSUPPORTED: "DFLLDIAG1" (PORT_AT_HOST_PLUS )
phamj4@UTIL0H4GHXD1 (License server system does not support this feature. (-
18,327))
```

These error messages refer to the licenses for two built-in features used internally by DataFlux for debugging. These licenses are not distributed, so the license checkout request process for these two licenses fails and produces these errors. This is normal and should occur only one time, when the license checkout request is made for the first time.

Appendix

[Appendix A: dfPower Architect Configuration Directives](#)

[Appendix B: dfPower Data Access Component Directives](#)

Appendix A: dfPower Architect Configuration Directives

Following is a list of common configuration settings for your architect.cfg file. There is an additional list of Architect Configuration directives when you use DataFlux Integration Server and dfPower Studio together. Refer to the DataFlux Integration Server Online Help for these directives.

Setting	Description
BLUEFUSION/QKB	Path to Blue Fusion QKB. # Example BLUEFUSION/QKB=C:\Program Files\DataFlux\QltyKB\CI\2008A
BLUEFUSION/SURFACEALL	Many of the Standardization definitions do not default to allow parsed input. If you do not have dfPower Customize, use this configuration setting to allow parsed input. If set to true, parsed input is allowed. # Example BLUEFUSION/SURFACEALL=true
CLUSTER/BYTES	Bytes of memory to use for Clustering/Cluster Update step. # Example # 1024 = 1 Kilobyte, 41943040 = 40 Megabytes CLUSTER/BYTES=67108864
CLUSTER/LOG	Clustering/Cluster Update log file generation. 0 means no log file will be created, 1 means a log file will be created. # Example CLUSTER/LOG=0
DFCLIENT/CFG	dfIntelliServer Client Config Path. # Example DFCLIENT/CFG=C:\Program Files\DataFlux\dfIntelliServer\etc\dfclient.cfg

Setting	Description
EMAILCMD	<p>Email command to use for Monitor email alerts. Please enter your SMTP server name.</p> <pre># Example EMAILCMD=cscript -nologo "C:\Program Files\DataFlux\dfPower Studio\8.2\bin\mail.vbs" -s smtp_server_name_here "%T" < "%B"</pre>
FRED/LOG	<p>Set it to 1 or 0 to control logging. By default logging is turned off.</p> <pre># Example FRED/LOG</pre>
MONITOR/REPOSFIL	<p>Repository configuration file.</p> <pre># Example MONITOR/REPOSFIL=</pre>
SORTBYTES	<p>Bytes of memory to use while sorting, joining, etc.</p> <pre># Example # 1024 = 1 Kilobyte, 41943040 = 40 Megabytes SORTBYTES=32000000</pre>
TEMP	<p>Path where temporary files are stored.</p> <pre># Example TEMP=C:\Program Files\DataFlux\dfPower Studio\8.2\temp\</pre>
VERIFY/BF	<p>Path to internal Verify data libraries.</p> <pre># Example VERIFY/BF=C:\Program Files\DataFlux\dfPower Studio\8.2\datilib\</pre>
VERIFY/CACHESIZE	<p>A percentage (0 to 100) which tells Verify how much data to cache in memory. 100 would take up more memory, but would be faster.</p> <pre># Example VERIFY/CACHESIZE=20</pre>
VERIFY/CANADA	<p>Path to Canada verify data.</p> <pre># Example VERIFY/CANADA=C:\Program Files\DataFlux\dfPower Studio\8.2\mgmtrsrc\RefSrc\SERPData</pre>
VERIFY/GEO	<p>Path to geocoding data.</p> <pre># Example VERIFY/GEO=C:\Program Files\DataFlux\GeoPhoneData</pre>

Setting	Description
VERIFY/PRELOAD	<p>A string representing which US states to preload into memory to make Verify run faster. Enter ALL for all data.</p> <p># Example VERIFY/PRELOAD=</p>
VERIFY/USERDI	<p>Enable/disable RDI processing. 1 means enable, 0 means disable.</p> <p># Example VERIFY/USERDI=0</p>
VERIFY/USEDPV	<p>Enable/disable DPV processing. 1 means enable, 0 means disable.</p> <p># Example VERIFY/USEDPV=0</p>
VERIFY/USPS	<p>Paths to USPS data.</p> <p># Example VERIFY/USPS=C:\Program Files\DataFlux\dfPower Studio\8.2\mgmtrsrc\refsrc\USPSData\</p>
VERIFY/USPSINST	USPS data installed flag.
VERIFYINTL/CFG	<p>Config file for Verify International.</p> <p># Example VERIFYINTL/CFG=</p>
VERIFYWORLD/DB	<p>Path to where Verify-World data is stored.</p> <p># Example VERIFYWORLD/DB=C:\Program Files\DataFlux\dfPower Studio\8.2\mgmtrsrc\refsrc\</p>
VERIFYWORLD/UNLK	<p>Unlock code for Verify-World step (must be enclosed in single quotes).</p> <p># Example VERIFYWORLD/UNLK= ' '</p>

Appendix B: dfPower Data Access Component Directives

The following table lists the settings in the app.cfg file, which are used by the Data Access Component (DAC) to determine the operation it will perform.

Setting	Description
Command file execution	<p>Specifies a text file with SQL commands (one per line). These commands run in turn, on any new connection. For example, they can be used to set session settings. This is only implemented for the ODBC driver.</p> <p>The USER\savedconnectiondir configuration value may specify the path to the saved connections. The DAC checks for files with the same filename as the DSN and a .sql extension.</p>
DAC Logging	<p>Specifies whether or not to create a log file for DAC operations. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - Checks the USER\logfile configuration value and then the DAC checks SYSTEM\logfile for a string representing a log file name.</p> <p>UNIX - Checks the sql_log.txt file in the current working directory.</p>
DFTK log file	<p>Specifies the log file that interacts with the DFTKSRV layer and is only useful for debugging issues specific to dftksrv. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows- Checks the USER\dftklogfile value and the SYSTEM\dftklogfile value. The \$DFTKLOGFILE value.</p> <p>UNIX - Checks the \$DFTKLOGFILE value.</p>
Disable CEDA	<p>Specifies whether to disable CEDA. This setting is only applicable to TKTS connections. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - Checks the USER\dftkdisableceda configuration value, which should specify any non-null value, for example, yes. The SYSTEM\dftkdisableceda value. The \$DFTKDISABLECEDA value.</p> <p>UNIX - Checks the \$DFTKDISABLECEDA value.</p>

Setting	Description
Oracle NUMBER(38) handling	<p>Specifies whether to treat NUMBER (38) columns as an INTEGER (which is the default) or a REAL value. This setting applies if Oracle is the only driver to which you are connecting. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - Checks the USER\[dsn_name]\oranum38real has a double word value of 1. Next, the DAC checks that SYSTEM\[dsn_name]\oranum38real has a double word value of 1.</p> <p>UNIX - Checks the \$HOME/.dfpower/dsn.cfg file for [dsn_name] = oranum38real.</p>
Suffix for CREATE TABLE statements	<p>Specifies a string that is appended to every CREATE TABLE statement. If you include %t in this string, it is substituted with the table name. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - Checks the USER\[dsn_name]\postcreate specified string. Next, the DAC checks that SYSTEM\[dsn_name]\postcreate specifies a string.</p> <p>UNIX - This setting is not supported.</p>
System saved connection	<p>Specifies where to find system saved connections. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - Checks the DAC/SAVEDCONNSYSTEM configuration value. Next, the DAC checks the DFEXEC_HOME environment variable, in the \$DFEXEC_HOME\etc\dsn directory.</p> <p>UNIX - Checks the \$DFEXEC_HOME/etc/dsn directory.</p>
Table type filter	<p>Limits the list of tables to several preset types. The default is 'TABLE','VIEW','ALIAS','SYNONYM'. If you set this value to * (asterisk) the list will not be filtered. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - The USER\[dsn_name]\tablelistfilter specifies a comma delimited string that lists single quoted values that indicate table types. Next, the DAC checks whether SYSTEM\[dsn_name]\tablelistfilter specifies a comma delimited string.</p> <p>UNIX - This setting is not supported.</p>

Setting	Description
TK Path	<p>Specifies where TK files are located. The dfktsrv path and core directory should be specified. \$DFTKPATH may specify the TK path. If it does not, the DAC checks the following values and locations, based on your operating system:</p> <p>Windows - The USER\tkpath value. The SYSTEM\tkpath value. The \$DFEXEC_HOME\bin;\$DFEXEC_HOME\bin\core\sasext location.</p> <p>UNIX - Check \$TKPATH. Next, check \$DFEXEC_HOME/lib/tkts.</p>
TKTS DSN directory	<p>Specifies the path where TKTS DSNs are stored in XML files. \$DFTKDSN may specify the path to the TKTS DSN directory. If it does not specify the value, the DAC checks the following values and locations, based on your operating system:</p> <p>Windows - The \$DFEXEC_HOME\etc\dfstkdsn\ directory.</p> <p>UNIX - The \$DFEXEC_HOME/etc/dfstkdsn\ directory.</p>
TKTS log file	<p>Specifies the log file that is produced by the TKTS layer and is useful for debugging TKTS issues. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - The USER\tktslogfile configuration value. The SYSTEM\tktslogfile value. The \$TKTSLOGFILE value.</p> <p>UNIX - The \$TKTSLOGFILE value.</p>
TKTS startup sleep	<p>Specifies how much time, in seconds, to delay between the start of the dfktsrv program and the booting of TK. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - The USER\tktssleep configuration value. Next, the DAC checks the SYSTEM\tktssleep value.</p> <p>UNIX - This setting is not supported.</p>
Use braces	<p>Specifies whether to enclose DSN items with braces when they contain reserved characters. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - The USER\[dsn_name]\usebraces has a double word value of 1, where [dsn_name] is the name of the DSN. Next, the DAC will check the SYSTEM\[dsn_name]\usebraces value.</p> <p>UNIX - The \$HOME/.dfpower/dsn.cfg file for [dsn_name] = usebraces.</p>

Setting	Description
User saved connection	<p>Specifies where to find user saved connections. The DAC checks the following values and locations, based on your operating system:</p> <p>Windows - The USER\savedconnectiondir configuration value. Next, the DAC checks the application settings directory for the user, which is usually in the \Documents and Settings directory, in the DataFlux\dac\[version] subdirectory.</p> <p>UNIX - The \$HOME/.dfpower/dsn directory.</p>

Glossary

C

case definition

A set of logic used to accurately change the case of an input value, accounting for unique values that need to be cased sensitively, such as abbreviations and business names.

chop tables

A proprietary file type used by DataFlux as a lex table to separate characters in a subject value into more usable segments.

D

data type

The semantic nature of a data string. The data type provides a context determining which set of logic is applied to a data string when performing data cleansing operations. Example data types are: Name, Address, and Phone Number.

definition

Another name for an algorithm. The definitions in the Quality Knowledge Base are the data management processes that are available for use in other SAS or DataFlux applications like dfPower Studio or SAS Data Quality Server.

G

grammars

Parsing rule libraries used for complex parsing routines.

I

identification definition

A set of logic used to identify an input string as a member of a redefined or user-defined value group or category.

L

locale guessing

A process that attempts to identify the country of origin of a particular piece of data based on an address, a country code, or some other field.

M

match code

The end result of passing data through a match definition. A normalized, encrypted string that represents portions of a data string that are considered to be significant with regard to the semantic identity of the data. Two data strings are said to "match" if the same match code is generated for each of them.

match definition

A set of logic used to generate a match code for a data string of a specific data type.

match value

A string representing the value of a single token after match processing.

P

parse

The process of dividing a data string into a set of token values. For example: Mr. Bob Brauer [Mr. = Prefix , Bob = Given Name, Brauer = Family Name]

parse definition

A name for a context-specific parsing algorithm. A parse definition determines the names and contents of the sub-strings that will hold the results of a parse operation.

pattern analysis definition

A regular expression library that forms the basis of a pattern recognition algorithm.

phonetics

An algorithm applied to a data string to reduce it to a value that will match other data strings with similar pronunciations.

Q

QKB

The Quality Knowledge Base (QKB) is the collection of files and configuration settings that contain all DataFlux data management algorithms. The QKB is directly editable using dfPower Studio.

R

regular expression

A mini-language composed of symbols and operators that enables you to express how a computer application should search for a specified pattern in text. A pattern may then be replaced with another pattern, also described using the regular expression language.

S

sensitivity

Regarding matching procedures, sensitivity refers to the relative tightness or looseness of the expected match results. A higher sensitivity indicates you want the values in your match results to be very similar to each other. A lower sensitivity setting indicates that you would like the match results to be "fuzzier" in nature.

standardization definition

A set of logic used to standardize a string.

standardization scheme

A collection of transformation rules that typically apply to one subject area such as company name standardization or province code standardization.

T

tokens

The output strings of a parse process. These are words or atomic groups of words with semantic meaning in a data string. A set of expected tokens is defined for each data type.

V

vocabularies

A proprietary file type used for categorizing data look-ups pertinent to a specific subject area.