



Mainframe Data Integration™ (MDI) Installation and User's Guide

Legal Notices

© 2020 Luminex Software, Inc. All rights reserved. No part of this work may be reproduced or disclosed to third parties in any form or by any means, graphically, mechanically or electronically, including but not limited to photocopying, recording or taping without the prior written permission of Luminex Software, Inc.

Luminex Mainframe Data Integration (“MDI”) products are sold and licensed by Luminex Software, Inc. (“Luminex”) pursuant to the terms and conditions of Luminex’s standard Purchase and License Agreement or other applicable agreement (“Luminex Agreement”). Consult your Luminex Agreement for the specific terms and conditions governing the sale and license of your MDI. All software is covered by copyright, trade secret and/or patent protection and is owned by Luminex. No part may be copied, modified or transferred without the prior written permission of Luminex. You may not reverse engineer the hardware or reverse compile the software. Hardware and software described herein is confidential, proprietary and trade secret property of Luminex and may not be disclosed to third parties or used except as permitted in your Luminex Agreement.

Luminex, Luminex MDI, MDI SecureTransfer, MDI SAS Language Processor, MDI BigData Transfer, MDI Cross-Platform Data Sharing, MDI zKconnect, Luminex Channel Gateway, Luminex CGX, MDI File Watcher and MDI PDQ are trademarks of Luminex Software, Inc. All other trademarks and trade names are the property of others.

DISCLAIMER: While every effort has been made to ensure the technical accuracy of this manual, Luminex makes no warranties or representations, either express or implied, with respect to the contents herein. Further, should the procedures set forth in this manual not be followed or be misapplied, Luminex disclaims any warranty for damage to the product or any other liability.

MDI Installation and User Guide

File Watcher Software release 1007

PDQ Software version 2 release 1

Document version 1.12

Luminex Software, Inc.
871 Marlborough Avenue
Suite 100
Riverside, CA 92507

Phone: +1 (951) 781-4100

Fax: +1 (951) 781-4105

info@luminex.com
www.luminex.com

August 2020

Table of Contents

1. Introduction	1
1.1 Introduction to Mainframe Data Integration.....	1
1.2 Mainframe Data Integration Solutions.....	1
1.2.1 MDI SecureTransfer	1
1.2.2 MDI Cross-Platform Data Sharing.....	2
1.2.3 MDI BigData Transfer.....	2
1.2.4 MDI SAS Language Processor.....	2
1.3 Benefits	2
2. Architecture	5
2.1 General Overview of Mainframe Data Integration	5
2.2 Terms	6
3. Components.....	8
3.1 Mainframe Software/Batch Interface.....	8
3.2 Profiles	8
3.3 MDI Platform.....	9
3.4 Storage	9
3.5 Virtual Tape Volumes	9
4. Planning for MDI	10
4.1 Sizing Inputs	10
4.2 SMF Analysis for Sizing Overview	11
4.3 Planning for MDI Installation Overview	12
5. MDI Profile Creation and the Installation Workbook.....	13
6. MDI Mainframe Software Installation and Configuration	14
6.1 Installation Checklist	14
6.2 Sample LUMXPROC Procedure (PROC).....	17
6.3 Sample LUMXPDQ PROC	17
7. Mainframe Security Setup.....	19
7.1 MDI Security System Definitions	19
7.2 Security Definitions	20

7.3	Security Examples for MDI SecureTransfer Profile Definition Scenarios	21
7.4	Security Examples for MDI:XPDS Profile Definition Scenarios	22
7.5	Security Examples for MDI:SLP Profile Definition Scenarios.....	23
7.6	Security Example for MDI BigData Transfer Profile Definition Scenario	24
7.7	PDQ Security Definitions	24
7.7.1	Started Task	25
7.7.2	USERID and TOKEN	25
7.7.3	Data Set Name.....	27
7.7.4	Unprotected Data Set Names	27
8.	MDI File Watcher and Pending Data Queue (PDQ)	28
8.1	Introduction.....	28
8.2	MDI File Watcher/PDQ Concepts	28
8.3	Prerequisites.....	29
8.4	Planning and Installation.....	29
8.5	Data Conversions.....	37
8.6	DCB Exit.....	38
8.7	DCB Table.....	39
8.7.1	Email Notification Settings	39
8.7.2	DCB Parameters	39
8.8	PDQ Parameters.....	40
8.8.1	DDCMDPRINT=Y	41
8.8.2	DEV=devnum1[-devnum2] STORCLAS=mtl-storclass	41
8.8.3	DSNPREF=cgx-communications-dsnprefix	41
8.9	PDQ Operator Commands	41
8.9.1	PSTOP	42
8.9.2	DCXCODE=datasetname(member).....	42
8.9.3	DISABLE=devnum1[-devnum2]	42
8.9.4	ENABLE=devnum1[-devnum2]	42
8.9.5	LCG	42
8.9.6	LDEV	43
8.9.7	LREQ.....	43
8.10	PDQ Messages.....	44

8.11	Audit Logs	44
8.11.1	Additional Error Information	45
8.11.2	RetryXXX Messages	45
8.11.3	File Disposition	45
9.	MDI Data Conversions	46
9.1	Introduction.....	46
9.2	Data Formats Supported Overview.....	46
9.3	Conversion Types.....	46
9.4	iconv Library.....	46
9.4.1	Tested Conversions.....	47
9.5	Trailing Space Handling	47
9.5.1	EBCDIC to ASCII Handling	47
9.5.2	ASCII to EBCDIC Handling.....	47
9.6	Conversions Moving Data from the Mainframe.....	47
9.6.1	Native (or Binary)	48
9.6.2	strip bdw	48
9.6.3	strip all.....	48
9.6.4	ASCII.....	48
9.6.5	ascii LF	48
9.6.6	ascii CRLF.....	48
9.6.7	iconv	48
9.6.8	iconv LF	48
9.6.9	iconv CRLF	48
9.6.10	custom	49
9.6.11	custom LF.....	49
9.6.12	custom CRLF	49
9.7	Conversions Moving Data to the Mainframe.....	49
9.7.1	Native (or Binary)	49
9.7.2	EBCDIC	49
9.7.3	ebcdic NPC.....	49
9.7.4	iconv	49
9.7.5	custom	49

9.8	Custom Translation Tables	50
9.9	JCL Parameter Examples	50
10.	Syntax Rules for Parameters and KEYWORD=VALUE Pairs	52
10.1	The -PARM Parameter	52
10.2	KEYWORD=VALUE Pairs	52
10.2.1	Values	52
10.3	The -DD_ <dd name> Parameter	53
10.4	Special Handling of the CONVERSION Keyword	53
10.5	Keyword Substitution for File Names	54
10.5.1	Keyword #MDY	54
10.5.2	Keyword #YDM	54
10.5.3	Keyword #DSN	55
10.5.4	Examples	55
10.6	Additional Syntax Rules	55
10.6.1	Multi-Word Values	55
10.6.2	Lines Extending Past Column 71	56
10.7	Examples	56
11.	MDI SecureTransfer (MDI:ST)	57
11.1	Introduction	57
11.2	SecureTransfer Transfer Methods	57
11.2.1	Transferring Data from the Mainframe – PUT	57
11.2.2	Transferring Data from a Non-Mainframe Platform – GET	57
11.2.3	Transferring Data from a Non-Mainframe Platform – Push Method	57
11.3	SecureTransfer Components	58
11.3.1	Mainframe	58
11.3.2	MDI Platform	58
11.3.3	Storage	58
11.3.4	MDI Reporting Interface	58
11.4	SecureTransfer Planning Overview	59
11.5	Profile Configuration	59
11.5.1	SecureTransfer JCL	60
11.6	JCL Parameters and KEYWORD=VALUE Pairs	60

11.6.1	The -PARM Parameter	61
11.6.2	-DD_<DD name> Parameter	61
11.7	SecureTransfer Parameters	64
11.7.1	destination	64
11.7.2	login	65
11.7.3	password	65
11.7.4	cipher	66
11.7.5	maxretries	66
11.7.6	delay_time	67
11.7.7	sshopts	67
11.7.8	logdir	67
11.7.9	preaction/postaction	67
11.7.10	email_error_list	68
11.7.11	email_list	68
11.7.12	emailonerror	68
11.7.13	emailall	68
11.7.14	Handling Emails	69
11.7.15	oformat Keyword	69
11.8	Sample JCL – PUT	70
11.8.1	Dataset Transfer to Virtual Tape	71
11.8.2	LUMXPROC	71
11.8.3	PROFILE	71
11.8.4	DD Name	71
11.8.5	SYSIN	71
11.9	Sample JCL – GET	71
11.9.1	Virtual Tape Dataset	72
11.9.2	LUMXPROC	72
11.9.3	PROFILE	72
11.9.4	DD Name	72
11.9.5	SYSIN	72
11.10	Executing a Test Using SecureTransfer	73
11.11	Results	73

11.12 Migration/JCL Conversion/Professional Services	73
11.12.1 Scratch Tape Management.....	73
11.13 SecureTransfer Security Requirements.....	73
11.14 SFTP Host Requirements.....	74
11.15 Licensing.....	74
12. Cross-Platform Data Sharing (MDI:XPDS)	75
12.1 Introduction.....	75
12.2 File System Mount Point	76
12.3 Cross-Platform Data Sharing Components	76
12.3.1 Mainframe	76
12.3.2 MDI Platform	76
12.3.3 Storage.....	77
12.3.4 MDI Reporting Interface	77
12.4 XPDS Planning Overview	77
12.5 XPDS Profile Configuration	78
12.5.1 XPDS JCL	78
12.6 XPDS JCL Parameters and KEYWORD=VALUE Pairs.....	79
12.6.1 Operational Arguments.....	79
12.6.2 JCL Parameters and Arguments	79
12.6.3 -DD_<DD name> Parameter.....	80
12.6.4 XPDS Parameters	83
12.7 File Path Conventions.....	85
12.8 Sample JCL – PUT.....	87
12.8.1 Dataset Transfer to Virtual Tape.....	88
12.8.2 LUMXPROC.....	88
12.8.3 PROFILE.....	88
12.8.4 DD Name.....	88
12.8.5 SYSIN	88
12.9 Sample JCL – GET	88
12.9.1 Virtual Tape Dataset	89
12.9.2 LUMXPROC.....	89
12.9.3 PROFILE.....	89

12.9.4 DD Name.....	89
12.9.5 SYSIN	89
12.10 Executing a Test Using XPDS	89
12.11 Job Results	90
12.12 Migration/JCL Conversion/Professional Services	90
12.13 Scratch Tape Management	90
12.14 RACF Profile Security	90
12.15 MDI Default User Requirements	90
12.16 Profile Licensing	91
13. MDI BigData Transfer (MDI:BDT).....	92
13.1 Introduction.....	92
13.1.1 BigData Transfer Components	92
13.1.2 BigData Transfer Planning Overview	93
13.1.3 Profile Configuration	94
13.2 Configure Azure for Use with BigData Transfer	94
13.2.1 Create an Active Directory Web Application	95
13.2.2 Finding Your Application ID	96
13.2.3 Generating and Finding Your Authentication Key	96
13.2.4 Find Tenant ID.....	97
13.2.5 Assign Application to Role.....	97
13.2.6 Configuring a Firewall.....	97
13.3 Data Conversion.....	98
13.4 BigData Transfer JCL	98
13.5 BigData Transfer JCL Parameters and KEYWORD=VALUE Pairs	99
13.5.1 The -PARM Parameter	99
13.5.2 -DD_<DD name> Parameter.....	99
13.5.3 hdfs_azure_application_id.....	100
13.5.4 hdfs_azure_application_key	100
13.5.5 hdfs_azure_tenant_id.....	101
13.5.6 hdfs_host	101
13.5.7 hdfs_protocol.....	101
13.5.8 hdfs_port.....	101

13.5.9	hdfs_path	102
13.5.10	hdfs_directory	102
13.5.11	hdfs_java	102
13.5.12	login	102
13.5.13	bucket	102
13.5.14	compression	103
13.5.15	Passing the credentials file using a JCL DD statement	103
13.5.16	email_error_list	104
13.5.17	email_list	104
13.5.18	emailonerror	104
13.5.19	emailall	104
13.5.20	Handling Emails	104
13.6	JCL Examples	105
13.6.1	Direct Protocol JCL Example	105
13.6.2	WebHDFS Protocol JCL Example	106
13.6.3	Azure JCL Example	107
13.7	Executing a Test Using BigData Transfer	109
13.8	Results	109
13.9	Migration/JCL Conversion/Professional Services	109
13.10	Scratch Tape Management	109
13.11	BigData Transfer Security Requirements	109
13.12	Licensing	110
14.	MDI SAS Language Processor (MDI:SLP)	111
14.1	Introduction	111
14.2	SAS Language Processor Components	112
14.2.1	Mainframe	112
14.2.2	MDI Platform	112
14.2.3	Optional SAS Language Server	112
14.2.4	Storage	112
14.2.5	MDI Reporting Interface	112
14.3	Planning for SLP	113
14.3.1	SLP Planning Overview	113

14.3.2	Discovery Process Phase I.....	114
14.3.3	Discovery Process Phase II	115
14.3.4	Sizing for Virtual Tape VOLSERS and Tape Drives	115
14.3.5	Naming Conventions for Directories and Dataset Naming Standards	117
14.3.6	Remote Execution of the SAS Language Compiler/Interpreter	117
14.3.7	Moving Partitioned Data Set (PDS) to the MDI:SLP Platform.....	118
14.3.8	Profile Configuration	118
14.3.9	Data Conversion from the Mainframe	118
14.4	SLP JCL	119
14.4.1	Referencing SAS Language Programs in the JCL.....	119
14.5	SLP JCL Parameters and KEYWORD=VALUE Pairs	120
14.5.1	Parameters Set in the SLP Profile on the MDI Platform	120
14.5.2	The -PARM Parameter	120
14.5.3	Keywords & Values.....	121
14.5.4	Email Setup	125
14.5.5	Sample JCL for the MXG Use Case	126
14.5.6	Data Migration Examples JCL and PROCs	130
14.6	Important Locations on the SLP Platform	132
14.7	Executing a Test using SLP.....	132
14.8	Results	133
14.9	SLP Requirements.....	133
14.10	Licensing.....	133
14.11	References.....	133
15	MDI zKonnnect	134
15.1	Introduction.....	134
15.2	Components	134
15.3	Planning	135
15.4	Collect Information on the Kafka Topic(s)	136
15.5	Collect Information on the Number of Partitions per Topic	136
15.6	Collect Information on the IP address and Port Number for Zookeeper	137
15.7	Collect Information on the IP addresses and Port Numbers for the Kafka Brokers	137
15.8	Collect Information on Authentication Requirements	137

15.9	Planning for Virtual Tape Usage	137
15.9.1	Number of Virtual Tapes and Drives to Define	138
15.9.2	Device Type.....	138
15.9.3	Device Esoteric.....	138
15.10	Profile Configuration.....	138
15.11	zKonnnect JCL.....	139
15.12	zKonnnect JCL Parameters and KEYWORD=VALUE Pairs	139
15.12.1	The -PARM Parameter	139
15.12.2	The -DD_<DD name> Parameter.....	139
15.12.3	Keywords & Values.....	140
15.12.4	Values	140
15.12.5	Data Conversion	140
15.12.6	Comments.....	140
15.12.7	More Syntax Rules	141
15.12.8	zKonnnect Supported Keywords.....	141
15.12.9	zKonnnect JCL Example.....	142
15.13	Executing a Test Using zKonnnect.....	143
15.13.1	Results	143
15.14	Migration/Conversion/Professional Services.....	144
15.15	Scratch Management.....	144
15.16	Security Requirements	144
15.16.1	Security Example for zKonnnect Definition	144
15.17	Licensing.....	145
Appendix A.	Message Codes.....	146
Appendix B.	Security Specifications	150
Appendix C.	LSCRUP Scratch List Update Utility	154
Appendix D.	Program Characteristics and Requirements	155
Appendix E.	Restore .XMT File Using TSO Receive	157
Appendix F.	PDQ - DCB Exit Sample: DCXTYPE.....	158

This page left intentionally blank

1. Introduction

1.1 Introduction to Mainframe Data Integration

Mainframe Data Integration (MDI) is an extensible Platform that leverages the mainframe FICON channel to securely share and transfer data between mainframes and distributed systems environments.

The MDI Platform is built on Luminex's Channel Gateway (CGX), a modular, highly customizable architecture that enables limitless implementations for data integration and secure movement of mainframe data. The Platform appears as a virtual tape subsystem to the z/OS operating system on the mainframe. MDI uses this virtual tape interface to transfer data from and to the mainframe, via the FICON channel, which results in a secure and efficient transfer of data using very little system overhead.

MDI includes a mainframe batch interface used to facilitate the transfer of data between the mainframe and the MDI Platform. The batch interface is used to issue instructions to the software resident on the MDI Platform regarding actions to take with the data. MDI software can perform a variety of actions such as:

- Transferring data from the mainframe to an internal or external SFTP server
- Transferring data from an Open Systems Platform to the mainframe
- Share mainframe data with Open Systems applications, computing grids, and computing appliances; return processed data back to the mainframe or distribute from these Platforms
- Transfer data from the mainframe to Hadoop, Azure, Cloud Object Storage and so forth
- Transfer data and SAS programs to the MDI Platform for processing and return the results back to the mainframe

The FICON channel is used by the mainframe to attach to both DASD and virtual tape devices. By leveraging the FICON channel on the mainframe, MDI provides a highly secure, efficient (in terms of CPU usage) and fast (moving data at up to 800 MB/s via multiple data streams) data transport mechanism for sharing mainframe data. FICON can be deployed in a redundant fashion for high availability and is highly secure as it is connected directly from the mainframe to the supported device.

1.2 Mainframe Data Integration Solutions

1.2.1 MDI SecureTransfer

MDI SecureTransfer (MDI:ST) is a Managed File Transfer (MFT) solution used to securely move data to and from the mainframe. MDI:ST uses FICON in place of TCP/IP to securely move mainframe data to the MDI Platform where it is sent using SFTP to the destination IP address, server name or DNS name. MDI:ST includes software components on the mainframe and on the MDI Platform.

1.2.2 MDI Cross-Platform Data Sharing

MDI Cross-Platform Data Sharing (MDI:XPDS) is a data sharing solution that uses the FICON channel to share data with distributed Platforms. MDI:XPDS can transfer mainframe data to and from the mainframe. MDI:XPDS places data in a specified directory on a Network File System that is mounted to the MDI Platform. Distributed applications can access the data in the directory, process the data and place the data in a specified directory to be transferred to the mainframe as a cataloged tape data set.

1.2.3 MDI BigData Transfer

MDI BigData Transfer (MDI:BDT) transfers mainframe data to and from the Hadoop Distributed File System (HDFS). The MDI:BDT software on the mainframe and on the MDI Platform work together to transfer data from the mainframe, over secure FICON channels, to the MDI Platform where either the HDFS script provided by Apache Hadoop or a webHDFS REST API ingest the data into the HDFS.

1.2.4 MDI SAS Language Processor

MDI SAS Language Processor (MDI:SLP) executes programs written in the SAS Language on the MDI Platform. Input files and SAS programs are transferred from the mainframe over FICON to the MDI Platform, executed, and the resulting output is distributed locally or transferred to the mainframe. MDI:SLP supports the SAS Institute compiler.

1.3 Benefits

MDI Secure Transfer (MDI:ST) Benefits:

- Uses the secure FICON channel to transfer data instead of TCP/IP on the mainframe
- Clients may shut down TCP/IP ports that introduce risk, such as Port 21
- Transfers data faster off the mainframe than TCP/IP by utilizing the FICON I/O channel
- Consumes fewer system resources than FTP, SFTP and FTPS including 3rd party software products that utilize TCP/IP protocols for transferring data
- Utilizes a well-known and simple to use utility, ICEGENER (or similar), to transfer data
- Does not require x.509 digital certificates to encrypt/decrypt data
- Provides a GUI interface for audit reporting and performance data

MDI Cross-Platform Data Sharing (MDI:XPDS) Benefits:

- Uses the secure FICON channel to transfer data instead of TCP/IP on the mainframe
- Data can be securely moved to and from the mainframe to NFS storage to be processed by distributed applications and computing grids
- Data processed by distributed applications and computing grids can be transferred to the mainframe and cataloged for downstream processing
- Transfers data faster off the mainframe than TCP/IP by utilizing the FICON I/O channel
- Consumes fewer system resources than FTP, SFTP and FTPS including 3rd party software products that utilize TCP/IP protocols for transferring data
- Utilizes a well-known and simple to use utility, ICEGENER (or similar), to transfer data
- Does not require x.509 digital certificates to encrypt/decrypt data
- Provides a GUI interface for audit reporting and performance data

MDI BigData Transfer Benefits:

- Uses the secure FICON channel to transfer data instead of TCP/IP on the mainframe
- Built-in HDFS connectors invokes scripts provided by Apache Hadoop or the webHDFS REST API to ingest data
- Transfers data faster off the mainframe than TCP/IP by utilizing the FICON I/O channel
- Consumes fewer system resources than FTP, SFTP and FTPS including 3rd party software products that utilize TCP/IP protocols for transferring data
- Allows clients to move large files from the mainframe into HDFS clusters
- Utilizes a well-known and simple to use utility, ICEGENER (or similar), to transfer data
- Provides a GUI interface for audit reporting and performance data

MDI SAS Language Processor Benefits:

- Supports the execution of SAS Language programs off the mainframe using the SAS Institute compiler
- Off-hosting SAS executions saves valuable processing cycles (MSUs) on the mainframe
- Uses the secure FICON channel to transfer data instead of TCP/IP on the mainframe
- With proper sizing, performance is comparable to or faster than the mainframe
- Utilizes a well-known and simple to use utility, ICEGENER (or similar), to transfer data

Reduction of Mainframe Overhead Costs

All MDI Solutions perform the majority of the workload off the mainframe on the MDI Platform. With all MDI Solutions, the mainframe is subject to the CPU overhead of the copy utility only, which is minimal. Off-hosting CPU overhead helps to reduce the workload on the mainframe. In some cases, such as with MDI SAS Language Processor, the amount of CPU overhead can be significant enough for the client to benefit in a number of ways:

- Reduce the 4-hour rolling average to avoid capping and possibly reduce peaks for sub-capacity software license discounts
- Schedule other workloads
- Reduce overall MSU usage and postpone processor upgrades

Reduction in Licensing Costs

Some MDI Solutions can replace other licensed software on your mainframe. As an example, MDI SecureTransfer can replace competing Managed File Transfer solutions currently in use. The SAS Language Processor can help to reduce the licensing cost of the SAS compiler by moving the license to a less expensive Linux server. Of course, reduction in licensing cost is dependent upon each client's workload requirements and current license agreements with their vendors.

Security

All Luminex MDI Solutions use the FICON channel to move data to and from the mainframe instead of the network. This affords Luminex clients the opportunity to make their mainframe more secure. Data breaches are on the rise and the most popular point of entry into any company's network is FTP Port 21. Currently, Port 21 is listed in the Consensus Intrusion Database (www.sans.org) as one of the "Top Ten Ports" targeted on computer systems. It is responsible for data

breaches all over the world including large US corporations resulting in millions of dollars in various damages including loss of business, reissuing credit cards, settling law suits and more. Mark Wilson, Technical Director of RSM Partners states “If you replace mainframe FTP with a channel/FICON based solution, you can mitigate most FTP security issues a great deal, if not remove them completely. This is the real benefit of a solution such as MDI SecureTransfer.”

In addition to using secure FICON to move data to and from the mainframe, the MDI solutions use mainframe security solutions (RACF, ACF2 and Top Secret) to control access to the use of profiles defined on the MDI Platform. The security mechanism, PassTicket, is deployed for files coming from Open Systems to the mainframe to ensure that the user ID sending data to the mainframe has the proper credentials in place to do so. For more information about the PassTicket, the following link describes the PassTicket as deployed by the IBM Security solution, RACF: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.icha700/pass.htm

2. Architecture

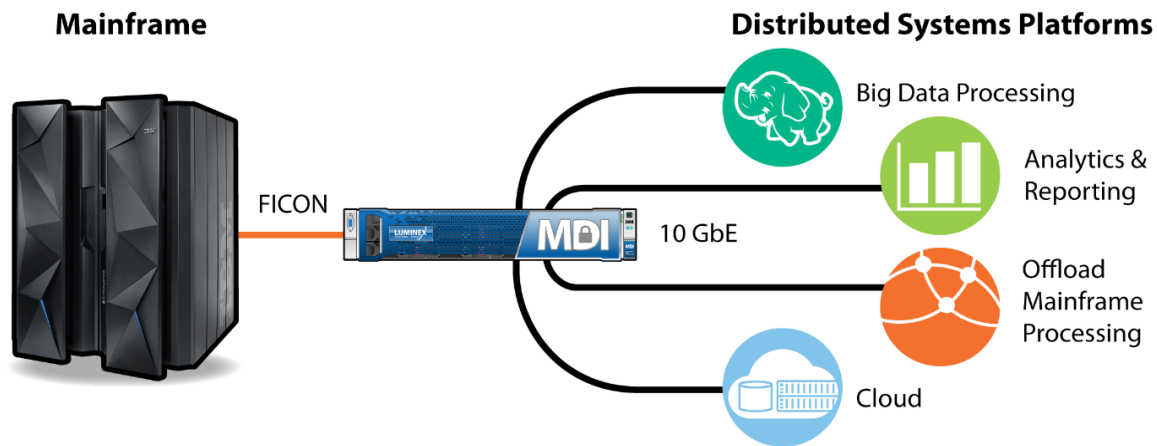
This section provides a general overview of the MDI architecture and key terms.

2.1 General Overview of Mainframe Data Integration

To implement one or more MDI Solutions, the following is required:

- The Luminex Channel Gateway (CGX) MDI Platform
- One or more MDI solutions; SecureTransfer, XPDS, BigData Transfer, or SAS Language Processor
- MDI-emulated 3590 tape drives on a mainframe FICON channel
- Connectivity to one or more distributed systems platforms
- Virtual tape volumes defined in the tape management system for use with data transfers
- Open Systems NFS storage
- A 10 GbE (1 GbE is also supported) ethernet connection to the network
- Mainframe load modules installed in accessible libraries
- Mainframe JCL, including MDI product specific parameters use to communicate to the MDI Platform
- RACF (ACF2 or Top-Secret) security controls

MDI Solutions allow bidirectional sharing, transformation, and movement of data between mainframe and distributed systems Platforms.



Depending on the use case, MDI Platforms can be configured with a disk cache using internal storage, or it can use dedicated or shared external storage.

When data is sent to the distributed systems platform, it can be converted to data consistent with that platform. When it is retrieved from the platform, it is presented to the mainframe in a virtual tape context that is readable by the mainframe.

2.2 Terms

Name	Short Name	Description
3590 Virtual Tape	VTAPE	Virtual tape defined in the client's environment for use by the MDI solution to move data from and to the mainframe.
Channel Gateway X	CGX	The name of the Luminex virtual tape controller.
Data Transformation or conversion	Transformation	The act of transforming data to enable its use on another Platform or in another language. As an example, EBCDIC to ASCII.
DCB Table	DCB Table	A product file that provides a means to associate file names, DCB attributes, conversion information and so forth for data coming from distributed to the mainframe
File Watcher	File Watcher	Code on the MDI Platform that "watches" the directories for files to send to the mainframe. Multiple File Watchers may be deployed
LUMXPROC	LUMXPROC	Name of the mainframe load module and procedure that is executed on the mainframe. Provides communication between mainframe and MDI Platform
MDI Batch Interface	LUMXPROC	Name of the mainframe load module and procedure that is executed on the mainframe. Provides communication between mainframe and MDI Platform
MDI BigData Transfer	MDI:BDT	BigData Transfer allows for mainframe data to be integrated into a Hadoop Distributed File System (HDFS) for data analytics using FICON in place of TCP/IP for better performance and security.
MDI Cross-Platform Data Sharing	MDI:XPDS	Cross-Platform Data Sharing allows for the integration of mainframe data with Open Systems applications using NFS mounted storage. Data from the mainframe can be made available for Open Systems applications and after processing, sent back to the mainframe for additional processing.
MDI Platform	MDI Platform	A Luminex Channel Gateway (CGX) appliance, which is a Linux server with a CentOS operating system, configured with one or more MDI solutions.
MDI Profile	Client determined	Technical name of the profile that is called/invoked on the MDI Platform when executing LUMXPROC on the mainframe and code on the MDI Platform for each MDI Product.
MDI SAS Language Processor	MDI:SLP	SAS Language Processor executes programs written in the SAS Language off the mainframe to reduce system overhead associated with SAS processing.

MDI SecureTransfer	MDI:ST	SecureTransfer is a Managed File Transfer solution that uses FICON in place of TCP/IP on the mainframe and SFTP from the MDI Platform to securely transfer data.
Pending Data Queue	PDQ	Started task on the mainframe that communicates with the File Watcher on the MDI Platform
PDQ Queue Name	Queue Name	The name on the mainframe LPAR, SYSTEM or SYSPLEX associated with the PDQ started task. Multiple PDQ Started tasks are supported to distinguish what data is to be sent and cataloged on what system.
RACF, Top Secret or ACF2 Security Profile	Security Profile	RACF or other mainframe security product profile that is defined to the security product on the mainframe to govern the security of the entity it is protecting.
Security PassTicket	PassTicket	A security mechanism that allows for a one-time-only password that is generated by a requesting product or function. It's an alternative to the RACF (or other Security product) password or password phrase. PassTicket removes the need to send RACF passwords and password phrases across the network in clear text. It makes it possible to move the authentication of a mainframe application user ID from RACF to another authorized function executing on the host system or to the workstation local area network (LAN) environment.
Slots	Slots	Represents the number of files that can be processed simultaneously. Each File Watcher watches 8 slots by default, client can add more up to 20. Each slot is an individual directory on storage that is NFS mounted to the MDI Platform.
Tape Catalog Database	TCDB	The client's tape management system database used to store information about tape volumes in the client's computing environment.
Target System or Server	Target	A Unix, Windows or Linux target for the data transfer from the mainframe. A target can also be an HDFS or Cloud destination.

3. Components

Mainframe Data Integration solutions vary, but regardless of the type, each solution has required components, the MDI batch interface (LUMXPROC), a configured profile, an MDI Platform, virtual tape volumes and Open Systems storage. These components are covered in the following sections.

3.1 Mainframe Software/Batch Interface

Each MDI Solution uses a Luminex mainframe program, LUMXPROC. The LUMXPROC program is typically executed as the second step in a z/OS batch job. The JCL has one DD statement for each output and/or input tape file. These DD statements merely identify the input and output MDI virtual tape files for off-host processing.

Prior to the LUMXPROC step, an ICEGENER (or similar dataset copy program) step is typically executed to copy the data from disk to MDI virtual tape. This step is not necessary if the data already resides on MDI virtual tape. Each outgoing mainframe data file to be processed is written to and cataloged on its own MDI virtual tape VOLSER.

At completion of the LUMXPROC step, each incoming off-host file is cataloged as a virtual tape data set as defined by the corresponding DISP=(NEW,CATLG) DD statement.

LUMXPROC requires a SYSIN DD statement for passing parameters to each MDI Solution's profile. LUMXPROC waits until the MDI profile has completed its processing and then the job step completes. Informational messages are provided in the batch job's SYSPRINT report.

3.2 Profiles

A profile is a customized deployment of the software residing on the MDI Platform. Several profiles, each with unique mix of parameters, may exist for each MDI solution. The client chooses which parameters to define in the profile and which to define in the JCL. Parameters cannot exist both in the profile and in the JCL.

Profiles are created on the MDI Platform to define parameters such as the following:

- The destination information in the form of an IP address, Server Name, DNS name or directory name on NFS mounted storage
- User credentials
- Which encryption cipher to use
- The number of parallel tasks
- The number of retries and how long to delay between retries
- Any pre-actions or post-actions
- Working directory names for temporary files
- The types of data conversions to be performed, as an example, ASCII to EBCDIC, blocked or unblocked, and so forth

3.3 MDI Platform

The Mainframe Data Integration Platform (MDI) is an x86 Linux server with Luminex code that appears as a virtual tape subsystem to the mainframe. Data is transferred from and to the mainframe via virtual tape. A two-step batch interface on the mainframe is used to transfer data to and from the MDI Platform. ICEGENER (or similar utility) moves the data over FICON and a procedure, LUMXPROC, directs the profile on the MDI Platform to do something with the data. The framework enables agile customization to meet a specific client need.

3.4 Storage

Each MDI solution requires some amount of storage to house the data being transferred via virtual tape. In some cases, such in the case of MDI SecureTransfer, data on virtual tape is only needed until the file transfer is complete. Retention of the virtual tape is controlled by the client via the batch JCL described in the Mainframe Software/Batch Interface section. In other use cases, such as MDI SAS Language Processor, the data being transferred may need to be retained for some period of time.

To determine how much storage is required, the Luminex support team assists the client in sizing the environment for the MDI Solution. The client can choose to use storage already available in-house or purchase additional storage from Luminex.

3.5 Virtual Tape Volumes

The MDI Platform is a virtual tape subsystem. Luminex created the MDI solutions to utilize the existing Channel Gateway (CGX) virtual tape system. It is the virtual tape FICON channel that provides for communication between the mainframe and the MDI Platform. Data is transferred to the MDI Platform via virtual tape volumes controlled by the CGX. Therefore, virtual tape volumes need to be defined that are unique to this CGX system for use with the MDI solutions. Typically, these virtual tape volumes are temporary and can be scratched right away. However, with some MDI solutions, such as SAS Language Processor, the client may choose to house SMF data on these virtual tape volumes long-term. The number of virtual tape drives and volumes that need to be allocated varies by the type of MDI solution purchased. Luminex Support can assist in determining the number of virtual tape drives and volumes to assign.

4. Planning for MDI

This section details configuration and sizing considerations as well as planning information to prepare for a smooth implementation.

4.1 Sizing Inputs

Sizing is required to determine the client's need for Virtual Tape Drives, Virtual Tapes, MDI Platforms and Open Systems Storage. Each MDI Solution has unique sizing requirements. When more than one (1) MDI Platform is deployed for performance or high-availability, storage is required to share the virtual tape data between the two (or more) Platforms. The following questions should be considered and discussed with the Luminex Support team for proper sizing:

MDI SecureTransfer

- How many file transfers do you perform per day (on average)?
- How long (in days) do you want to retain the input (virtual tape) for the file transfer?
- How many large files are transferred?
- What is the average size of the large file transfers?
- What are your high-availability and disaster recovery requirements for file transfer?

MDI Cross-Platform Data Sharing

- How many data sets are moved to Open System storage per day (on average)?
- What is the average size of the data?
- How many large files are moved?
- What is the average size of the large files?
- How long (in days) does the data need to remain on Open System's storage?
- What are your high-availability and disaster recovery requirements for data to be shared with Open Systems processing?

MDI BigData Transfer

- How many data sets are moved to the Hadoop Distributed File System (HDFS) per day (on average)?
- What is the average size of the data?
- How many large files are moved?
- What is the average size of the large files?
- How long (in days) does the data need to remain on virtual tape before it can be deleted?
- What are your high-availability and disaster recovery requirements for moving data to the HDFS?

MDI SAS Language Processor

MXG Use Case:

- Will mainframe SMF data be stored on MDI:SLP owned virtual tape?
- How often (in hours or days) do you dump SMF data for processing?
- How large is your daily SMF file during normal production processing?
- How large is your daily SMF file during peak production processing?
- What are your daily, weekly, monthly, quarterly, and annual retention requirements for SMF

data?

- What is the average size of your MXG Performance Database (PDB) during normal production processing?
- What is the average size of your PDB during peak production processing?
- How long do you retain your daily, weekly, monthly, quarterly, annual PDBs?
- Approximately how many SAS Language programs are in use today?
- Each MDI:SLP batch job requires a minimum of three virtual tape drives; how many jobs need to execute concurrently?
- What is the Service Level Agreement (SLA) for making the PDB available to end-users and data analyst for reporting?
- What are your high-availability and disaster recovery requirements for processing SMF data?

Other SAS Programs Use Case:

- Approximately how many SAS Language programs are in use today?
- How many programs are executed concurrently at peak processing times?
- How long do you retain the input data?
- What is the average size of the input data?
- How long do you retain the output data?
- How do you distribute the output reports?
- What are your high-availability or disaster recovery requirements for these SAS programs?

4.2 SMF Analysis for Sizing Overview

A tool to help size the usage of FTP, FTPS, SFTP, other Managed File Transfer solutions and/or SAS on the client's system, is the SMF Analysis. This analysis can tell us how many file transfers occur, what data is being transferred and indicate where the peak processing occurs. For MDI:SLP clients, the analysis can provide information on the number of SAS programs that have executed, their job names, and how many unique job names found in the data. It can also show peak processing times for SAS programs. To create the analysis, the client needs to collect the following records from all LPARs:

- Type 30 subtype (2,3) interval records
- Type 70,72 RMF; Type 74 is optional
- Type 119 subtype (3,70) for FTP client and server records
- Type 119 subtypes (96,97) for IBM SSH (used by IBM SFTP and Co:Z)
- If 119's are not available on your processor, type 118 are acceptable, requiring subtype (3,70,74,75)

Types 30, 118 and 119 write other subtypes that are very large and not useful for our SMF Analysis reporting.

The Connect:Direct product does not cut SMF 119 records. If you are currently using Connect:Direct, reporting is available in the product that provides information about your file transfer activity.

Luminex provides an instruction sheet with sample JCL to assist the client in collecting these

records for sizing requirements.

Once the records are collected, Luminex creates reports and charts that show file transfer, tape and/or SAS program executions on the client's system. These reports and charts provide the data that is needed to size the number of MDI Platforms and the amount of storage that is needed to house the data.

4.3 Planning for MDI Installation Overview

Installation of any MDI solution starts with the installation of the MDI Platform which is the Luminex Channel Gateway (CGX) Linux server and all of the associated tasks surrounding the installation of a virtual tape subsystem. The Luminex Support team provides each client with an Install Planning Workbook to be utilized during the installation. This workbook records all of the information required for a successful implementation. The Luminex Support team arranges a series of calls with the client in order to review and collect the required information for the install activity.

Following the installation, or concurrent with its installation, the mainframe components are installed and tailored to your environment.

The following is a high-level list of the installation tasks and the associated sections of this manual to reference for more information:

- Install MDI Hardware (See *Luminex Hardware Installation and Planning*)
- Define and configure MDI Platform profiles (Performed by Luminex Support team)
- Ensure all applicable ports and firewalls are open (Assisted by Luminex Support team)
- Install MDI Software ([Section 6: MDI Mainframe Software Installation and Configuration](#))
- Install the mainframe components of the Pending Data Queue (PDQ) [if MDI:ST or MDI:XPDS] ([Section 8: MDI File Watcher and Pending Data Queue \(PDQ\)](#))
- Create security profiles and permit users ([Section 7: Mainframe Security Setup](#))
- Generate JCL (See the solution-specific section of this manual and the JCL SAMPLIB provided with the product).
- Acquire Open Systems storage and NFS mount point
- Ensure an adequate number of virtual tape drives and volumes have been defined
- Perform system and/or benchmark testing as desired

5. MDI Profile Creation and the Installation Workbook

The Luminex Support team provides each client with an Installation Workbook unique to their site. A copy of the customized workbook is retained by the Luminex Support team for as long as Luminex solutions are installed.

This workbook assists the client and the Luminex Support team in the installation tasks. Once the MDI Platform is installed, the Luminex Support team creates one or more customized MDI Profiles for use with the MDI solution(s) purchased. The client names the profile(s) using a nomenclature that is meaningful to their environment. The Luminex Support team assists in this process.

The workbook contains several tabs:

- IO Definition
- CGX IP Networking (Basic)
- MDI xx (where xx represents the MDI solution being installed)
- CGX_MDI Installation Checklist
- Open Issues
- Sample MDI Test Plan
- HCD Report
- License Keys
- Support Certificate

This workbook is reviewed, and information provided by the client. The name(s) of the MDI profile(s) created must be communicated to the z/OS Systems Administrator and the Security Administrator in order to build security rules to govern who can use the profiles.

The workbook is a living document and each one is unique to the customer site.

6. MDI Mainframe Software Installation and Configuration

6.1 Installation Checklist

Item #	Task	Assigned To
1	Upload and install the mainframe load modules: LUMXPROC, LSCRUP and when applicable LUMXPDQ, LUMXPDQC, LUMXPDQR, LUMXPDQU, LUMXPDQV Upload the PDQ DCBTABLE sequential file to the z/OS mainframe.	z/OS Systems Administrator
2.	Upload the sample PROCLIB and SAMPLIB	z/OS Systems Administrator
3.	Collect the name of the MDI profiles and the direction (PUT, GET or BOTH) created on the MDI Platform from the Workbook.	z/OS Systems Administrator + Luminex Support
4.	Define the mainframe security rules to grant read access to the MDI profiles. Setup security for the LUMXPDQ started task (when applicable). Setup security for the PassTicket if using MDI:ST or MDI:XPDS. See Mainframe Security Setup for more information.	Security Administrator + Luminex Support
5.	Create JCL for LSCRUP and schedule job to run daily	z/OS Systems Administrator
6.	Create JCL for LUMXPROC procedure and test	z/OS Systems Administrator
7.	Create the PDQ started task and test (when applicable)	z/OS Systems Administrator
8.	Verify that the Tape Management System is updating the TCDB scratch status.	z/OS Systems Administrator

Step 1. Upload the LOADLIBs

The load modules are shipped in TSO XMIT format. See [Appendix E: Restore .XMT File Using TSO Receive](#) for instructions on how to perform a TSO receive of the product libraries.

Batch Interface and Scratch Update Load Modules

The load modules LUMXPROC and LSCRUP can be placed into a non-authorized load library.

LUMXPROC Load Module

All MDI Solutions utilize a mainframe program and procedure by the same name, LUMXPROC. This non-APF authorized program is used to communicate between the mainframe and the MDI

Platform. The LUMXPROC program is executed as a step in a z/OS batch job. It has one DD statement for each output and/or input MDI virtual tape file. These DD statements identify the input and output MDI tape files for the off-host processing.

Prior to the LUMXPROC step, each outgoing mainframe data file to be processed must have been written to and cataloged on an MDI virtual tape VOLSER. This is accomplished by writing the file to the MDI owned virtual tape as step one of the batch job or at some point prior to running LUMXPROC. At completion of the LUMXPROC step, each incoming off-host file is cataloged as a tape data set as defined by the corresponding DISP=(NEW,CATLG) DD statement.

LUMXPROC also requires a SYSIN DD statement for passing parameters to the MDI profile. LUMXPROC waits until the MDI profile has completed its processing and then the job step completes.

For more information about the LUMXPROC load module, refer to [Section 3.1: Mainframe Software/Batch Interface](#).

Scratch Update Processing LSCRUP Load Module

LSCRUP is a load module to be installed on the mainframe to extract a list of scratch VOLSERs from your existing tape management report. The process is shipped as a README.FIRST.TEXT file inside of the LSCRUP.ZIP software distribution and included a LOADLIB and sample JCL for the Luminex LSCRUP program.

Pending Data Queue (PDQ) Load Modules

LUMXPDQ, LUMXPDQC, LUMXPDQR, LUMXPDQU and LUMXPDQV require APF authorization and must be placed in an authorized library.

LUMXPDQ is a name of an authorized load module and a started task procedure to be installed on the mainframe. Both MDI SecureTransfer and MDI Cross-Platform Data Sharing use this MDI feature called Pending Data Queue or PDQ. This feature allows data from Open Systems or other mainframes to be sent securely to your mainframe.

The Luminex **Pending Data Queue (PDQ) started task** establishes the FICON connection(s) with the MDI Platform, and handles receiving and cataloging data where the distributed-systems side needs to push data to the mainframe.

The Luminex Pending Data Queue started task must be started after the tape management system task (such as RMM) at IPL.

Step 2. Upload the Sample PROCLIB and JCL SAMPLIB

The procedure library is shipped in TSO XMIT format. See [Appendix E: Restore .XMT File Using TSO Receive](#) for instructions on how to perform a TSO receive of the product libraries.

The PROCLIB contains procedures for LUMXPROC and LUMXPDQ. The sample procedures can be tailored and deployed in the appropriate procedure library. LUMXPDQ

runs as a started task; all of the requirements, security and otherwise, that are required at your site to establish this procedure as a started task. The SAMPLIB contains JCL examples for all MDI solutions.

Step 3. Collect Name(s) and Direction(s) of MDI Profile(s)

The Workbook is used to record the names of the MDI profiles created and the direction (PUT, GET or BOTH) as part of the installation of the MDI Platform.

Refer to [Section 5: MDI Profile Creation and the Installation Workbook](#) for more information on the MDI profile name(s) and direction(s).

Step 4. Define Security Profiles and Permit Users

Security credentials govern who can use the MDI profiles in the client environment.

Refer to [Section 7: Mainframe Security Setup](#) for instructions on setting up security and permitting user IDs for MDI profiles.

The Pending Data Queue requires additional security provisioning for use of the PassTicket. See [Section 7: Mainframe Security Setup](#) for more information.

Step 5. Setup LSCRUP JCL and Schedule to Execute Daily

The Luminex utility, LSCRUP, is executed to extract scratch VOLSERS from a tape management scratch report. The extracted VOLSER list is used to update the MDI Platform scratch list. The JCL needs to be customized and scheduled to execute shortly after daily the daily tape management system (TMS) housekeeping tasks. The STEPLIB in the procedure should point to the library containing the installed load module. For more information about the LSRUP utility, refer to [Appendix C: LSCRUP Scratch List Update Utility](#).

Step 6. Create JCL for LUMXPROC and Test

The PROCLIB distribution file contains procedures for LUMXPROC for you to tailor for your environment. The STEPLIB in each procedure should point to the library containing the installed load module. Edit the PROFILE= parameter to point to the profile name created for your site. Perform initial testing of the procedure. Contact Luminex Support if assistance is needed.

If needed in the LUMXPROC procedure, set the VPRE=parm to restrict MDI activity to virtual tape VOLSERS with this prefix.

Step 7. Create the PDQ Started Task and Test

The PROCLIB distribution file contains procedures for the PDQ started task. Once the security requirements for the PDQ started task have been completed. The task can be started, preferably on a test system for initial testing.

See [Section 8: MDI File Watcher and Pending Data Queue \(PDQ\)](#) for more information. Contact Luminex Support if assistance is needed.

Step 8. Verify the Tape Management System

After some initial testing and creation of output virtual tape files, execute the LSCRUP utility and verify that the tape management system (TMS) is updating the TCDB scratch status. This should be done after the some of the MDI Platform virtual tape VOLSERs have expired and returned to scratch status. The use attribute of the used VOLSER in the TCDB should be PRIVATE immediately after the test. After the subsequent daily TMS housekeeping, the VOLSER should be in SCRATCH status in the TMS and use attribute SCRATCH in the TCDB.

6.2 Sample LUMXPROC Procedure (PROC)

Here is a sample PROC that could be used for MDI processing.

```
//LUMXPROC PROC PROFILE=
//*
//* LUMINEX MDI PROCESSING: BATCH INTERFACE
//*
//X          EXEC  PGM=LUMXPROC,REGION=200K,PARM='PROFILE=&PROFILE'
//STEPLIB DD    DISP=SHR,DSN=LUMINEX.MDI.LOADLIB
//SYSPRINT DD    SYSOUT=*
//          PEND
```

If there is a need to implement a restriction on VOLSER prefixes (example: all VOLSERs must start with MD), then the PROC could be set up as follows.

```
//LUMXPROC PROC PROFILE=
//*
//* LUMINEX MDI PROCESSING: BATCH INTERFACE
//* MDI IS RESTRICTED TO VOLSER PREFIX: MD
//*
//X          EXEC  PGM=LUMXPROC,REGION=200K,
//          PARM='VPRE=MD,PROFILE=&PROFILE'
//STEPLIB DD    DISP=SHR,DSN=LUMINEX.MDI.LOADLIB
//SYSPRINT DD    SYSOUT=*
//          PEND
```

6.3 Sample LUMXPDQ PROC

Below is a sample PROC that could be used for MDI PDQ processing.

```
//PDQ          PROC
//*
//* LUMINEX PENDING DATA QUEUE
//*
// SET QNAME=SYSPLEX2
//*
//*****
//PDQ          EXEC  PGM=LUMXPDQ,REGION=0M,
// PARM='QNAME=&QNAME,APPID=MDIWFILE'
//*
//STEPLIB DD DISP=SHR,DSN=LUMINEX.MDI.PDQ.LOADLIB
//SYSPRINT DD SYSOUT=*
//CMDPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//*
```

```
//PARMFILE DD DISP=SHR,DSN=LUMINEX.MDI.PDQ.PARMLIB(PDQPARM)
//*
//*****
//
//          PEND
```

The PARMFILE DD specifies the PDQ parameter file. The parameters in that file are explained in this document in [Section 8.8: PDQ Parameters](#).

The SYSPRINT DD is used by PDQ for an activity log. Sample activity log messages are shown below.

CG	DATE	TIME	USERID	DEV	ACT	VOLSER	STATUS	DATA-SET-NAME
01	12.11	111844	RSOSM1	9A3E+	RSV	Q04660	VOL-RESERVED	RS.ORDER.NEW.SM112
01	12.11	111854	IAREP99	9A3D+	RSV	Q04661	VOL-RESERVED	SHIP.ARCHIVE.PHOTO.DAM-AGE.BIN
01	12.11	111855	RSOSM1	9A3E-	CAT	Q04660	DSN-CATALOGED	RS.ORDER.NEW.SM112
01	12.11	111856	IAREP99	9A3D-	CAT	Q04661	DSN-CATALOGED	SHIP.ARCHIVE.PHOTO.DAM-AGE.BIN

The CMDPRINT DD (optional) can be used by PDQ as an alternative to the console for responses to the PDQ operator commands. However, this must be enabled by the DDCMDPRINT=Y statement in the parm file.

The SYSTSPRT DD contains any SAY messages from the DCB exit program (REXX).

7. Mainframe Security Setup

The SAF interface is used to control what user ID(s) have access to perform operations at the installation site. The LUMXPROC program verifies that the JOB submitter has the proper authority to use the specified MDI profile(s). Each MDI solution has unique security setup requirements. The Pending Data Queue (PDQ) requires a started task setup on the mainframe as well as a security PassTicket setup for each user ID sending data to the mainframe.

The examples in this section show how to define the security profiles using IBM's RACF security product. If you are using one of Computer Associates (CA) products, Top Secret or ACF2, and your Security Administrator is not able to convert these RACF commands to your product's syntax, you may open a support ticket with CA and they will convert the commands to either Top Secret or ACF2 for you.

7.1 MDI Security System Definitions

Each MDI product uses profiles objects that resides on the MDI Platform, customized for the site by the Luminex Support team. Program LUMXPROC communicates with the profiles on the MDI Platform via commands and control information passed over FICON using the mainframe tape protocol. The LUMXPROC program obtains information from the profile which is used to formulate a call to the z/OS System Authorization Facility (SAF) to determine whether the user ID has the proper authority to use the specified profile. The key is the Profile name, which is specified in the JCL EXEC statement PARM= parameter. It has these characteristics:

- **mdi-profile-name** is a 1 to 16-byte character string that you have selected for your site, such as 'HOST1PUT', 'DEPT2GET', or 'APP3BOTH' as examples. The first character must be an upper-case letter (A to Z). The remaining characters can be upper-case letter (A to Z) and/or numeric (0 to 9). More examples of profile names and RACF statements are provided at the end of this section.

The LUMXPROC mainframe program passes the profile name to the MDI Platform and obtains the following information pertaining to security:

- **mdi-profile-class** is a fixed name assigned to each MDI product. The mdi-profile-class is the hierarchy that all security profile names are defined under. The following is the assigned class names for each MDI solution:

MDI Solution	Class Name
SecureTransfer (ST)	SFTP
Cross-Platform Data Sharing (XPDS)	EXP
SAS Language Processor (SLP)	SLP
BigData Transfer (BDT)	HDFS
zKconnect	ZBUS

- **mdi-direction** is either PUT, GET or BOTH.
 - A PUT operation is described as sending the file from the mainframe to another destina-

tion. Defining a security profile for PUT operations allows the site to control who can execute LUMXPROC to send files from the mainframe to another destination.

- A GET operation is described as receiving a file to the mainframe from another source. Defining a security profile for GET operations allows the site to control who can execute LUMXPROC to receive (pull) files to the mainframe.
- BOTH can be used when no specification is required in permitting user IDs to file transfer operations. When BOTH is used, only one security profile is required to be defined to your security system. You don't need to define separate profiles for PUT and GET operations.

These options are specified by the site and added to the profile by the Luminex Support team at profile creation. After LUMXPROC obtains the profile information it makes this SAF call:

```
RACROUTE REQUEST=AUTH,CLASS='FACILITY',ATTR=READ
```

And checks the following entity:

```
LUMXPROC.<class>.<mdi-direction>.<mdi-profile-name>
```

7.2 Security Definitions

A Security Administrator needs to define a FACILITY class profile or profiles to your security server and permit user ID(s) to the profile in order to successfully execute LUMXPROC on your system.

Step 1: Gather the following information:

- mdi-direction – PUT, GET or BOTH
- mdi-profile-name – name of the SecureTransfer profile or profiles determined by the installation site
- owner ID – the name of a user or group that owns the SecureTransfer FACILITY class profile(s).
- access-list – a list of users and groups who are authorized to execute the SecureTransfer profile(s)

Step 2: Define the FACILITY class profile to RACF.

Sample statement 1 is the format for a FACILITY class profile for the SecureTransfer solution using class name SFTP:

```
RDEFINE FACILITY LUMXPROC.SFTP.mdi-direction.mdi-profile-name OWNER(owner-ID)  
UACC (NONE)
```

Sample statement 2 is the format for a FACILITY class profile for the XPDS solution using class name OXP.

```
RDEFINE FACILITY LUMXPROC.OXP.mdi-direction.mdi-profile-name OWNER(owner-ID)  
UACC (NONE)
```

Sample statement 3 is the format for a FACILITY class profile for the SAS Language Processor solution using class name SLP.

```
RDEFINE FACILITY LUMXPROC.SLP.mdi-direction.mdi-profile-name OWNER(owner-ID)
UACC (NONE)
```

Sample statement 4 is the format for a FACILITY class profile for the BigData Transfer solution using class name HDFS.

```
RDEFINE FACILITY LUMXPROC.HDFS.mdi-direction.mdi-profile-name OWNER(owner-ID)
UACC (NONE)
```

Step 3: After defining the FACILITY class profile that has been RACLISTed, the Security Administrator needs to issue a SETROPTS refresh command to update the in memory RACF list.

Sample statement:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Step 4: Lastly, permit users or groups to the security profile to enable those users or group members to execute LUMXPROC.

Sample statement 1 to permit users to execute the SecureTransfer profile:

```
PERMIT LUMXPROC.SFTP.mdi-direction.mdi-profile-name CLASS(FACILITY) ID(access-list)
ACCESS(READ)
```

Sample statement 2 to permit users to execute the Cross-Platform Data Sharing profile.

```
PERMIT LUMXPROC.OXP.mdi-direction.mdi-profile-name CLASS(FACILITY) ID(access-list)
ACCESS(READ)
```

Sample statement 3 to permit users to execute the SAS Language Processor profile.

```
PERMIT LUMXPROC.SLP.mdi-direction.mdi-profile-name CLASS(FACILITY) ID(access-list)
ACCESS(READ)
```

Sample statement 4 to permit users to execute the BigData Transfer profile.

```
PERMIT LUMXPROC.HDFS.mdi-direction.mdi-profile-name CLASS(FACILITY) ID(access-list)
ACCESS(READ)
```

7.3 Security Examples for MDI SecureTransfer Profile Definition Scenarios

Scenario #1:

SecureTransfer profile PAYDATA is configured to perform the PUT operation and has no hard-coded destination (IP address, server name or DNS name) in the profile definition on the MDI Platform. When the site uses this profile, they will specify the destination of the PUT operation in the JCL SYSIN data stream. Only the Production Services (group PRODSERV) and certain developers (DEVEL1 and DEVEL2) can use this profile.

RACF Security Definition Examples for Scenario #1:

```
RDEFINE FACILITY LUMXPROC.SFTP.PUT.PAYDATA OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.SFTP.PUT.PAYDATA CLASS(FACILITY) ID(PRODSERV) ACCESS(READ)
PERMIT LUMXPROC.SFTP.PUT.PAYDATA CLASS(FACILITY) ID(DEVEL1) ACCESS(READ)
PERMIT LUMXPROC.SFTP.PUT.PAYDATA CLASS(FACILITY) ID(DEVEL2) ACCESS(READ)
```

Scenario # 2:

SecureTransfer profiles CLOUD1GET and CLOUD1PUT are configured on the MDI Platform to perform GET and PUT operations respectively from and to the corporate data lake. All programmers who are members of group APPDEV can PUT files to the data lake. Analysts that are members of group CLBUSDEV can GET files from it.

RACF Security Definition Examples for Scenario #2:

```
RDEFINE FACILITY LUMXPROC.SFTP.GET.CLOUD1GET OWNER(MDIADMIN) UACC(NONE)
RDEFINE FACILITY LUMXPROC.SFTP.PUT.CLOUD1PUT OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.SFTP.GET.CLOUD1GET CLASS(FACILITY) ID(CLBUSDEV) ACCESS(READ)
PERMIT LUMXPROC.SFTP.GET.CLOUD1PUT CLASS(FACILITY) ID(CLBUSDEV APPDEV) ACCESS(READ)
```

Scenario # 3:

Profile FINBOTH was configured to send (PUT) and receive (GET) Finance department data to and from the company EDI server. Only members of the finance department who are in group FINEDI are able to use this profile, as well as batch jobs that are executed via the job scheduler, user ID JOBSCHEDA.

RACF Security Definition Examples for Scenario #3:

```
RDEFINE FACILITY LUMXPROC.SFTP.BOTH.FINBOTH OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.SFTP.BOTH.FINBOTH CLASS(FACILITY) ID(FINEDI JOBSCHEDA) ACCESS(READ)
```

Scenario # 4:

A company has implemented SecureTransfer and has decided to restrict usage to scheduled batch jobs that have met rigorous change control requirements and the SecureTransfer administration group. In this example, a single generic profile is all that is required for the enterprise. The SecureTransfer user ID is MDIADMIN and the job scheduler user ID is JOBSCHEDA.

RACF Security Definition Examples for Scenario #4:

```
RDEFINE FACILITY LUMXPROC.** OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.** CLASS(FACILITY) ID(MDIADMIN JOBSCHEDA) ACCESS(READ)
```

7.4 Security Examples for MDI:XPDS Profile Definition Scenarios

Scenario #1:

MDI:XPDS profile APPKLM is configured to drop files into a directory pre-defined on NFS mounted storage for the KLM application. The NFS Mount address has been configured in the profile definition on the MDI Platform. User IDs permitted to this profile can send data from the mainframe to this storage device (PUT) and pull (GET) data from this device. The KLM application group (KLM023, KLM056, KLM330), KLM developers (DEVEL1 and DEVEL2) can use

this profile.

RACF Security Definition Examples for Scenario #1:

```
RDEFINE FACILITY LUMXPROC.OXP.BOTH.APPKLM OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.OXP.BOTH.APPKLM CLASS(FACILITY) ID(KLM023) ACCESS(READ)
PERMIT LUMXPROC.OXP.BOTH.APPKLM CLASS(FACILITY) ID(KLM056) ACCESS(READ)
PERMIT LUMXPROC.OXP.BOTH.APPKLM CLASS(FACILITY) ID(KLM330) ACCESS(READ)
```

Scenario #2:

MDI:XPDS profile TOHP is configured to drop files into a directory pre-defined on NFS mounted storage for the HP application team. The NFS Mount address has been configured in the profile definition on the MDI Platform. User IDs permitted to this profile can send data from the mainframe to this storage device (PUT) only. The HP application group (HPALM, HPRWG, HPKBL) and the site's scheduling system's user ID(CA7ID) can use this profile.

RACF Security Definition Examples for Scenario #1:

```
RDEFINE FACILITY LUMXPROC.OXP.PUT.TOHP OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.OXP.PUT.TOHP CLASS(FACILITY) ID(HPALM) ACCESS(READ)
PERMIT LUMXPROC.OXP.PUT.TOHP CLASS(FACILITY) ID(HPRWG) ACCESS(READ)
PERMIT LUMXPROC.OXP.PUT.TOHP CLASS(FACILITY) ID(HPKBL) ACCESS(READ)
PERMIT LUMXPROC.OXP.PUT.TOHP CLASS(FACILITY) ID(CA7ID) ACCESS(READ)
```

7.5 Security Examples for MDI:SLP Profile Definition Scenarios

Scenario #1:

MDI:SLP profile MXGKT01 is configured to transfer the site's SMF data and SAS language programs to the storage attached to the MDI Platform for MXG processing. The NFS Mount address has been configured in the profile definition on the MDI Platform. User IDs permitted to this profile can send data from the mainframe to this storage device (PUT) and pull (GET) data from this device. The Capacity Planning group (TECH01, TECH53, TECH24, TECH36) and the site's scheduling system (CNTLMKT1) can use this profile.

RACF Security Definition Examples for Scenario #1:

```
RDEFINE FACILITY LUMXPROC.SLP.BOTH.MXGKT01 OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.SLP.BOTH.MXGKT01 CLASS(FACILITY) ID(TECH53) ACCESS(READ)
PERMIT LUMXPROC.SLP.BOTH.MXGKT01 CLASS(FACILITY) ID(TECH24) ACCESS(READ)
PERMIT LUMXPROC.SLP.BOTH.MXGKT01 CLASS(FACILITY) ID(TECH36) ACCESS(READ)
PERMIT LUMXPROC.SLP.BOTH.MXGKT01 CLASS(FACILITY) ID(CNTLMKT1) ACCESS(READ)
```

Scenario #2:

MDI:SLP profile SLPCECLB is configured to transfer the site's SAS language programs and input data to the storage attached to the MDI Platform for SAS processing. The NFS Mount address has been configured in the profile definition on the MDI Platform. User IDs permitted to this profile can send data from the mainframe to this storage device (PUT) and pull (GET) data from this device. The SAS Programming group (SAS01HH, SAS44JK, SAS23NTM, SAS9B-K1A) and the site's scheduling system (OPSADMIN) can use this profile.

RACF Security Definition Examples for Scenario #1:

```
RDEFINE FACILITY LUMXPROC.SLP.BOTH.SLPCECLB OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.SLP.BOTH.SLPCECLB CLASS(FACILITY) ID(SAS01HH) ACCESS(READ)
PERMIT LUMXPROC.SLP.BOTH.SLPCECLB CLASS(FACILITY) ID(SAS44JK) ACCESS(READ)
PERMIT LUMXPROC.SLP.BOTH.SLPCECLB CLASS(FACILITY) ID(SAS23NTM) ACCESS(READ)
PERMIT LUMXPROC.SLP.BOTH.SLPCECLB CLASS(FACILITY) ID(SAS9BK1A) ACCESS(READ)
PERMIT LUMXPROC.SLP.BOTH.SLPCECLB CLASS(FACILITY) ID(OPSADMIN) ACCESS(READ)
```

7.6 Security Example for MDI BigData Transfer Profile Definition Scenario

Scenario #1:

MDI:BDT profile TOAZURE is configured to transfer the site's VSAM files and Copybooks to Azure. User IDs permitted to this profile can send data from the mainframe to Azure (PUT). The Data Analyst group (DKTAL, DKTKK, DKTYK) and the site's scheduling system (OPER) can use this profile.

RACF Security Definition Examples for Scenario #1:

```
RDEFINE FACILITY LUMXPROC.HDFS.PUT.TOAZURE OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.HDFS.PUT.TOAZURE CLASS(FACILITY) ID(DKTAL) ACCESS(READ)
PERMIT LUMXPROC.HDFS.PUT.TOAZURE CLASS(FACILITY) ID(DKTKK) ACCESS(READ)
PERMIT LUMXPROC.HDFS.PUT.TOAZURE CLASS(FACILITY) ID(DKTYK) ACCESS(READ)
PERMIT LUMXPROC.HDFS.PUT.TOAZURE CLASS(FACILITY) ID(OPER) ACCESS(READ)
```

7.7 PDQ Security Definitions

Since LUMXPDQ typically executes as a started task, the normal rules for defining a started task PROC, including defining a User ID for the task, to the security system apply. These steps are not detailed in this section. Additional detailed information on PDQ and security is provided in [Appendix B: Security Specifications](#).

To set up the security definitions, the following items must be known.

- File Watcher Application ID (MDIWFILE)
- Queue Name (assigned by installer)
- Name of the PDQ started task(s) (assigned by installer)
- Data set names (HLQ) of data sets to be pushed to the mainframe (if not already protected by a security profile)
- User Id(s) or group ID that are allowed to push files onto the mainframe and catalog them
- The secret 16-hex-char token that is to be associated with all File Watcher user IDs for Pas-sTicket generation and mainframe validation. (created by Security Administrator)
- Prefix Value of the DSNPREF parameter set in the started task (assigned by the installer)

The File Watcher resides on the MDI Platform and watches for new files on a mounted file system. The application ID of this file watcher is "MDIWFILE".

The Queue Name should be unique across the entire data center. At a minimum, there should be a different queue name for each sysplex. This is because the distributed systems files need to be pushed to a specific sysplex so that the mainframe applications on that sysplex can process their designated files. It is possible to have more than one queue name per sysplex.

The examples in this section use IBM RACF command syntax but only those that are relevant to PDQ are shown. Continuations of the command are indented on the subsequent line. The items that must be substituted into these commands are bracketed with “<” and “>”. The full name of the item was mentioned above, but only an abbreviation appear in the syntax example. These substitutions are taken from the following:

<appid> <qname> <stcname> <userid> <token>

7.7.1 Started Task

The started task needs to be given authority to access the PDQ queue name. See [Appendix B: Security Specifications](#).

```
RDEFINE FACILITY LUMXPDQ.CONNECT.<qname> UACC(NONE) ...
```

```
PERMIT LUMXPDQ.CONNECT.<qname> CLASS(FACILITY) ID(<stcname>)  
ACCESS(READ) ...
```

The started task needs to be given authority to create the dynamically created data set that is a result of the prefix value of the DSNPREF parameter set in the started task by the installer.

The DSNPREF parameter provides a prefix for the data set that is dynamically created. As an example, if the DSNPREF= is set to SYSS.PDQ, the resulting data set name contains the following:

SYSS.PDQ.*stcid.ddname*

Where *stcid* is the started task ID assigned by the operating system to the PDQ started task and the *ddname* is the DDNAME used for the device address such as COMM038F, where 038F is the device address. Since these values are not known until the task is started, and can change when the task is recycled, the security profile to be created must use generic substitution values for the last 2 nodes of the data set name. An example, using SYSS.PDQ as the DSNPREF value:

```
ADDSD 'SYSS.PDQ.**' GEN OWNER(group) UACC(NONE)
```

The started task ID must be permitted to this profile with UPDATE access. As an example:

```
PERMIT 'SYSS.PDQ.**' GEN ACCESS(UPDATE) ID(stcid)
```

Where *stcid* is the started task User ID.

7.7.2 USERID and TOKEN

The USER ID(s) that push files to the mainframe do not have to be the same as any existing USER ID that has mainframe LOGON privileges. But the USER ID needs permission to create a

tape data set name that contains the file data that is being pushed.

If the user IDs that are to be used for MDI PDQ are not currently defined, they need to be defined to the mainframe security system.

By convention, the MDI Platform file watcher application generates a PassTicket and uses that in place of the mainframe password. This means that the conventional mainframe passwords are not used by PDQ and thus don't need to be given to the File Watcher application or to any distributed system (user).

The use of PassTicket requires the PTKTDATA security class. This class must be activated in the mainframe security system. An example of the RACF commands to create this class is provided below:

```
SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKDATA)
SETROPTS RACLIST(PTKTDATA) REFRESH
```

The LUMXPDQ program does not specifically call the SAF router to test profiles in this class. Instead, the PassTicket authentication is built in to the normal SAF user ID validation process. Failure to properly set up the PTKTDATA profiles causes PassTickets to be treated as invalid passwords.

The best practice for setting up PTKTDATA profiles is to define a separate profile for each (File Watcher) user ID that is to be used to push files to the mainframe.

The File Watcher secret token must be assigned to each user ID. The token is 16 digits in length using hexadecimal digits (0123456789ABCDEF). The RACF administrator or other project members can create the token. The token should not start with zero (0).

Since this token is associated with the File Watcher application ID, it must be unique to PDQ processing and not be the same as any other tokens in use for other non-MDI applications.

MDI currently supports only one token value per MDI File Watcher application ID. Thus, each user ID should be defined using this same token value.

The UID profile should be defined as follows.

```
RDEFINE PTKTDATA MDIWFIL.<userid> SSIGNON(KEYMASKED(<token>))
UACC(NONE)
```

This PassTicket token must also be defined on the associated MDI Platform by the Luminex Support team.

If PTKTDATA is RACLISTed, you may also need to do the following.

SETROPTS REFRESH RACLIST(PTKTDATA)

In the following example, The PTKTDATA class is created as a RACLISTed resource, and UIDs TEK21, TEK45 and ADMING are permitted using token 1A2B2C3D4EF6789.

```
SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
RDEFINE PTKTDATA MDIWFIL.TEK21 SSIGNON(KEYMASKED(1A2B2C3D4E5F6789)) UACC(NONE)
RDEFINE PTKTDATA MDIWFIL.TEK45 SSIGNON(KEYMASKED(1A2B2C3D4E5F6789)) UACC(NONE)
RDEFINE PTKTDATA MDIWFIL.ADMING SSIGNON(KEYMASKED(1A2B2C3D4E5F6789)) UACC(NONE)
SETROPTS RACLIST (PTKTDATA) REFRESH
```

7.7.3 Data Set Name

The data sets to be created on the mainframe must have a valid high-level qualifier. Valid is defined as having an alias defined in an ICF catalog and RACF (or other) protections established. If the data set to be created is a GDG, a GDG base must also be defined. The UIDs pushing data to the mainframe must have ALTER access to the high-level qualifiers they are going to create.

Typically, the creation of data set profiles and access permissions is already well understood, and processes have been established by the security system administrator. At a minimum, each high-level qualifier should be protected by a generic profile such as the following for HLQ “MYDATA”.

```
ADDSD 'MYDATA.**' GEN OWNER(<group>) UACC(NONE) ...
PERMIT 'MYDATA.**' GEN ACCESS(ALTER) ID(<userid>)
```

7.7.4 Unprotected Data Set Names

By default, PDQ cannot create an unprotected tape data set name.

If the security system administrator has determined that not all data sets need to be protected, an additional security system setting is required to give permission to the PDQ started task to allow this. Define the following to the security system:

```
RDEFINE FACILITY LUMXPDQ DSNOPROF. <gname> UACC(NONE) ...
PERMIT LUMXPDQ.DSNOPROF.<qname> CLASS(FACILITY) ID(<stcname>)
ACCESS(READ) ...
```


8. MDI File Watcher and Pending Data Queue (PDQ)

8.1 Introduction

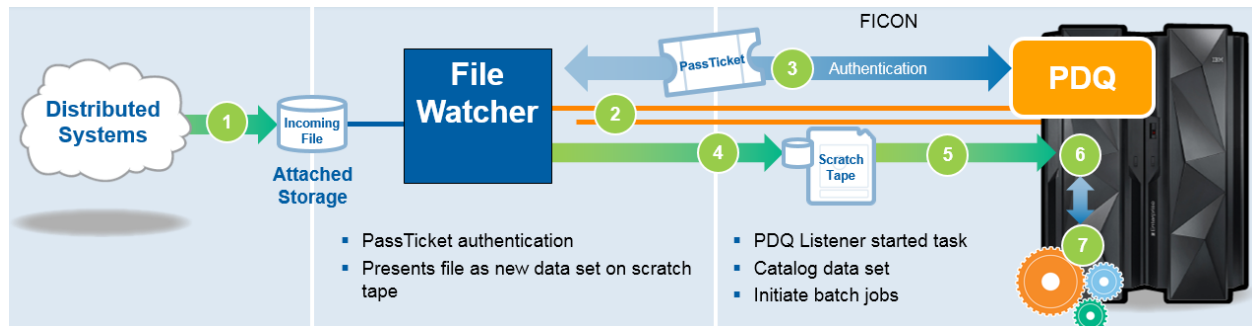
The Luminex **Pending Data Queue (PDQ) started task** establishes the FICON connection(s) with the MDI Platform, and handles receiving and cataloging data where the distributed-systems side needs to push data to the mainframe.

The **MDI File Watcher** monitors a directory structure into which a user can move files. When a file arrives and is complete (no longer changing), the File Watcher converts the file into a virtual tape image and works with PDQ to catalog the tape data set on the mainframe.

8.2 MDI File Watcher/PDQ Concepts

The MDI Platform (or collection of MDI Platforms) hosts one or more Luminex MDI file watchers for passing files to the mainframe. The file watcher monitors an NFS mounted storage device for incoming files, and then associates a mainframe user ID and data set name with each file. For each data set, various DCB characteristics also need to be provided so that the converted virtual tape data is in a format that can be read by mainframe access methods.

Whenever needed for an incoming distributed-systems file, the PDQ started task (STC) allocates a scratch MDI tape, and then catalogs the corresponding mainframe data set name when the file has been converted to virtual tape data on that MDI VOLSER. The PDQ STC can be connected to multiple MDI servers, and multiple PDQ started tasks can be connected to each MDI server. Each connection is established from the PDQ STC via dynamic allocation to a tape device on the corresponding MDI server.



PassTicket is a mechanism for shifting user authentication from the mainframe security system to an outside application. The Luminex MDI File Watcher application must perform this authentication or relay an already established authentication. For each file to be processed, a PassTicket for the authenticated user ID must be generated and passed to the mainframe. This PassTicket is used instead of a mainframe password and is only good for a relatively short period of time. The mainframe security system and the PassTicket generator must use the same secret token to generate or verify the PassTicket.

Since the computer data center could have more than one mainframe and more than one sysplex,

it is necessary to designate a PDQ queue name that (at a minimum) represents the sysplex that is to receive the distributed systems files. There may be reasons to have more than one queue name within a sysplex to set up different security rules for each queue or for accounting purposes.

For each file that is moved to the mainframe, there are data set characteristics that must be specified so that mainframe programs can make sense of the data. These are: (maximum) block size [BLKSIZE], (maximum) record size [LRECL], and record/blocking format [RECFM]. In addition, the file data can be converted from ASCII to EBCDIC, or converted using a pre-defined character conversion table.

8.3 Prerequisites

The following MDI components must be installed before or simultaneous with the install of MDI-PDQ:

- **MDI Platform** ... see *Luminex Hardware Installation and Planning Guide*. This includes the MDI Platform hardware and the corresponding z/OS requirements, and Luminex utility mainframe program(s).
- **MDI Batch Interface** ... see [Section 6: MDI Mainframe Software Installation and Configuration](#). This provides the LUMXPROC software for z/OS and is needed to update the PDQ file watcher DCBTABLE(s).

8.4 Planning and Installation

Many of the following steps involve the Installation team, a Distributed Systems Administrator, a z/OS Systems Programmer, and a Security Administrator. There are some steps that are not dependent on prior steps, but this sequence roughly follows the order in which decisions must be made. Additional information about some of these steps is provided in subsequent sections.

Task #	Task	Assignment
1.	Identify the distributed systems users or external clients that are providing files to be transferred to the mainframe. This does not need to be a complete list, just some representative cases of the intended activity.	z/OS Systems Programmer and Project Team
2.	Identify the distributed systems or external client's file names to be transferred to the mainframe. This does not need to be a complete list, just some representative cases of the intended activity	z/OS Systems Programmer and Project Team
3.	Identify the z/OS mainframe user IDs that are to be used to create the output files on the mainframe. These mainframe user IDs don't need to have mainframe logon privileges, they only require security authorization to create the output data set on the mainframe. Multiple distributed system users can share a mainframe user ID if desired.	z/OS Systems Programmer and Project Team

4.	Identify storage on which users can place files to be sent to the mainframe. This storage needs to be NFS mounted to the MDI Platform or configured to that the users and the MDI Platform have shared access.	z/OS Systems Programmer and Project Team
5.	Determine how the distributed system users are mapped or grouped into mainframe user IDs. Usually these two types of user IDs are not exactly the same. Some method is needed for associating one to the other.	z/OS Systems Programmer and Project Team, Security Administrator
6.	Decide, for each incoming file, how the corresponding mainframe data set name is determined. If all incoming file names are known in advance and seldom change, then a simple list of these file names can be associated with the corresponding mainframe data set name(s). In more complicated scenarios, it's possible to use portions of the file name, date and time. The use of mainframe GDG base names for the data set name should also be considered. When using a GDG, specify the base name (without the G00V0000) as the data set name. File names cannot exceed 240 characters including periods. Data set names that exceed the limit are moved to the hold directory.	z/OS Systems Programmer and Project Team, Security Administrator

7.	<p>Determine, for each file, the appropriate DCB characteristics and data/record/block conversion method.</p> <p>These attributes determine the format of the tape. Specifying the correct DCB characteristics requires some knowledge of what mainframe program is going to process the data, the structure and content of the distributed-system file, and whether any type of data conversion or record-breaking or block-breaking is required. For example, consider these questions for each incoming distributed-systems file.</p> <p>See Section 9: MDI Data Conversions for information on the available “Conversions Moving Data to the Mainframe.”</p> <p>RECFM must be one of the following: F FB FBS FBA FBM V VB VBS VBA VBM U</p> <p>LRECL must be in the range 1-32760, but is should not be specified if RECFM=U.</p> <p>BLKSIZE must be in the range 0-262144, but is typically specified as 0 to allow the mainframe to determine the optimum maximum block size for the tape device.</p>	z/OS Systems Programmer and Project Team
8.	<p>Identify the method by which users place files into the File Watcher directory structure and verify that it is secure. This is typically achieved by SFTP from the system the file originates from.</p> <p>The means of access to the directory structure is determined and maintained by the site’s infrastructure. It is incumbent on the site’s distributed systems administrator to configure the storage access means which may include usernames, passwords, and directory permissions.</p>	Distributed Systems Administrator

9.	Assign a queue name for the mainframe SYSPLEX that is the target of the file transfers. This is a 1-8 character alphanumeric value that must start with an alpha character. Note: SYSPLEXs or LPARs within a SYSPLEX with non-shared ICF catalogs require their own queue name. Multiple queue names are supported for multiple SYSPLEX/LPAR environments.	z/OS Systems Administrator
10.	Ensure that the security rules, both for the profile and for the PassTicket have been setup and have been tested. See Section 7: Mainframe Security Setup for more information.	z/OS Systems Administrator/Security Administrator

11.	<p>Create the File Watcher directory structure on the shared storage.</p> <p>The file watcher needs a directory structure that contains the queue name and the mainframe user ID.</p> <p><code>/.../queueName/userid/</code></p> <p>Any number of directories can be specified in place of the “...” in this example.</p> <p>As an example: <code>/tomf/prod/lpar1/tek21</code></p> <p>Action Required: Provide the directory name to the Luminex Support team. The Luminex Support team needs to code the directory name in the file watcher config file, parameter “watch_dir”.</p> <p>It is imperative that proper permissions and authorization be configured by the installation site such that the access to this directory structure is limited to authorized users.</p> <p>Note that all unrecognized files are moved to the “hold” directory.</p> <p>Incoming files that cannot be associated with a mainframe DSN are moved into a “hold” directory. When needed, this hold directory is created within user ID directory.</p> <p>Example: <code>/.../queueName/userid/hold</code></p> <p>To retrigger action by file watcher, the file should be moved (up one level) to the <i>userid</i> directory.</p> <p>Action Required: This handling of unrecognized files may need to be documented for the distributed-system users.</p>	Distributed Systems Administrator
-----	--	-----------------------------------

12.	<p>Grant permissions to the MDI Platform to access the File Watcher directory structure on the designated storage system.</p> <p>The MDI Platform must be able to scan directories, create the “hold” directory, delete files that have been processed, and move unrecognized files to the hold directory.</p>	Distributed Systems Administrator
13.	<p>Verify that a distributed system file that has not changed size in 60 seconds is sufficient time to stream the file as ready for transfer to the mainframe.</p> <p>This criterion may depend on how the file is being delivered into the File Watcher directory structure. If a longer detection time is required, the Luminex Support team can make this adjustment in the File Watcher config.</p>	Distributed Systems Administrator
14.	Create the mount point on the MDI Platform and connect to the directory structure.	Luminex Support Team
15.	<p>Install and configure the File Watcher on the MDI Platform.</p> <p>Use the queue name as the name of the config and file watcher object</p>	Luminex Support
16.	<p>Verify that the MDI Platform has the correct GMT. PASSTICKET generation and evaluation requires correct GMT.</p> <p>Verify that the mainframe LPARs have the correct GMT.</p>	Luminex Support and z/OS Systems Administration
17.	<p>Verify that the MDI Platform has correct GMT.</p> <p>PASSTICKET generation and evaluation requires correct GMT.</p> <p>Verify that the mainframe LPARs have the correct GMT.</p>	Luminex Support and z/OS Systems Administration

18.	<p>Add the TOKEN to the File Watcher security file on the MDI Platform.</p> <p>This token is randomly generated by the mainframe security administrator (see Section 7.1.2: USERID and TOKEN). It consists of 16 hex digits. This token value must be implemented in mainframe security and on each of the MDI servers.</p>	Luminex Support and Security Administration
19.	<p>Activate the File Watcher on the MDI Platform.</p> <p>Once this has been activated, it is always running on the MDI Platform (except during Platform maintenance).</p>	Luminex Support
20.	<p>Optionally create a DCBEXIT program (REXX) to evaluate the incoming distributed-system file name.</p> <p>The exit sets values for: DSN, RECFM, LRECL, BLK-SIZE, CONVERSION, EMAIL.</p> <p>This program should implement the decisions made earlier when considering “mainframe data set names” and “DCB characteristics”.</p> <p>A sample DCBEXIT program (REXX) is provided in the SAMPLIB as member DCBEXIT1.</p> <p>See Section 8.6: DCB Exit for more information.</p>	z/OS Systems Administrator and Luminex Support

21.	<p>Define other distributed-system file names in DCBTABLE.</p> <p>The DCBEXIT is the preferred method for defining distributed-systems files. However, for any distributed-systems file name not handled by the DCBEXIT program, that file name and its characteristics must be defined in a DCBTABLE file. The active table resides on the MDI Platform.</p> <p>The DCBTABLE file also contains global email settings for sending file arrival or file error messages.</p> <p>Create a sequential file that contains the global and/or filename settings that are to be applied in the next step.</p> <p>See Section 8.7: DCB Table for more information.</p>	z/OS Systems Administrator and Luminex Support
22.	<p>Update the DCBTABLE on the MDI Platform.</p> <p>Sample update JCL is provided in the SAMPLIB as member PDQUPDAT.</p> <p>This step must read the sequential file created/updated in the prior step and apply it to the MDI Platform.</p>	z/OS Systems Administrator and Luminex Support
23.	<p>Customize the PDQ parm file.</p> <p>Sample parameters are provided in the SAMPLIB member PDQPARM.</p> <p>There must be at least one DEV= statement that identifies the MDI Platform tape devices.</p>	z/OS Systems Administrator and Luminex Support
24.	<p>Ensure the PDQ started task has been customized for your site and testing on a test system.</p> <p>Sample STC procedure JCL is provided in the SAMPLIB member PDQ.</p>	z/OS Systems Administrator

25.	<p>After testing, start the PDQ started task on at least one LPAR in the SYSPLEX.</p> <p>Ensure your IPL instructions or automated operations has been updated to bring up this started task after IPL.</p> <p>Caution: Any distributed-systems files in the file watcher directory structure will now be acted upon. Prior to this point, the file watcher may have detected files, but it could not take action since PDQ was not yet started.</p>	z/OS Systems Administrator
26.	<p>Create daily audit job for production batch.</p> <p>The file watch on the MDI Platform generates “audit log” entries whenever it attempts to process an incoming distributed systems file. These might be needed when researching a file that did not arrive on the mainframe. See Section 8.11: Audit Logs for more information.</p>	z/OS Systems Administrator
27.	<p>Create a distributed-systems file in the File Watcher directory structure.</p> <p>This is to verify that all components are working properly and that the corresponding tape file is cataloged on the mainframe.</p> <p>Create an unrecognized file name to confirm that it is moved into the hold directory.</p> <p>Use a mainframe program to read the cataloged tape file.</p> <p>This is to make sure the data was properly converted and formatted.</p> <p>*** Repeat the prior two steps as many times as necessary to verify that each type of file is correctly transferred to the mainframe.</p>	z/OS Systems Administrator

8.5 Data Conversions

Data conversions, such as converting ASCII data to EBCDIC are declared by a keyword and conversion name. The most common conversions of file data being transferred to the mainframe are one of the following:

- native ... no conversion is performed (default)

- ebcdic ... ASCII data is converted to EBCDIC
- custom/*name* ... a custom translation table is used

When defining a custom conversion, the name of the custom table is required. In most instances, the custom translation table is provided by the installation site and the site assigns a name to it. To declare a custom conversion, the keyword custom and the name is to be in lower case. Example: custom/thai_ebcdic

See [Section 9.8: Custom Translation Tables](#) for more information about custom translation tables.

8.6 DCB Exit

For each incoming distributed-systems file, a mainframe data set name must be assigned along with various mainframe data set characteristics (DCB parameters). These can be set via a DCB exit (written in REXX code) that runs under the PDQ started task, or via a DCB table that must be loaded onto the MDI Platform.

NOTE: If the DCB exit is not defined or it does not provide the DCB information, then the MDI Platform will perform a lookup in the most recently loaded DCB table for this queue name. If the incoming file name does not match any entry in the DCB table, then the file will be moved to the “hold” directory and not passed to the mainframe.

The following parameters are provided to the DCB exit.

- FULLNAME ... the full path and file name of the incoming file.
- USERID ... the mainframe user ID that is currently associated with the incoming file.
- WATCHER ... the configuration name of the file watcher that detected the file.
- SERVER ... the host name of the server which detected the file.

The DCB exit should provide the following information.

- DSN ... the mainframe data set name that is to be cataloged
- RECFM ... record/blocking format
- LRECL ... (maximum) logical record length
- BLKSIZE ... (maximum) block size or 0 (zero) to use system default
- CONVERSION ... data transformation technique that should be used
- EMAIL ... an address to which an email is sent when the incoming file is detected.

All of this information is required from the exit except CONVERSION and EMAIL.

If the DCB exit cannot determine what DSN or characteristics to use, then it should leave the DSN blank so that the MDI Platform will attempt to lookup the file name in the DCB table.

A sample DCB Exit is provided in [Appendix F: PDQ - DCB Exit Sample: DCXTYPE](#) and in the PDQ software package.

8.7 DCB Table

The DCBEXIT is the preferred method for defining distributed-systems files. However, for any distributed-systems file name not handled by the DCBEXIT program, that file name and its characteristics must be defined in a DCBTABLE file.

The DCBTABLE file can also contain global email notification settings for sending file-processed or file-error messages.

The active table resides on the MDI Platform. Typically, a source file is created on the mainframe and all settings are updated there. To update, run a mainframe job to copy from the source file to the MDI Platform.

The DCB table source file can be a simple sequential file on the mainframe. It may require a large LRECL value to accommodate entries that have many parameters. Mainframe file allocation settings of RECFM=VB LRECL=4000 BLKSIZE=27998 should be sufficient for most purposes.

8.7.1 Email Notification Settings

The DCB table provides global email notification settings regarding incoming files. There are two types of entries:

```
EMAIL_DISTRIBUTION=emailaddresses  
ERROR_DISTRIBUTION=emailaddresses
```

Using the EMAIL_DISTRIBUTION keyword, each time a successful transfer occurs, an email is sent to all email addresses associated with EMAIL_DISTRIBUTION, if defined, and those defined in the EMAIL= entry on the DCB entry line, if defined.

Using the ERROR_DISTRIBUTION keyword, each time an error occurs, an email is sent to all email addresses associated with ERROR_DISTRIBUTION.

Multiple mail addresses are to be comma delimited, with no spaces. For example:

```
ERROR_DISTRIBUTION=batchops@example.com,devteam@example.com
```

If the setting occurs more than once in the DCB table, only the last occurrence is in effect.

A sample DCB table file (with these keywords) is included in the PDQ software package.

8.7.2 DCB Parameters

The DCB table provides a secondary means to associate a file name to a set of attributes used to create the Luminex virtual tape and catalog the resulting tape. Each line of the DCB table defines a distributed-systems file (FILENAME=) and the associated parameters. Each DCB entry line will contain the following, delimited by semicolons. Of these parameters, only CONVERSION= and EMAIL= are optional.

- FILENAME= ... the case-sensitive name of the distributed-systems file
- DSN= ... the mainframe data set name that is to be cataloged
- RECFM= ... record/blocking format
- LRECL= ... (maximum) logical record length
- BLKSIZE= ... (maximum) block size or 0 (zero) to use mainframe system default
- CONVERSION= ... data transformation technique that should be used (default is native)
- EMAIL= ... an address to which an email is sent when the incoming file is detected.

Note that the specified FILENAME= is not specific to one user ID. Any user ID that has a file with this filename is processed using these settings.

Here is an example of how to specify the parameters for a given distributed-systems file name. All parameters for a given FILENAME= must be on the same line. Since this document cannot properly show that line, the examples below are wrapped.

```
FILENAME=mcp.warehouse.inventory.movement.txt;DSN=MCP.XMITAL.WHINV.MOVES;R
ECFM=VB;BLKSIZE=32000;LRECL=3200;CONVERSION=ebcdic;EMAIL=bartbillingsly@
example.com,charleschesterfield@example.com;dirkdrummond@example.com
```

```
FILENAME=Sales.SteveMilton.csv;DSN=RS.ORDER.NEW.SM112;RECFM=VB;BLKSIZE
=32760;LRECL=2100;CONVERSION=ebcdic;EMAIL=remotesales@example.com,smilton@
example.com
```

```
FILENAME=Sales.TimDiamond.csv;DSN=RS.ORDER.NEW.TD027;RECFM=VB;BLKSIZE=
32760;LRECL=2100;CONVERSION=ebcdic;EMAIL=remotesales@example.com,tdiamond@
example.com
```

```
FILENAME=damage.photo.jpeg;DSN=SHIP.ARCHIVE.PHOTO.DAMAGE.BIN;RECFM=U;B
LKSIZE=32760;LRECL=32760;CONVERSION=native;EMAIL=shipping.dept@example.com
```

8.8 PDQ Parameters

The PDQ started task has user configurable parameters that are kept in the dataset pointed to by the PARMFILE DD statement. A sample parameter file is shown below.

```
*****
*
* LUMINEX MDI -- PENDING DATA QUEUE
*
DSNPREF=LUMINEX.MDI.PDQ
*
*DDCMDPRINT=Y      Turn on the logging of operator command output
*DSNOPROF=A        Allow unprotected dataset names provided the
*                  LUMXPDQ.DSNOPROF profile is defined
*
&
*Devices associated with MDI server 1, defined as an MTL
DEV=9A00-9A3F STORCLAS=SC09A00
*
*Devices associated with MDI server 2, defined as non-SMS tape
```

```
DEV=9300.30
*
*****
```

- An asterisk in the first column of a statement designates the statement as a comment and it is ignored by PDQ.
- The default for most of the parameters is no, so commenting out a parameter statement is essentially specifying no for that statement.

8.8.1 DDCMDPRINT=Y

Specifies that in addition to the console PDQ writes responses to operator commands into the dataset pointed to by the CMDPRINT DD statement. Specifying yes for this parameter requires that there be an CMDPRINT DD statement in the JCL procedure for PDQ. It is recommended that it be a SYSOUT=* dataset, but if not, it should have RECM=VBA and LRECL=137.

8.8.2 DEV=devnum1[-devnum2] STORCLAS=mtl-storclass

The DEV= statement is used to specify the devices assigned to each MDI Platform. At least one DEV= statement is required. The device range can be specified in two ways. The first is xxxx-yyyy where xxxx is the first device in the range and yyyy is the last. The other way to specify this parameter is xxxx.yyy where xxxx is the first device in the range and yyy is the number of devices.

Optionally a STORCLAS=, DATACLAS=, or MGMTCLAS= parameter can be added to the DEV= statement. They specify the name of an SMS class used for storage management. This is used for MTL devices.

8.8.3 DSNPREF=cgx-communications-dsnprefix

DSNPREF= specifies a prefix used for the dynamically allocated dataset name used to connect to the MDI Platform.

The resulting MDI Platform communications data set name will be

dsnpref.stcid.ddname

DSNPREF is the prefix specified in the parameter described above, STCID is the unique started task ID assigned by the operating system to the PDQ started task (example STC01394) and DDNAME is the DDNAME used for the device (example COMM038F). The DDNAME is COMMxxxx where xxxx is the device number used for the communications connection to the MDI Platform. That device is chosen from the range of devices specified in the DEV= statement for the MDI Platform.

The DSNPREF parameter defaults to “LUMINEX.MDI.PDQ” if it is not specified.

8.9 PDQ Operator Commands

Since PDQ runs as a started task, PDQ commands are issued via the operator modify (F) command. The general syntax of the operator commands is

F started-task-name,command-and-arguments

Commands are case insensitive.

Command arguments are delimited by a single space. Double space ends the command.

The required “F started-task-name,” is always omitted from the commands below.

8.9.1 PSTOP

Initiates the shutdown of the PDQ started task.

The tasks that control the request tasks (what PDQ calls the CG Xtask) do not terminate until all the request tasks controlled by that CGX task terminate.

There is no indication sent to the MDI that PDQ has terminated.

8.9.2 DCXCODE=datasetname(member)

Loads and activates a new copy of the DCB EXIT (REXX) program.

Do not specify “(member)” if the file is not a member of a PDS.

8.9.3 DISABLE=devnum1[-devnum2]

Marks the specified devices as disabled (not-usable) for PDQ.

Only those devices declared to PDQ at start up by the DEV= statement are affected. This command does not add new (undeclared) devices to PDQ.

If the operator disables all the devices associated with a CGX all new work from that CGX is halted until such time as the operator re-enables at least two devices associated with that CGX. Two devices are needed because one is used for general communication with the CGX and the other is used to satisfy requests for new work.

PDQ disables devices on its own due to failures caused by some I/O error, lack of activity on the devices or other operating system problem. If all the devices for a particular CGX become disabled in this manner PDQ warns the operator of the situation so that corrective measures can be taken.

8.9.4 ENABLE=devnum1[-devnum2]

Marks the specified devices as enabled (usable) for PDQ.

Only those devices declared to PDQ at start up by the DEV= statement are affected. This command does not add new (undeclared) devices to PDQ.

8.9.5 LCG

Lists how many devices are enabled for PDQ usage for each MDI server (MDI Platform).


```
CG01: DEV TOTAL=16 ENABLED=16 DISABLED=0 CGSER=FFFFF11AC2110
CG02: DEV TOTAL=32 ENABLED=24 DISABLED=8 CGSER=FFFFFFA0AA065
CG03: DEV TOTAL=64 ENABLED=64 DISABLED=0 CGSER=FFFFFFA0AA265
CG04: DEV TOTAL=64 ENABLED=0 DISABLED=64 CGSER=FFFFFFA0AA665
```

CGnn ... is the Nth MDI Platform that was defined via the DEV= statements at start up.
TOTAL= ... is the number of DEV= devices associated with this MDI Platform.
ENABLED= ... is the number of devices that are enabled.
DISABLED= ... is the number of devices that are disabled.
CGSER= ... is the MDI Platform serial number reported in the NED.

PDQ needs at least 2 devices enabled on a MDI Platform or else it cannot receive incoming files.

8.9.6 LDEV

Lists each device (that was defined for use by PDQ) and its associated MDI Platform.

```
CG01: DEV=6600 ENA=Y USE=Y CGSER=FFFFF11AC2110 STORCLAS= MGMTCLAS= DATACLAS=
CG01: DEV=6601 ENA=Y USE=Y CGSER=FFFFF11AC2110 STORCLAS= MGMTCLAS= DATACLAS=
CG01: DEV=6602 ENA=Y USE=Y CGSER=FFFFF11AC2110 STORCLAS= MGMTCLAS= DATACLAS=
CG01: DEV=6603 ENA=Y USE=Y CGSER=FFFFF11AC2110 STORCLAS= MGMTCLAS= DATACLAS=
```

CGnn ... is the Nth MDI Platform that was defined via the DEV= statements at start up.
DEV= ... is the unit number used to allocate the device.
ENA= ... indicates whether the device is currently enabled for PDQ use.
USE= ... indicates whether the device is usable (online, not in use by any other job).
CGSER= ... is the MDI Platform serial number reported in the NED.
STORCLAS, DATACLAS, and MGMTCLAS ... show the corresponding value (if any) that was set when by the DEV= statement at PDQ start up. These fields will be blank for non-SMS managed devices.

8.9.7 LREQ

Lists the number of MDI File Watcher requests that are current in work on this PDQ started task for each MDI server (MDI Platform).

The output from the command is

```
CG01: CGSER=FFFFF11ACD310 NUMREQS=3

CG SECS DDNAME VOLSER STATUS DATASETNAME
01 1 RQ00992E PB5410 SENDING VOLSER RS.ORDER.NEW.SM112
01 2 RQ00992D PB5409 SENDING VOLSER SHIP.ARCHIVE.PHOTO.DAMAGE.BIN
01 1 RQ00992C PB5411 SENDING VOLSER RS.ORDER.NEW.TD027
```

The CG01: line is a header that is put out for each MDI Platform that PDQ is connected with.
The next line is the column headers for the rest of the output.

The CG column shows the CG ID that the request is associated with.
The SECS column shows the number of seconds that the request has been in this status.

The DDNAME column shows the DD name for the mounted tape for this request.
The VOLSER column shows the volume serial number for the mounted tape for this request.
The STATUS column shows a brief description of the status of the request.
The DATASETNAME column (not illustrated above) shows the DSN for the mounted tape file for this request

8.10 PDQ Messages

All PDQ messages start with the letters “PDQ”. For error messages those three letters are followed by a four-digit number that indicates to Luminex support the actual place in the code where the error occurred. That number is followed by a one-character indicator of the severity of the message. Those indicators are:

- ‘I’ – Informational message only.
- ‘W’ – Warning, an error has likely occurred, but the code was able to recover from it. Further investigation may be warranted.
- ‘E’ – Error, an error has definitely occurred and the code attempts to recover from it. Further investigation is definitely warranted, so looking at system error messages, SYSPRINT output or issuing PDQ operator commands may indicate what caused the problem.
- ‘S’ – A severe error has occurred that causes PDQ to terminate. Usually a dump file is produced but not always. In the case where there is no dump file looking at system error messages, SYSPRINT output or issuing PDQ operator commands may indicate what caused the problem. Contact LUMINEX Support for further help.

8.11 Audit Logs

The audit logs contain information on the processing of files. The audit logs indicate whether the process was successful, failed, or has been retried. The audit logs can be viewed in the Luminex Admin GUI or can be uploaded to the mainframe using the MAINT profile. Below is a screen capture from the GUI:

Date	Time	Host	TransactionID	Status	User	File	Size(Bytes)	DSN	Runtime(s)	Vol	DCB
2017-12-05	08:22:21	mdig8	151249089306836	Failure	ENGMG1	FILE02	17678336	TEST.PDQ.MTL.FILE02	0	UNK	FB/0/120
2017-12-05	08:28:40	mdig8	151249129534413	Success	ENGMG1	FILE01	17678336	TEST.PDQ.MTL.FILE01	25	Q04035	FB/0/120

The columns are defined as follows:

Date/Time	The time of completion
Host	The MDI Platform name
TransactionID	A unique identifier of the successful, failed, or retried process
Status	Will indicate: Failure, Success, or RetryXXX (more on this later)
User	The mainframe User ID
File	The file name

Size(Bytes)	the size of the file
DSN	The Dataset Name provided in the DCB table entry
Runtime(s)	the time of execution
Vol	The VOLSER on which the file resides
DCB	The DCB characteristic of the VOLSER. RECFM/Blocksize/LRECL

8.11.1 Additional Error Information

When an error occurs, information on the error is maintained on the MDI server. There is no additional information for both Success or RetryXXX status's. Details on the error can be obtained using the MAINT profile, or by clicking on the desired line in the Luminex ADMIN GUI. See *MDI MAINT Profile* for more information about the MAINT profile.

8.11.2 RetryXXX Messages

Upon some failures, a File Watcher slot releases the lock on the file. This allows another File Watcher slot on the current, or another server, to process the file again. The XXX portion of the message indicates the type of error encountered. They are defined below:

RetryCFG	A configuration error occurred parsing the File Watcher configuration file
RetryDCB	An error occurred accessing the DCB table
RetryLCK	The DCB table was “locked” for an extended time. It may be in the process of being updated, or another process failed to release it.
RetryKEY	An error occurred accessing the SecurityKey file needed to generate a Pass-Ticket
RetryTIC	An error occurred generating a Pass-Ticket
RetryTO	There was an error communicating with the PDQ started task. It is most likely not running. This is the most common retry-able error
RetryPDQ	An error was reported by PDQ. Most likely there was a problem cataloging the DSN.
RetrySIZ	The size of the processed file changed during processing. Increase the File Wait Time.

8.11.3 File Disposition

Upon Success, the incoming file is removed.

RetryXXX status leaves the file in its current location while the cause of the retry is determined and resolved.

Upon Failure, the incoming file is moved into the “hold” directory.

9. MDI Data Conversions

9.1 Introduction

The MDI product line includes options for data conversions that can be applied to data moving to and from the mainframe. The selection of the type of data conversion is specified in the JCL. A different conversion type can be applied to each file transferred.

9.2 Data Formats Supported Overview

Data transferred with MDI have both a block and record format associated with it. In some cases, the data may include a Block Descriptor Word (BDW) and Record Descriptor Word (RDW). MDI data conversions support the Fixed, Variable, and User Defined blocks. At this time, the use of Spanned Records is not supported.

In most instances, the JCL designer must have knowledge of the eventual use of the data. In some instances, data may not require conversion at all. In other instances, the Record Data Word (RDW) must be retained in the data for the application that will be utilizing the data, while in other cases it should be stripped. These actions are determined by the conversion name to be applied to the data.

Information to Collect and Consider for Data Conversions:

- Does the data need to be transformed, such as from ASCII to EBCDIC?
- How is the end of a line (or record) determined? Perhaps a line length is provided at the beginning of each line, or each line is terminated by a null x'00'. Or, perhaps there is a line feed character, or carriage return character, or some combination of these.
- What is the maximum number of bytes in a single line of data?
- What RECFM combinations are permitted by the mainframe program?
- What maximum LRECL is permitted by the mainframe program?
- Are there any other format restrictions imposed by the mainframe program?

9.3 Conversion Types

There are 3 types of conversions: built-in, iconv, and custom. The built-in conversions utilize Luminex tables that are built-in to the software. The iconv conversions utilize the Linux iconv library (more on that later in this document). The custom conversions utilize tables that can be placed on the MDI Platform. See [Section 9.8: Custom Translation Tables](#) for details on how to build custom conversion tables.

9.4 iconv Library

The Linux libiconv library can be used to perform many translations from one character set to another. The input and output character set names must be included in the JCL. This allows for a wide range of conversions. The combinations of input and output character sets is vast. It's important that the user verify the output closely. Luminex has no means to debug conversion issues with this library.

The JCL must indicate both the input and output conversion character sets (also called code pages). These are indicated on the same line as the conversion name. The arguments are: `icode=` and `ocode=` for input code page and output code page, respectively.

9.4.1 Tested Conversions

The following iconv conversions have been tested at Luminex and customer verified:

IBM1160 – EBCDIC-Thai
ISO-8859-11 – ASCII-Thai

9.5 Trailing Space Handling

The Luminex conversions perform special handling of trailing spaces at the end of an ASCII line or EBCDIC record. These are described below.

9.5.1 EBCDIC to ASCII Handling

When moving data from the mainframe and using an ASCII, custom, or iconv conversion, trailing spaces are removed from all lines. This is the default behavior regardless if the record format is Variable or Fixed. Native (or binary) or `strip_bdw` conversions do not apply.

To leave trailing spaces, use the optional argument of `nostrip` in the `ocode` (or `table`) keyword. For custom tables and iconv conversions, the `nostrip` parameter is added to the current value. For example, for native conversion such as `ascii_LF`, the `nostrip` option is defined as:

```
ocode=nostrip
```

For conversions that currently use `ocode` (or `table`), append the `nostrip` option:

```
ocode=mdi-ebc-ebc,nostrip
```

There shall be no spaces between these entries.

9.5.2 ASCII to EBCDIC Handling

When moving data to the mainframe, the following rules apply.

When converting to V or VB records using any EBCDIC conversion, an empty line on a file is converted to a space. This results in a 1-byte record (0 byte records are not allowed).

When converting to FB records, if necessary, EBCDIC spaces are added to lines to pad to the record size (LRECL).

9.6 Conversions Moving Data from the Mainframe

Data written from the mainframe to the MDI system has block and record format information associated with that data. The MDI data conversion utilizes this information when performing the conversion native.

If no conversion is specified, the native conversion is in effect. This conversion type works with Fixed, Variable, and Undefined record types. In this conversion, all potential RDW and BDWs

are left in the data. Additionally, there is no conversion of the data.

9.6.1 Native (or Binary)

This built-in conversion supports Variable, Fixed, and Undefined formats. The data is transferred without any conversion or modification. All BDW and RDW's are included in the data. The off-host application that may use this data must be BDW and RDW aware.

9.6.2 strip bdw

This built-in conversion supports only Variable formats. In this conversion, the BDW is removed from the data, no conversion of the data occurs. This type of conversion is common when transferring mainframe data to an off-host processor that understands the RDW and mainframe data.

9.6.3 strip all

This built in conversion supports only Variable formats. It is similar to strip_bdw except all RDW's are also removed.

9.6.4 ASCII

This built-in conversion converts EBCDIC data to ASCII. This conversion supports the Fixed and Variable blocks. Note that this conversion is rarely useful without the line feed and/or carriage return characters being added to the end of the record. NOTE: If EBCDIC data is received that does not have an ASCII equivalent, an ASCII space is put into the output. See the following ASCII conversions below.

9.6.5 ascii LF

This built-in conversion is the same as ASCII but adds a line feed character (hex value 0A) to the end of each record. This is of use when using the resulting ASCII file on Unix-type systems.

9.6.6 ascii CRLF

This built-in conversion is the same as ASCII but adds a carriage return and line feed character (hex values 0D/0A) to the end of each record. This is of use when using the resulting ASCII file on Windows systems.

9.6.7 iconv

This conversion will use the Linux iconv library.

9.6.8 iconv LF

This conversion, using the iconv library adds a linefeed at the end of each line. Note, this assumes that the output set supports the OD/OA values for linefeed and carriage returns. This is typically for files being viewed by Linux programs.

9.6.9 iconv CRLF

This conversion is the same as iconv_LF but adds a carriage return and line feed character (hex values 0D/0A) to the end of each record. These are typically required for files being viewed using Windows programs.

9.6.10 custom

This conversion utilizes a file with custom translation tables. The custom translation files reside on the MDI Platforms. Proven translations are included in the MDI release packages. Those that have been proven, and released, are described in this document. The custom translation file name is used as an argument in the JCL as well as the conversion name “custom”

9.6.11 custom LF

This conversion is the same as “custom” but a linefeed character is added the end of each record. This is typically for files being viewed by Linux programs.

9.6.12 custom CRLF

This conversion is the same as “custom” but adds a carriage return and line feed character (hex values 0D/0A) to the end of each record. This is of use when using the resulting ASCII file on Windows systems.

9.7 Conversions Moving Data to the Mainframe

Data read by the mainframe from the MDI system has block and record format information associated with that data. The MDI data conversion utilizes this information when performing the conversion.

9.7.1 Native (or Binary)

This built-in conversion type works with Fixed, Variable, and Undefined record types. When using Fixed Block and User Defined, no BDW or RDW data is added to the data. If required by the host application, it must exist in the original data. For Variable Block data, the BDW is inserted. It’s assumed that the original data contains the RDW information.

9.7.2 EBCDIC

This built-in conversion converts ASCII data to EBCDIC. This conversion works with Fixed and Variable block formats. When using Fixed Block, each ASCII line is padded with spaces to the defined record size. Any carriage returns and/or line feeds are removed from the data. For Variable Block, the data is converted and RDW and BDWs added to the data.

9.7.3 ebcdic NPC

This conversion is identical to EBCDIC except special print control characters are removed from the input file. Some ASCII reports may have this 0x0C character at the beginning of some lines. This character can cause lines to exceed the request LREC length.

9.7.4 iconv

The iconv library is used. As with the EBCDIC conversion above BDW and/or RDW information is added depending on the format. This conversion assumes that the input data is “ASCII-like” and removes linefeed and carriage returns from the data.

9.7.5 custom

This conversion takes an ASCII-type file and converts to EBCDIC using a custom translation. Carriage returns and linefeeds are removed. As with the EBCDIC conversion above BDW and/or

RDW information is added depending on the format.

9.8 Custom Translation Tables

Custom translation tables can be installed on the MDI Platform. The translation tables can be acquired from mainframe resident files, or codepages. The format used on the MDI Platform is identical to the mainframe codepages. The name of the custom table file is user-defined. The name of the file is used in the table=<file name> entry in the JCL.

The format of the table consists of 256 hexadecimal values. These values are output values for a hexadecimal input value. This value is used as an index into the table. A semi-colon can be used for comments. All values after the semi-colon are ignored. Below is a sample EBCDIC table whose input values are ASCII.

```
; ASCII-to-EBCDIC table for special report use
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;
00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F ; 00 ;
10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F ; 10 ;
40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61 ; 20 ;
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F ; 30 ;
7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6 ; 40 ;
D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D ; 50 ;
79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96 ; 60 ;
97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07 ; 70 ;
00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F ; 80 ;
10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F ; 90 ;
40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61 ; A0 ;
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F ; B0 ;
7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6 ; C0 ;
D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D ; D0 ;
79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96 ; E0 ;
97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07 ; F0 ;
```

Tables can be moved to the MDI Platform by the use of the MAINT profile. Tables can also be manually generated and moved to the MDI Platform by any of several other means.

The required location of the custom table file is /opt/luminex/MDITASK/current/etc. The file name is case-sensitive and must be specified exactly in the JCL.

9.9 JCL Parameter Examples

The table below contains many of the various combinations for conversion names and their associated parameters. This table does not contain all possibilities.

Transfer Type	Data Type	Strip Spaces	Mainframe Code Page	Server Code Page	Custom Conversion Table	Notes	JCL Parameters
PUT	ascii	Yes	-	-		-	conversion=ascii
PUT	ascii	Yes	-	-	-	Add CR/LF	conversion=ascii_CRLF
PUT	ascii	Yes	-	-	-	Add LF	conversion=ascii_LF
PUT	ascii	No	-	-	-	Add LF	conversion=ascii_LF;ocode=nostrip
PUT	ascii	No	-	-	-	Add CR/LF	conversion=ascii_CRLF;ocode=nostrip
PUT	ascii	No	-	-	-	Add LF	conversion=ascii_LF;ocode=nostrip
PUT	binary	No	-	-	-	No change to data	conversion=binary
PUT	binary	-	-	-	-	Remove BDW	conversion=strip_bdw
PUT	ascii	No	838	874	-	Add CR/LF	conversion=iconv_CRLF;icode=838;ocode=874
PUT	ascii	No	838	874	-	Add LF	conversion=iconv_LF;icode=838;ocode=974
PUT	ascii	No	-	-	mdi-ebc-asc		conversion=custom;table=mdi-ebc-asc
PUT	ascii	No	-	-	mdi-ebc-asc	Add CRLF	conversion=custom_CRLF;table=mdi-ebc-asc
PUT	ascii	No	-	-	mdi-ebc-asc	Add LF	conversion_LF=custom;table=mdi-ebc-asc
GET	ebcdic	-	-	-	-	Data must have RDW/BDW	conversion=binary
GET	ebcdic	-	-	-	-		conversion=ebcdic
GET	ebcdic	-	-	-	-	Remove Print Control	conversion=ebcdic_NPC
GET	ebcdic	-	838	874	-		conversion=iconv;icode=874;ocode=838
GET	ebcdic	-	-	-	mdi-asc-ebc		conversion=custom;table=mdi-asc-ebc

10. Syntax Rules for Parameters and KEYWORD=VALUE Pairs

Each MDI solution uses keywords to pass values to the profile on the MDI Platform.

These keywords and associated values must be coded using the proper syntax in the SYSIN DD card in the LUMXPROC batch JCL.

The LUMXPROC JCL must have the following minimum requirements:

1. Job Step with EXEC LUMXPROC,PROFILE=profile_name
2. DD statements in the JCL defining the data sets to be read (GET) or written (PUT)
3. SYSIN containing parms
4. -PARM keyword to indicate the beginning of the parameters
5. A -DD_<dd name> for each DD statement in the JCL.

10.1 The -PARM Parameter

The -PARM parameter is required. It is an indicator for the MDI profile on the MDI Platform that keywords and associated values are following. This parameter is followed by a space or equal (=) sign.

There are 2 types of information provided after the -PARM parameter:

1. KEYWORD=VALUE pairs to define specific parameters for the executing profile.
2. -DD_<dd name> parameters to describe the actions on a file associated with the DD statement.

10.2 KEYWORD=VALUE Pairs

Keywords are of the KEYWORD=VALUE format. Most keywords are unique to each MDI solution. Keywords can be coded in upper case, lower case or a mixture of upper and lower case.

KEYWORD=VALUE pairs can be delimited by a space or semi-colon.

Every KEYWORD= must have a value. KEYWORD=VALUE cannot have spaces before or after the equal sign (=).

Below are all valid syntax:

```
login=userid;password=pass
LOGIN=userid PASSWORD=pass
Login=userid; PassWord=pass
```

10.2.1 Values

Values are the variables entered after the equals (=) sign of the KEYWORD=VALUE pair.

Values that include file names and login credentials (user ID and Password) are case sensitive and must be coded as expected by the server.

Values that have spaces must be double quoted. As an example:

```
PREACTION="REMOVEFILE /a/b/c"
```

Values with multiple entries require double quotes and, in some cases, comma separators. As an example:

```
EMAILALL="alerts@luminex.com, user@example.com, support@example.com"
```

10.3 The -DD_ <dd name> Parameter

The DD_ <dd name> parameter defines the file name that is associated with the data set name in the JCL. The name of the DD statement is chosen by the JCL programmer. Below is a part of a DD statement in JCL:

```
//DOWNLOAD DD DISP=(OLD,DELETE),DSN=&F1DSN,
```

The file name is the name of the file that is transferred to or from the eventual destination. File names are case sensitive. If the file name contains spaces, the full file name must be enclosed by single quotes. File names with consecutive spaces are not allowed.

For example, the following file name is valid:

```
'file with spaces'
```

The following is not supported:

```
'file with spaces'
```

The following are all valid syntax:

```
-DD_TEST=testfile;  
-DD_TEST=testfile  
-DD_TEST='testfile'  
-DD_TEST='test file'
```

When multiple DD statements exist, multiple -DD_ <dd name> keywords are required, one for each data set name.

10.4 Special Handling of the CONVERSION Keyword

The CONVERSION= keyword is supported by all MDI solutions. It can be globally defined for all files if coded after the -PARM parameter. Alternatively, each file can be converted differently by including the CONVERSION=VALUE on each -DD_ <dd name> line.

For example, to globally define a conversion for all files, put the CONVERSION=VALUE on the

-PARM line:

```
-PARM CONVERSION=ascii_LF;  
-DD_UPLOAD=uploadfile;  
-DD_DOWNLOAD=downloadfile;
```

Files associated with one or more -DD_<dd name> parameters are converted using the conversion of ascii_LF,

To specify a different conversion for each file, add the CONVERSION=VALUE on each -DD_<dd name> line:

```
-DD_UPLOAD=uploadfile; CONVERSION=ebcdic  
-DD_DOWNLOAD=downloadfile; CONVERSION=ascii_LF;
```

Comments

Lines in the SYSIN DD card can be commented by keying an asterisk (*) in column 1. As an example:

```
//SYSIN DD *  
-PARMS  
Login=mike  
Password=pass  
preaction="listfilel /a/b/c"  
* -PARMS preaction="listfilel /a/b/c"  
-DD_UPLOAD=/a/b/c  
  oformat=ascii_LF  
/*
```

10.5 Keyword Substitution for File Names

There are several keywords that will cause a substitution or addition of a value into the file name. The # symbol denotes a keyword. The available keywords are:

```
#MDY  
#YDM  
#DSN
```

10.5.1 Keyword #MDY

When detected, the current date of the format <month><day><year> will be used. For example, if the date is June 3, 2019, the substitution will be 060319.

10.5.2 Keyword #YDM

Similar to the #MDY keyword substitution, the current date in the format of <year><month><day> will be substituted.

10.5.3 Keyword #DSN

The #DSN keyword will be substituted by the full DSN of the dataset.

10.5.4 Examples

Below are some examples of keyword substitution where the current date is June 3, 2019, and the full DSN is TEST.MDIDATA.ENG.FILE1:

```
-DD_FILE1=#DSN      -> -DD_FILE1=TEST.MDIDATA.ENG.FILE1
-DD_FILE1=data/file.#MDY      -> -DD_FILE1=data/file.060319
PREACTION="listlist data/file.#MDY -> PREACTION="listfile1 data/file.060319"
```

Combinations of keywords is also possible:

```
-DD_FILE1=datadir/#DSN.#YDM -> -DD_FILE=datadir/TEST.MDIDATA.ENG.FILE1.190306
```

Note: Keyword #DSN does not work in the PREACTION and POSTACTION commands.

10.6 Additional Syntax Rules

- VALUE's that consist of multiple words must be double quoted. See "preaction" above.
- Lines should be kept short for readability. For readability, it is recommended that parameters start in column 1. No line should run past column 71 or it may not be properly evaluated. KEYWORD=VALUE pairs can be placed on separate lines keeping in mind words cannot be split.
- A ++ can be coded as the last characters on a line to indicate the next line is a continuation. The lines are merged with no imbedded spaces. If a space is required, it should be coded on the first line before the ++. The continuation line must begin in column 1.

Below is a complex example of valid syntax:

```
000054 -PARM destination=g7ftp10g parallel=1
000055 login=programmer
000056 password=password
000057 emailall="users@company.com"
000058 preaction="LISTFILEL
000060 'test print133 data'"
000061 postaction="listfile1
000062 'test print133 data'"
000063 -DD_FBADOWN=""'test print133 data'"
000064 conversion=ascii_CRLF
000065 ocode=nostrip
000066 -DD_FBAUP=
000067 ""'test print133 data'" conversion=ebcdic_npc
```

10.6.1 Multi-Word Values

VALUES that include spaces must be enclosed in single quotes.

Example:

```
-DD_FBADOWN='test print133 data'
```

10.6.2 Lines Extending Past Column 71

Lines should be kept short for readability. It is recommended that parameters start in column 1 and must not run past column 71. Coding past column 71 may cause syntax errors or erroneous results.

If a line needs to extend past column 71, a ++ can be coded as the last characters to indicate the next line is a continuation.

The lines are merged with no embedded spaces. If a space is required, it should be coded on the first line before the ++.

The continuation line must begin in column 1.

Example:

```
*---+---1---+---2---+---3---+---4---+---5---+---6---+---7---+---8
-DD_COPYFILE='very.long.filename with several directories  /++
continuing/on.second.line/++
and.a.third  ++
with embedded spaces'
```

10.7 Examples

The SAMPLIB partitioned data set (PDS) provided with your MDI solution contains JCL examples that include a variety of KEYWORDS and VALUES for the -PARM and -DD_ parameters in the SYSIN DD card. Please refer to this PDS for more examples.

11. MDI SecureTransfer (MDI:ST)

11.1 Introduction

MDI:ST is a Managed File Transfer solution that appears as a virtual tape subsystem to the mainframe. Therefore, the MDI Platform is physically connected to the mainframe via the fiber connection (FICON) and added to the mainframe environment via a mainframe process of defining logical paths to devices, commonly referred to as an IOGEN, on the mainframe system. A tape range is defined and added to the tape management system.

MDI:ST includes a Mainframe Batch Interface. This interface consists of a program, called LUMXPROC, installed on the mainframe and used to direct MDI:ST where to move the data. To move data from the mainframe to the MDI Platform and subsequently to its destination, a batch job is executed using a utility, such as ICEGENER, to write the data to a virtual tape defined for use with the MDI Platform. The second step of the batch job executes LUMXPROC to direct MDI:ST to encrypt the data, using SFTP, optionally transform the data from EBCDIC to ASCII, and move it to the destination. The destination IP address, Server Name or DNS name, file name, as well as the server sign-on credentials are also part of the LUMXPROC communications.

11.2 SecureTransfer Transfer Methods

11.2.1 Transferring Data from the Mainframe – PUT

To move data from the mainframe to an internal or external IP address, server name or DNS name, a batch job is executed on the mainframe to direct MDI:ST to do a PUT action to the destination. The first step of the batch job copies the file to be transferred to an MDI:ST owned virtual tape. The second step executes LUMXPROC to communicate instructions to the MDI Platform including; where to send the data and how to transform the data, if needed. The security system (SAF interface) on the mainframe, controls the use of the LUMXPROC. Security profiles can be put in place to designate which user IDs have the ability to do PUT or GET actions or BOTH.

11.2.2 Transferring Data from a Non-Mainframe Platform – GET

There are two methods that can be used to move data from a non-mainframe Platform to the mainframe (GET). The first method is to execute LUMXPROC, to direct MDI:ST to pull the file to the mainframe. The second method is the Push Method described in the next section. In a GET operation, the execution of LUMXPROC results in a virtual tape being created and cataloged on the mainframe. The LUMXPROC step communicates tape formatting properties, and data set name information. The security system (SAF interface) on the mainframe, controls the use of the LUMXPROC usage and authorizes the cataloging of the output file. Once the data set is cataloged on the system, it's available to be used, copied to a disk data set, or deleted, as with any other data set.

11.2.3 Transferring Data from a Non-Mainframe Platform – Push Method

The second method that can be used to move data from a non-mainframe Platform to the mainframe, available with MDI SecureTransfer V2, are the Pending Data Queue (PDQ) and File Watch features. PDQ is a started task deployed on the mainframe to communicate with MDI:ST

via the FICON channel. To ensure only authorized users are sending data to the mainframe, a Security PassTicket is used to authenticate the user ID. Once authorized, PDQ can mount a scratch tape and catalog the file on the mainframe. To use the Push Method, PDQ and File Watch must be installed and configured. See [Section 8: MDI File Watcher and Pending Data Queue \(PDQ\)](#) for more information about this feature.

11.3 SecureTransfer Components

11.3.1 Mainframe

- LUMXPROC Load module
- LUMXPDQ, LUMXPDQC, LUMXPDQR, LUMXPDQU, LUMXPDQV Load modules
- LUMXPROC Procedure
- LUMXPDQ Procedure (started task)
- Batch JCL
- SAF Interface

11.3.2 MDI Platform

- Server Code
- MDI:ST Software
- Client's customized profile definitions
- File Watcher
- DCB Table

11.3.3 Storage

- Storage available on the MDI Platform (limited)
- Client provided Open Systems storage
- Optionally, Open Systems storage purchased with MDI SecureTransfer

11.3.4 MDI Reporting Interface

For reporting, a Graphical User Interface (GUI), is provided with MDI:ST. The GUI provides a dashboard of various reports including:

- Transmissions in Process
- Performance Reports
- Network Traffic
- MDI Put Bytes Transferred
- MDI Get Bytes Transferred
- Storage Consumption and Availability

Also available via the GUI is the ability to generate reports that show trending over a period of time:

- Performance Reports
- MDI Put Time
- MDI Get Time

- MDI Put Bytes Transferred
- MDI Get Bytes Transferred
- Storage Consumption and Availability

11.4 SecureTransfer Planning Overview

The software components for the MDI Solutions provide for communication over the FICON channel between the mainframe and the MDI Platform. The MDI Platform appears as a tape device to the mainframe, allowing for the transfer of mainframe data simply by writing data to MDI owned tape. For communication and data movement over FICON to occur, the batch interface components must be installed on the mainframe. MDI product code (SecureTransfer profile) is also installed and configured for your site on the MDI Platform by the Luminex Support team.

The following table describes the tasks necessary to begin using MDI SecureTransfer in your environment:

	Task:	Refer to Section in this Guide:
1	Product Installation on the mainframe	Section 6: MDI Mainframe Software Installation and Configuration
2	Setup and execution of LSCRUP scratch update	Appendix C: LSCRUP Scratch List Update Utility
3	Understanding and configuring Security Requirements	Section 6: MDI Mainframe Software Installation and Configuration
4	Configuring File Watch and PDQ	Section 8: MDI File Watcher and Pending Data Queue (PDQ)
5	SecureTransfer Profile Configuration	Section 4: Planning for MDI ; Section 5: MDI Profile Creation and the Installation Workbook ; Section 11.5: Profile Configuration
6	Understanding Data Conversion Options	Section 9: MDI Data Conversions
7	Understanding JCL Syntax Rules	Section 10: Syntax Rules for Parameters and KEYWORD=VALUE Pairs
8	Creating the JCL	Section 11.5.1: SecureTransfer JCL
9	Executing a test	Section 11.10: Executing a Test Using SecureTransfer
10	Converting existing JCL	Section 11.12: Migration/JCL Conversion/Professional Services
11	Interpreting Messages and Codes	Appendix A: Message Codes

11.5 Profile Configuration

When a SecureTransfer profile is configured, operational arguments such as user name, user password, host name or IP, can be configured in the profile or provided by the mainframe JCL. When operational arguments are configured in the profile, the same arguments must not be coded in the JCL. When operational arguments are found both in the JCL and in the SecureTransfer profile on the MDI Platform, an error is returned indicating that the argument(s) must be re-

moved from the JCL.

Providing operational arguments in the JCL provides flexibility in the client's environment; several users can use the same profile with a variety of options. However, if the client's goal is to mandate certain operational arguments, hard-coding options in the SecureTransfer profile on the MDI Platform achieves that goal.

[Section 11.6: JCL Parameters and KEYWORD=VALUE Pairs](#) describes the parameters specific to the SecureTransfer profile.

11.5.1 SecureTransfer JCL

MDI:ST uses JCL on the mainframe to copy the file to the MDI:ST Platform, via virtual tape, and transmit the file to a destination. The JCL typically contains a two-step process. Step 1 is copying the data set to be transferred from disk to MDI owned virtual tape. It's recommended that a copy utility, such as ICEGENER or if you are licensed for SYNCSORT, SYNCGENER be used in place of IEBGENER. These recommended copy utilities use buffering to execute faster and use less system overhead.

Step 2 Executes the LUMXPROC program passing parameters coded in the SYSIN DD card to the MDI Platform. If the data to be transferred has already been copied to MDI owned virtual tape, Step 1 can be eliminated.

[Section 6: MDI Mainframe Software Installation and Configuration](#) contains general information about installing the LUMXPROC program that the JCL executes.

Note: ICEGENER or SYNCGENR (if licensed) should be used in place of IEBGENER for better performance and better use of system resources.

[Section 10: Syntax Rules for Parameters and KEYWORD=VALUE Pairs](#) describe general syntax rules for coding parameters and keywords/value pairs in the SYSIN DD card in the JCL. Please review this section carefully as many of the values provided in the JCL must be coded in lower case or mixed case, as they are transferred to distributed platforms for processing. **It is recommended that the CAPS OFF command be issued in your ISPF editor when working with MDI related JCL to prevent lower case or mixed case values being automatically converted to upper case.**

11.6 JCL Parameters and KEYWORD=VALUE Pairs

Parameters specific to the SecureTransfer profile are designated in the SYSIN DD section of the JCL.

To successfully execute a file transfer using the SecureTransfer batch interface (LUMXPROC) the following information must be provided either in the profile definition on the MDI Platform (previously configured by Luminex Support team) or in the LUMXPROC step of the JCL.

- Login user ID and password to sign on to the destination server

- Destination IP address, DNS name or server name
- The type of encryption cipher to be used if other than the default (must be supported by the destination server)
- Optionally, pre-actions can be performed on the destination server such as “make directory”
- Optionally, post-actions can be performed after the file transfer such as “list file”
- Optionally, converting the data from EBCDIC to ASCII or ASCII to EBCDIC with the oformat or conversion keywords. More information on converting mainframe data can be found in [Section 9: MDI Data Conversions](#).

11.6.1 The -PARM Parameter

The **-PARM parameter is required in the JCL**. It is an indicator for the MDI SecureTransfer profile configured on the MDI Platform that keywords and associated values are following.

The -PARM indicator is followed by a space or equal (=) sign.

There are 2 types of information provided after the -PARM parameter:

- 1) KEYWORD=VALUE pairs to define specific parameters for the SecureTransfer profile.
- 2) -DD_<dd name>= parameter describes the actions specific to the file associated with the DD statement.

11.6.2 -DD_<DD name> Parameter

The -DD_<DD name> parameter provides the file name on the server for PUT operations or on the mainframe for GET operations. This file name provides input or output for a data transfer. The filename immediately follows the -DD_<DD name>= value. A -DD_<DD name>= value is required for each DD statement in the JCL. This defines either the input or output file name. For example, if two files are to be transferred, the DD statements associated with the transfers could be:

```
//COPYFIL1 DD DISP=OLD,DSN=PROD.MASTER.FILE1.MDI,
//          UNIT=&OUTDEV
//*
//COPYFIL2 DD DISP=OLD,DSN=PROD.MASTER.FILE2.MDI,
//          UNIT=&OUTDEV
```

The associated -DD_ values are:

```
-DD_COPYFIL1=syslog_020316
-DD_COPYFIL2=report.txt
```

Directory paths must be included with the filename.

Each user, whose user name is the login name, has a “home” directory on the SFTP host. If only the file name is included in the -DD_<DD name>=value, then that file is transferred to or from the user’s home directory.

If a directory is included with the file name, there are several syntaxes that specify whether the directory is a sub-directory off the user’s home directory, or an explicitly defined directory. For example, if the user’s home directory is /home/Bob, the following transfers the file, report.txt, to

or from /home/Bob:

```
-DD_COPYFILE1=report.txt
```

The following copies report.txt into a sub-directory, /pub, off /home/Bob. Note that there is no leading / in front of pub.

```
-DD_COPYFILE1=pub/report.txt
```

If the user has permissions to access other directories on the SFTP host, the following syntax can be used. As an example, the user has permissions to access /Documents/reports. To transfer the file, report.txt, to or from /Documents/reports:

```
-DD_COPYFILE1=/Documents/reports/report.txt
```

Note that there is a / before Documents. This implies an explicitly defined directory that is not associated with the user's home directory. The user must have permissions to access this directory.

Sub-directories are not automatically created. If a sub-directory is to be created, the use of the preaction parameter is required. For example, to create a sub-directory, mysubdir, off the user's home directory:

```
preaction="MAKEDIR mysubdir"
```

To create a sub-directory, /myfiles, off /Document/pub:

```
preaction="MAKEDIR /Document/pub/myfiles"
```

Note: If multiple layers of sub-directories are to be created, MAKEDIRP is required. For example, to create the sub-directory, /myfiles/092216, off the user's home directory:

```
preaction="MAKEDIRP myfiles/092216"
```

11.6.2.1 Keywords & Values

All parameters are specified in the NAME=VALUE format, where NAME is a keyword that defines the VALUE.

Most keywords are unique to each MDI solution. Keywords can be coded in upper case, lower case or a mixture of upper and lower case. KEYWORD=VALUE pairs can be delimited by a space or semi-colon.

Every KEYWORD= must have a value. KEYWORD=VALUE cannot have spaces before or after the equal sign (=).

Below are all valid syntax:

```
login=userid;password=pass
LOGIN=userid PASSWORD=pass
Login=userid; PassWord=pass
```

Values

Values are the variables entered after the equals (=) sign of the KEYWORD=VALUE pair.

Values that have spaces must be double quoted. As an example:

```
PREACTION="REMOVEFILE /a/b/c"
```

Values with multiple entries require double quotes and, in some cases, comma separators. As an example:

11.6.2.2 The File Name Value in the DD_<dd name>= Parameter

The file name is the name of the file being transferred to the destination server or from a server to the mainframe. **File names are case sensitive.** If the file name contains spaces, the full file name must be enclosed by single quotes. File names with consecutive spaces are not allowed. For example, the following file name is valid:

```
'file with spaces'
```

The following is *not* supported:

```
'file   with   spaces'
```

The following are valid syntax:

```
-DD_TEST=testfile;
-DD_TEST=testfile
-DD_TEST='testfile'
-DD_TEST='test file'
```

When multiple DD statements exist in the JCL, multiple -DD_ <dd name>=value keywords are required, one for each data set name.

11.6.2.3 Special Handling of Conversion Keyword

The CONVERSION= value keyword is supported by all MDI solutions. It can be globally defined for all files if coded after the -PARM parameter. Alternatively, each file can be converted differently by including this KEYWORD=VALUE on the -DD_<dd name> line.

For example, to globally define a conversion for all files, put the KEYWORD=VALUE on the -PARM line:

```
-PARM CONVERSION=ascii_LF;
-DD_UPLOAD=uploadfile;
-DD_DOWNLOAD=downloadfile;
```

Files associated with one or more -DD_<dd name>= parameters are converted using the conver-

sion of ascii_LF,

To specify a different conversion for each file, define on the -DD_<dd name>= line:

```
-DD_UPLOAD=uploadfile; CONVERSION=ebcdic  
-DD_DOWNLOAD=downloadfile; CONVERSION=ascii_LF;
```

11.6.2.4 Comments

Lines in the SYSIN DD card can be commented by keying an asterisk (*) in column 1. In the example below, the preactions keyword will not be executed.

```
//SYSIN DD *  
-PARMS  
Login=mike  
Password=pass  
* preaction="listfile1 /a/b/c"  
-DD_UPLOAD=/a/b/c  
oformat=ascii_LF
```

11.6.2.5 More Syntax Rules

Values that contain multiple words must be double quoted. An example of this is shown in the “preaction” section below.

Lines should be kept short for readability. No line should run past column 71 or it may not be properly evaluated. KEYWORD=VALUE’s can be placed on separate lines keeping in mind words cannot be split. Below is a complex example of valid syntax:

```
000054 -PARM destination=g7ftp10g parallel=1  
000055 login=programmer  
000056 password=password  
000057 emailall="users@example.com"  
000058 preaction="LISTFILEL testfile, LISTFILEL  
000060 'test print133 data'"  
000061 postaction="listfile1  
000062 'test print133 data'"  
000063 -DD_FBADOWN="'test print133 data'"  
000064 conversion=ascii_CRLF  
000065 ocode=nostrip  
000066 -DD_FBAUP=  
000067 "'test print133 data'" conversion=ebcdic_npc
```

11.6.2.6 Examples

The SAMPLIB partitioned data set (PDS) provided with your MDI solution contains JCL examples that include a variety of KEYWORDS and VALUES for the -PARM and -DD_ parameters in the SYSIN DD card. Please refer to this PDS for more examples.

11.7 SecureTransfer Parameters

11.7.1 destination

This mandatory parameter can be defined in the JCL or profile configuration on the MDI Platform.

The destination indicates the IP address, server name or DNS name of the SFTP server where the data is to be sent. If using a server name, MDI can use DNS, or entry in the hosts file, to associate a name to an IP address. The format of this parameter is:

```
destination=<host name/IP>
```

For example, to transfer files between the mainframe and the SFTP host server name labbox1:

```
destination=labbox1
```

11.7.2 login

This mandatory parameter can be defined in the JCL or profile configuration on the MDI Platform.

This is the username, or login, that will be used to access the SFTP host. This login, and its associated password, must be previously defined on the SFTP host. The format of this parameter is:

```
login=<user name>
```

For example, a user name of User1:

```
login=User1
```

When username and password is defined on the SFTP host, a home directory in which files can be transferred is also defined. The ability to transfer files to or from other directories, must be configured by the SFTP server administrator.

11.7.3 password

This mandatory parameter can be defined in the JCL or in the profile configuration on the MDI Platform.

This is the password that is associated with the login, both of which is used to access the SFTP host. This login and password must be previously defined on the SFTP host. The format of this parameter is:

```
password=<password>
```

For example, for a password of secret1:

```
password=secret1
```

This setting is mandatory in the JCL or Profile.

NOTE: Even when SSH keys are exchanged between the MDI Platform and the destination SFTP host, a password still needs to be defined, even if not used. For example:

```
password=none  
password=dummy
```

The login and/or password can be defined in a security protected data set. For example, the partitioned data set USER1.MDI.PARMLIB(LOGIN) contains:

```
Login=User1;password=secret1
```

Below is a section of the JCL using a PARMLIB to hold login information:

```
000020 //SYSIN DD *  
000021 -PARM  
000022 //      DD DISP=SHR,DSN=USER1.MDI.PARMLIB(LOGIN)  
000023 //      DD *  
000024 destination=sftpghost2
```

11.7.4 cipher

This optional parameter can be defined in the JCL or profile configuration on the MDI Platform.

This parameter sets the encryption cipher used during the SFTP transfer. If the cipher is not specified, the MDI Platform negotiates with the SFTP host to determine the best cipher method. Not all ciphers are supported by all SFTP hosts. It is recommended that any specified cipher be tested with Luminex support prior to implementation.

Some common ciphers are listed below. They are listed in order of highest to lowest performance. Each has different encryption algorithms. The user must determine which cipher satisfies their desired performance and security requirements. The SFTP host must also support the specified cipher.

- arcfour
- blowfish
- aes128-ctr
- aes192-ctr
- cast128-cbc
- 3des-cbc
- aes256-ctr

11.7.5 maxretries

This optional parameter can be defined in the JCL or profile configuration on the MDI Platform.

This setting determines the number of retries of failed transfers to perform. The SFTP protocol performs two (2) retries by default. The maxretries setting allows the client to set a higher or lower number of retries. This setting, along with the delay_time setting, determines that amount of time a network can be down before an error is declared.

Note, errors associated with failed login and/or passwords, and with failed permissions, are not retried and an error is declared immediately.

11.7.6 delay_time

This optional parameter can be defined in the JCL or profile configuration on the MDI Platform.

This value is used in conjunction with maxretries. It defines the time, in seconds, between retries. The default is 10 seconds.

11.7.7 sshopts

This optional parameter can be defined in the JCL or profile configuration on the MDI Platform.

Some network configurations will require additional options for ssh. For example, a broadcast IP may be required. The values of this parameter will be directly inserted into the lftp argument line. For example, to set a broadcast IP of 10.20.30.4:

```
sshopts="-b 10.20.30.4"
```

11.7.8 logdir

This optional parameter can be defined in the JCL or profile configuration on the MDI Platform.

This parameter enables the feature and defines a location on the SFTP host in which log files are transferred. The log file contains a single line of information indicating success or failure. The filename is of the format:

```
<filename>-<Transaction ID>
```

Where the Transaction ID is a unique number associated with each job. For example, to copy log files to: /home/logs:

```
LOGDIR=/home/logs
```

The SFTP host administrator is responsible for making the LOGDIR directory accessible by all SFTP users defined in the JCL or profile configuration on the MDI Platform.

It is possible to declare a different user/password on the LOGDIR value. For example, for the user Bob, with the password Secret1, and a log directory of /home/logs, the LOGDIR entry is:

```
LOGDIR=/home/logs:Bob:Secret1
```

11.7.9 preaction/postaction

This optional parameter can be defined in the JCL or Profile configuration on the MDI Platform.

The preaction parameter allows for a command to be executed on the SFTP host. This action is performed before any file transfer. The postaction is run after all files have been transferred. For example, to get a listing of all files whose names begin with "sys" in the defined directory, one could enter the following:

```
preaction="LISTFILEL sys*"
```


The available keywords are described below.

LISTFILE – Performs a short listing of the file(s). No error if the file does not exist.
LISTFILEL – Performs a long listing of the file(s). No error if the file does not exist.
TESTFILE – Test for presence of the file. Job fails if file does not exist.
TESTDIR – Tests for presence of a directory. Job fails if the directory does not exist.
REMOVEFILE – Deletes the specified file. No error if the file does not exist.
MAKEDIR – Creates the specified directory. Job fails if the directory cannot be created.
MAKEDIRP – Creates the specified directory, including non-existent sub-directories.
REMOVEDIR – Removes the specified directory. No error if directory does not exist.
MOVEFILE – Moves the specified file to another directory. Job fails if the operation fails.

Multiple actions can be performed. When multiple actions are coded in a list, each action requires a semi-colon (;) delimiter as shown below:

```
preaction="LISTFILEL sys*; LISTFILEL file*;LISTFILEL **"
```

Contact Luminex for more details on the available actions that can be performed.

11.7.10 email_error_list

This optional parameter can only be defined in the profile configuration on the MDI Platform.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output

11.7.11 email_list

This optional parameter can only be defined in the profile configuration on the MDI Platform.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output.

11.7.12 emailonerror

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output. Multiple emails must be comma separated and double quoted. For example:

```
emailonerror="errors@example.com,alerts@luminex.com"
```

11.7.13 emailall

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receive an email of all job results. The body of the email contains identical information as the job SYSTPRINT output. Multiple emails must be comma

separated and double quoted. For example:

```
emailall="user1@example.com,alerts@luminex.com"
```

```
EMAILALL="alerts@luminex.com, user@example.com, support@example.com"
```

11.7.14 Handling Emails

Emails can be defined in both JCL and the profile configuration on the MDI Platform. If defined in both places, the lists are merged and duplicates removed. If an error occurs, the email addresses defined in: `emailonerror` (JCL), `emailall` (JCL), `email_error_list` (configuration), and `email_list` (configuration) are merged and duplicates removed.

11.7.15 oformat Keyword

This keyword, and the synonym keyword “conversion”, are used to define a conversion name. Each conversion name has a set of properties associated with it. Conversion names are defined in the “MDI Data Conversions” manual. There are 3 sets of conversion types: built in, iconv, and custom. Each can have different additional arguments.

The conversion keyword is used when transferring data to and from the mainframe. Commonly used conversion names are: `ascii`, `ascii_LF`, and `ebcdic`. The conversion keyword is defined on the `-DD` line. For example, to convert a mainframe file to ASCII with added linefeeds:

```
-DD_COPYFILE1=pub/report.txt oformat=ascii_LF
or
-DD_COPYFILE1=pub/report.txt conversion=ascii_LF
```

Some special conversions such as `iconv` and `custom`, require additional arguments. When using `offormat=iconv`, the following are also required:

```
icode=<character set name>
ocode=<character set name>
```

The `icode` parameter defined the type of file that is used as input.

The `ocode` parameter (or synonymous keyword “table”) defines the type of file the output should be. Both are required.

The `iconv` library contains a number of conversions, most of which require testing. It is recommended that Luminex be contacted to perform testing on the desired file and conversion formats before deploying.

The `offormat=custom` conversion allows for use of custom translation tables. The table defines the format of the output file. These tables reside on the MDI Platform in the form of files. To implement the custom table, the file name is declared in the `ocode=<file name>` entry. For example, to use the translation table of: `ascii_korean`, the `-DD` entry is:

-DD_COPYFILE=report.txt oformat=custom ocode=ascii_korean
or
-DD_COPYFILE=report.txt conversion=custom table=ascii_korean

Tables can be provided by the client and installed on the MDI Platform with the assistance of Luminex support.

By default, all PUT conversions, except binary, remove EBCDIC spaces at the end of lines. To disable this, the “nostrip” option can be use in the ocode/table keyword. For built-in conversions, ascii and ascii_LF, for example, the ocode value is:

```
ocode=nostrip
```

Other conversion types that currently use the ocode/table value will require the “nostrip” to be added to the current value. For example:

```
ocode=UTF8,nostrip
```

11.8 Sample JCL – PUT

The following JCL is a simple example that transfers a single file to an SFTP host.

Description of the important parameters is described below.

```
1 //MDITAPE JOB (ACCT),'SFTP NEW DATASET',CLASS=H,
2 // CLASS=A,NOTIFY=&SYSUID
3 /**
4 /** ICEGENER COPIES A DATASET TO MDI/CGX TAPE
5 /**
6 /**
7 // SET OUTDEV=MDITAPE UNITNAME FOR MDI CGX CONTROLLER
8 // JCLLIB ORDER=(LUMENG.MDI.JCL) JCLLIB CONTAINING LUMXPROC
9 /**
10 /**
11 //STEP010 EXEC PGM=ICEGENER
12 /**
13 /** SYSUT1 INPUT FILE CAN BE ON DISK OR ANY TAPE
14 /** SYSUT2 OUTPUT MUST BE MDI/CGX TAPE
15 /**
16 //SYSUT1 DD DISP=OLD,DSN=PROD.MASTER.FILE INPUT MASTER FILE
17 //SYSUT2 DD DSN=PROD.MASTER.FILE.MDI, OUTPUT COPY
18 // DCB=*.SYSUT1, USE SAME DCB AS INPUT
19 // VOL=(,RETAIN), DON'T DISMOUNT TAPE
20 // DISP=(NEW,CATLG),
21 // UNIT=&OUTDEV MDI OUTPUT UNIT MUST BE CODED
22 //SYSIN DD DUMMY
23 //SYSPRINT DD SYSOUT=*
24 /**
25 /**
26 /** USE MDI TO TRANSFER FILE THE TAPE JUST CREATED
27 /** INPUT FILE IS THE FILE JUST CREATED
28 /** OUTPUT FILE ON THE SERVER IS PROD.MASTER.FILE.NEW
29 /** Profile must be defined to RACF
```

```

30 // SET PROFILE=SFTP2HOSTA PROFILE IS PREDEFINED TO PROCESS FILE
31 /**
32 //STEP020 EXEC LUMXPROC,PROFILE=&PROFILE
33 /**
34 //COPYFILE DD DISP=OLD,DSN=PROD.MASTER.FILE.MDI,
35 // UNIT=&OUTDEV MUST SPECIFY MDI UNIT
36 //SYSIN DD *
37 -PARM destination=labbox1;login=User1;password=secret1;
38 postaction="LISTFILEL prod.master.file.new";
39 -DD_COPYFILE=prod.master.file.new
40 /**
41 /**

```

11.8.1 Dataset Transfer to Virtual Tape

In the above JCL example line 11 ICEGENER creates the output tape. Multiple application programs or copy utilities are supported. It's recommended that high-speed copy utilities, such as ICEGENER, be deployed to copy the dataset to a tape.

11.8.2 LUMXPROC

In the example above line 32 shows the execution of the LUMXPROC program. This step must follow the transfer of the dataset(s) to the MDI Platform. This step communicates information to the SecureTransfer profile on the MDI Platform to instruct SecureTransfer what to do with the data set being transferred.

11.8.3 PROFILE

Line 32 also includes the SecureTransfer profile name. In this case the name is SFTP2HOSTA. This SecureTransfer profile must be pre-defined and configured on the MDI Platform.

11.8.4 DD Name

Line 34 defines the DD name and associated dataset parameters. The DD name of COPYFILE is important and is utilized later in the SYSIN to associate a file name to the dataset.

11.8.5 SYSIN

Line 36 is the start of the SYSIN statement. The information associated with SYSIN is continued on subsequent lines. Line 37 contains the –PARM parameter. A semi-colon delimited list of keywords and values follows. Additional available parameters are described later in this document.

Another keyword in the –PARM parameters is the –DD_<DD name>. In this example, the keyword is –DD_COPYFILE=. The value following this keyword is the file name to be associated with the dataset defined in the DD COPYFILE statement.

11.9 Sample JCL – GET

The following JCL transfers a file from an SFTP host to the mainframe. The output file resides as a dataset on a virtual tape. It can be copied to DASD using ICEGENER after the virtual tape is created and cataloged.

```

1 //MDIGET JOB (ACCT),'UPLOAD NEW FILE',CLASS=H,
2 // CLASS=A,NOTIFY=&SYSUID

```

```

3 /**
4 /** THIS UPLOADS A SERVER FILE AND CREATES A NEW DATASET
5 /** ON A MDI/CGX TAPE
6 /**
7 /** AFTER THE TRANSFER THE TAPE CAN BE READ BY ANY AUTHORIZED JOB
8 /**
9 /**
10 /** JCLLIB ORDER=(LUMENG.MDI.JCL) JCLLIB CONTAINING LUMXPROC
11 /**
12 /** Profile must be defined to RACF
13 /** SET PROFILE=SFTPFROMHOSTA PROFILE IS PREDEFINED TO PROCESS FILE
14 /**
15 /**STEP100 EXEC LUMXPROC,PROFILE=&PROFILE
16 /**
17 /** OUTPUT DCB MUST MATCH BETWEEN THE SERVER AND JCL
18 /**
19 /**NEWDS DD DISP=(NEW,CATLG),DSN=PROD.UPLOAD.FILE,
20 /** DCB=(LRECL=1024,BLKSIZE=32768,RECFM=FB),
21 /** UNIT=MDITAPE MUST SPECIFY MDI UNIT
22 /**SYSIN DD *
23 -PARM destination=labbox1;
24 login=User1;
25 password=secret1;
26 postaction="LISTFILEL prod.master.file.new";
27 -DD_NEWDS=prod.master.file.new
28 /**

```

11.9.1 Virtual Tape Dataset

Lines 19-21 define the scratch VOLSER and dataset that receives the corresponding SFTP data. Access to the dataset associated with the tape is external to this JCL. It can be accessed by any authorized program or utility. The tape created is cataloged and can be accessed by the dataset name.

11.9.2 LUMXPROC

Line 15 shows the execution of the LUMXPROC program.

11.9.3 PROFILE

Line 15 also includes the SecureTransfer profile name. In this case the name is SFTPFROM-HOSTA. The SecureTransfer profile must be pre-defined and configured on the MDI Platform.

11.9.4 DD Name

Line 19 defines the DD name and associated dataset parameter. The DD name of NEWDS is important and is utilized later in the SYSIN to associate a file name to the dataset.

11.9.5 SYSIN

Lines 22-27 are the SYSIN statement. The information associated with SYSIN is continued on subsequent lines. Line 23 contains the –PARM parameters. A semi-colon delimited list of arguments follows. Another keyword in the SYSIN parameters is the –DD_<DD name>. In this example, the keyword is –DD_NEWDS=. The value following this keyword is the file name to be associated with the dataset defined in the DD NEWDS statement.

11.10 Executing a Test Using SecureTransfer

After installation and configuration is complete, it's recommended to execute a PUT and GET operation, from one or more servers that you want to exchange data with, to test the product and ensure that all security provisions have been put in place. Using the JCL examples provided above, create the JCL required to execute a test. Submit the job(s) and check the job's SYSOUT for zero return codes. A non-zero return code indicates that there was a problem with the job's execution. See [Appendix A: Message Codes](#) for non-zero return code information.

If you would like assistance creating the JCL and executing the tests, or resolving a non-zero return code, contact the Luminex Support team.

11.11 Results

Any failure to transfer data to or from the MDI Platform results in an ABEND code associated with that transfer program. The entire batch job abends if any step fails. If the LUMXPROC step fails, an MDI message-code is returned, and the appropriate ABEND code delivered. See [Appendix A: Message Codes](#) for non-zero return code information and help resolving the issue. If you would like assistance resolving a non-zero return code, contact the Luminex Support team.

11.12 Migration/JCL Conversion/Professional Services

Managed File Transfer solutions all have unique JCL, keyword and parameter requirements. FTP, FTPS and SFTP have unique parameters as well. Converting hundreds or thousands of existing JCL can be a time consuming and costly task. Luminex provides Professional Services to help clients convert their JCL quickly and easily. Luminex JCL Conversion Utility can convert hundreds of JCL decks in just minutes. The conversion utility works with all Managed File Transfer solutions, FTP, FTPS or SFTP. For more information on Luminex JCL Conversion Professional Services, please contact your Luminex Sales Representative.

11.12.1 Scratch Tape Management

The Scratch List Update Utility or LSCRUP is a package provided to you by the Luminex Support Team. This package contains a LOAD module and sample JCL (see MDI Mainframe Software Installation and Configuration, Scratch Update Processing for more information about installing the LSCRUP load module. LSCRUP is used to extract a list of scratch VOLSERS from your existing tape management report. A subsequent ICEGENER step then writes that scratch list to the MDI Platform. For more information about executing the LSCRUP scratch utility, see [Appendix C: LSCRUP Scratch List Update Utility](#).

11.13 SecureTransfer Security Requirements

The use of the mainframe program LUMXPROC to execute the SecureTransfer profile or profiles requires the setup of RACF permissions. The Security Administrator at your installation needs to define a FACILITY class profile or profiles to your security server and permit user ID(s) to the profile(s) in order to successfully execute LUMXPROC on your system. See the Mainframe Security Setup section of this guide for detailed information on security requirements.

11.14 SFTP Host Requirements

An IP connection between the MDI Platform and the SFTP host must be available. Access through network port 22 is required.

The user must be defined on the SFTP host as well as the user password. The user's home directory must also be defined. If a general-purpose directory, or directories, are to be accessible, the SFTP host must declare these. Typically, authentication and directory permissions are managed by the client's IT department.

11.15 Licensing

The Mainframe Data Integration solutions are licensed by MDI Platform. The use of more than one MDI Platform requires additional product licenses per Platform. Multiple product profiles can be deployed on a single MDI platform at no additional cost.

12. Cross-Platform Data Sharing (MDI:XPDS)

12.1 Introduction

MDI Cross-Platform Data Sharing (MDI:XPDS) is a bi-directional data sharing solution used to integrate mainframe data with Open System's applications and processes. This integration allows for processing data on multiple platforms as part of an enterprise wide application architecture.

As an example:

- Mainframe data can be transferred to a directory where an Apache Ignite computing grid accesses the data, processes it and returns the file back to the mainframe for further processing;
- Mainframe data can be converted to ASCII, transferred to a directory to be ingested by an Open Systems application for processing and distribution;
- Open Systems applications may place data in a directory to be converted to EBCDIC and transferred to the mainframe for processing.

The MDI:XPDS product enables the transfer of data, over the FICON channel, between the mainframe and NFS storage mounted to the MDI:XPDS Platform. After data is processed, it can be pushed to the mainframe, using the PDQ and File Watch features, for downstream batch processing and other uses. When multiple MDI Platforms are configured together in a cluster; it is expected that the storage is shared among all of the MDI Platforms. This sharing can be provided via NFS or a clustered file system. For purposes of this document, the storage location is simply termed a "file system."

To move data from the mainframe to a file system, the dataset(s) are first written to virtual tape(s) using the MDI Batch Interface (LUMXPROC) described earlier in this guide. When the data transfer is complete, the LUMXPROC program provides all the necessary information (file name/destination directory name) to direct MDI:XPDS to move the data to a directory defined on the destination file system.

To push data from a file system to the mainframe, LUMXPROC directs MDI:XPDS to transfer the file from a pre-defined "watched" directory on the file system. MDI:XPDS communicates to the Pending Data Queue (PDQ) started task on the mainframe to verify the user's credentials using a PassTicket, a security mechanism that is supported by all mainframe security products. Once approved, PDQ mounts a scratch virtual tape with the designated dataset name and tape formatting properties, writes the data to the tape, closes the tape and catalogs the tape dataset on the mainframe. When tape creation is complete, the dataset is then available to be used as any other virtual tape. The tape dataset may be copied to a disk dataset if required.

Any errors generated by the process are reported either as standard tape errors such as Unit or Equipment Checks, or through LUMXPROC, all of which causes an abend of the batch job. If the abend is in the LUMXPROC step, the batch job may be restarted at this step, to avoid copying the data again. Otherwise, the job can be restarted from the first step. Standard SYSPRINT is

generated on all jobs.

All data to and from the mainframe is transferred as data to or from a virtual tape. The virtual tape resides on storage made available to MDI:XPDS. This requires that the tape range defined to MDI:XPDS also be defined in the mainframe's tape management system. In most cases the tape data is transient, meaning the virtual tape can have a short expiration period. Once expired, the virtual tape can be re-used.

JCL is required to perform all data transfers via virtual tape. The data transfer can be performed by any application or utility, such as ICEGENER.

12.2 File System Mount Point

MDI requires that the file system be permanently mounted. The file system directory must be configured such that the user has read and/or write permissions. Multiple file system mount points may exist on a single MDI Platform, however, having too many mount points can affect system performance. Consult the Luminex Support team if system performance degradation becomes an issue.

MDI has a standardized naming convention for file system mounts. The mount point has the following format:

```
/luminex/storage/xxx
```

Where xxx is a name that is somewhat descriptive of the storage. Examples of a mount point used in this document will simply be of the form:

```
/luminex/storage/NFSstore
```

12.3 Cross-Platform Data Sharing Components

12.3.1 Mainframe

- LUMXPROC Load module
- LUMXPDQ, LUMXPDQC, LUMXPDQR, LUMXPDQU, LUMXPDQV Load modules
- LUMXPROC Procedure
- LUMXPDQ Procedure (started task)
- Batch JCL
- SAF Interface

12.3.2 MDI Platform

- Server Code
- MDI:XPDS Software
- Client's customized profile definitions
- File Watcher
- DCB Table

12.3.3 Storage

- Storage available on the MDI Platform (limited)
- Client provided Open Systems storage
- Optionally, Open Systems storage purchased with MDI:XPDS

12.3.4 MDI Reporting Interface

For reporting, a Graphical User Interface (GUI), is provided with MDI:XPDS. The GUI provides a dashboard of various reports including:

- Transmissions in Process
- Performance Reports
- Network Traffic
- MDI Put Bytes Transferred
- MDI Get Bytes Transferred
- Storage Consumption and Availability

Also available via the GUI is the ability to generate reports that show trending over a period of time:

- Performance Reports
- MDI Put Time
- MDI Get Time
- MDI Put Bytes Transferred
- MDI Get Bytes Transferred
- Storage Consumption and Availability

12.4 XPDS Planning Overview

The software components for the MDI Solutions provide for communication over the FICON channel between the mainframe and the MDI Platform. The MDI Platform appears as a tape device to the mainframe, allowing for the transfer of mainframe data simply by writing data to MDI owned tape. For communication and data movement over FICON to occur, the batch interface components must be installed on the mainframe. MDI product code (XPDS profile) is also installed and configured for your site on the MDI Platform by the Luminex Support team.

The following table describes the tasks necessary to begin using XPDS in your environment:

	Task:	Refer to Section in this Guide:
1	Product Installation on the mainframe	Section 6: MDI Mainframe Software Installation and Configuration
2	Setup and execution of LSCRUP scratch update	Appendix C: LSCRUP Scratch List Update Utility
3	Understanding and configuring Security Requirements	Section 6: MDI Mainframe Software Installation and Configuration ; Section 7: Mainframe Security Setup

4	Configuring File Watch and PDQ	Section 8: MDI File Watcher and Pending Data Queue (PDQ)
5	XPDS Profile Configuration	Section 4: Planning for MDI ; Section 5: MDI Profile Creation and the Installation Workbook ; Section 12.5: XPDS Profile Configuration
6	Understanding Data Conversion Options	Section 9: MDI Data Conversions
7	Understanding JCL Syntax Rules	Section 10: Syntax Rules for Parameters and KEYWORD=VALUE Pairs
8	Creating the JCL	Section 12.5.1: XPDS JCL
9	Executing a test	Section 12.10: Executing a Test Using XPDS
10	Converting existing JCL	Section 12.12: Migration/JCL Conversion/Professional Services
11	Interpreting Messages and Codes	Appendix A: Message Codes

12.5 XPDS Profile Configuration

An XPDS profile is configured on the MDI platform to transfer data to or from a file system. An XPDS profile transfers data in one or both directions, depending on how the profile is configured. The direction is determined in the XPDS profile that resides on MDI Platform. Upon installation, XPDS profiles are configured by the Luminex Support team to provide the desired functionality. Each profile is assigned a unique name. The LUMXPROC JCL references the desired profile name in the `PROFILE=value` JCL parameter where *value* is the client defined name assigned to the profile.

Providing operational arguments in the JCL provides flexibility in the client's environment; several users can use the same profile with a variety of options. However, if the client's goal is to mandate certain operational arguments, hard-coding options in the profile on the MDI Platform achieves that goal.

When operational arguments are found both in the JCL and in the XPDS profile on the MDI Platform, an error is returned indicating that the argument(s) must be removed from the JCL.

12.5.1 XPDS JCL

XPDS uses JCL on the mainframe to copy the file to the XPDS Platform, via virtual tape, and places the file in a specified directory on NFS mounted storage. The JCL typically contains a two-step process.

Step 1 is copying the data set to be transferred from disk to MDI owned virtual tape. It's recommended that a copy utility, such as ICEGENER or if you are licensed for SYNCSORT, SYNC-GENER be used in place of IEBGENER. These recommended copy utilities use buffering to execute faster and use less system overhead.

Step 2 Executes the LUMXPROC program passing parameters coded in the SYSIN DD card to the MDI Platform. If the data to be transferred has already been copied to MDI owned virtual

tape, Step 1 can be eliminated.

[Section 6: MDI Mainframe Software Installation and Configuration](#) contains general information about installing the LUMXPROC program that the JCL executes.

[Section 10: Syntax Rules for Parameters and KEYWORD=VALUE Pairs](#) describe general syntax rules for coding parameters and keywords/value pairs in the SYSIN DD card in the JCL. Please review this section carefully as many of the values provided in the JCL must be coded in lower case or mixed case, as they are transferred to distributed platforms for processing. **It is recommended that the CAPS OFF command be issued in your ISPF editor when working with MDI related JCL to prevent lower case or mixed case values being automatically converted to upper case.**

12.6 XPDS JCL Parameters and KEYWORD=VALUE Pairs

Parameters specific to the XPDS profile are designated in the SYSIN DD section of the JCL.

12.6.1 Operational Arguments

When an XPDS profile is configured, operational arguments, such as a directory, can be provided by the mainframe JCL or by the profile on MDI. If the mount point directory is defined in the profile configuration, that information is not required in the JCL. If not defined in the profile configuration, the full path is required in the JCL.

Parameters that are only supplied by the JCL are:

- preaction – action to be performed on the file system prior to the transfer
- postaction – action to be performed on the file system after the transfer

Parameters set only in the profile are as follows:

- Email address – one or more email addresses to send alert information
- Parallel Operation Count – the number of simultaneous file system transfers per individual job
- Directory – the base directory for file transfer (optional)
- Direction – either PUT (to file system), GET (from file system), or BOTH

12.6.2 JCL Parameters and Arguments

The JCL can designate one or more files, or datasets, that can be transferred between the mainframe and the file system. The MDI profile performs the transfer of these files simultaneously. The number of simultaneous transfers is defined by the Parallel Operation Count described above. For example, if the count is 4 and there are 5 files to transfer, 4 occur at the same time. The 5th one begins transfer as soon as one of the first 4 have completed.

Additional parameters are passed to an MDI profile via the SYSIN line. Two keywords are used to indicate and delimit these parameters. All values following –PARM are specific to the pro-

file. All values following –ARGS are not evaluated by the profile but are passed directly to any third-party program that may be called by the profile program. The XPDS profile does not utilize the –ARGS keyword. All values following –DD_<DD name> pertain to a DD statement defined earlier in the JCL. These provide information specific to the DD statement.

12.6.2.1 -PARM Parameter

The -PARM parameter is required in the JCL. It is an indicator for the XPDS profile configured on the MDI Platform that keywords and associated values are following.

The -PARM indicator is followed by a space or equal (=) sign.

There are 2 types of information provided after the -PARM parameter:

- 1) KEYWORD=VALUE pairs to define specific parameters for the XPDS profile.
- 2) -DD_<dd name>= parameter describes the actions specific to the file associated with the DD statement.

Below are additional available keywords that are provided by SYSIN and after the –PARM keyword.

12.6.3 -DD_<DD name> Parameter

The –DD_<DD name> keyword provides the file name used during the transfer for the corresponding DD name. This file name provides input, or output, for a data transfer. The filename immediately follows the –DD_<DD name> value. A –DD_ entry is required for each DD statement in the JCL. This defines either the input or output file name. For example, if two files are to be transferred, the DD statements associated with the transfers could be:

```
//COPYFIL1 DD DISP=OLD,DSN=PROD.MASTER.FILE1.MDI,
//          UNIT=&OUTDEV
//*
//COPYFIL2 DD DISP=OLD,DSN=PROD.MASTER.FILE2.MDI,
//          UNIT=&OUTDEV
```

The associated –DD_ values are:

```
-DD_COPYFIL1=syslog_020316
-DD_COPYFIL2=report.txt
```

Directory paths must be included with the filename. If only the file name is included in the -DD_ value, then the file transfer occurs to the directory defined at that user's home directory. If a directory is included with the file name, there are several syntaxes that specify whether the directory is a sub-directory off the user's home directory, or an explicitly defined directory. For example, if the home directory for the user Bob is /home/Bob, the following transfers the file, report.txt, to or from /home/Bob:

```
-DD_COPYFILE1=report.txt
```

The following copies report.txt into a sub-directory, /pub, off /home/Bob. Note that there is no leading / in front of pub.

-DD_COPYFILE1=pub/report.txt

To directly access a directory, the following syntax can be used. To transfer the file, report.txt, to or from /luminex/storage/NFSstore/reports:

-DD_COPYFILE1=/luminex/storage/NFSstore/reports/report.txt

Note: There is a / before luminex. This implies an explicitly defined directory path.

Sub-directories are not automatically created. If a sub-directory is to be created, the use of the preaction parameter is required. For example, to create a sub-directory, mysubdir, in the mount point directory /luminex/storage/NFSstore:

```
preaction="mkdir /luminex/storage/NFSstore/mysubdir"
```

Note: If multiple layers of sub-directories are to be created, the -p option is required. For example, to create the sub-directory, myfiles/092216, off the mount point:

```
preaction="mkdir -p /luminex/storage/NFSstore/myfiles/092216"
```

12.6.3.1 Keywords & Values

All parameters are specified in the NAME=VALUE format, where NAME is a keyword that defines the VALUE.

Most keywords are unique to each MDI solution. Keywords can be coded in upper case, lower case or a mixture of upper and lower case. KEYWORD=VALUE pairs can be delimited by a space or semi-colon.

Every KEYWORD= must have a value. KEYWORD=VALUE cannot have spaces before or after the equal sign (=).

Below are all valid syntax:

```
login=userid;password=pass  
LOGIN=userid PASSWORD=pass  
Login=userid; PassWord=pass
```

Values

Values are the variables entered after the equals (=) sign of the KEYWORD=VALUE pair.

Values that have spaces must be double quoted. As an example:

```
PREACTION="REMOVEFILE /a/b/c"
```

Values with multiple entries require double quotes and, in some cases, comma separators. As an

example:

12.6.3.2 The File Name Value in the DD_<dd name>= Parameter

The file name is the name of the file being transferred to the destination server or from a server to the mainframe. **File names are case sensitive.** If the file name contains spaces, the full file name must be enclosed by single quotes. File names with consecutive spaces are not allowed. For example, the following file name is valid:

```
'file with spaces'
```

The following is *not* supported:

```
'file      with      spaces'
```

The following are valid syntax:

```
-DD_TEST=testfile;  
-DD_TEST=testfile  
-DD_TEST='testfile'  
-DD_TEST='test  file'
```

When multiple DD statements exist in the JCL, multiple -DD_ <dd name>=value keywords are required, one for each data set name.

12.6.3.3 Special Handling of Conversion Keyword

The CONVERSION= value keyword is supported by all MDI solutions. It can be globally defined for all files if coded after the -PARM parameter. Alternatively, each file can be converted differently by including this KEYWORD=VALUE on the -DD_<dd name> line.

For example, to globally define a conversion for all files, put the KEYWORD=VALUE on the -PARM line:

```
-PARM CONVERSION=ascii_LF;  
-DD_UPLOAD=uploadfile;  
-DD_DOWNLOAD=downloadfile;
```

Files associated with one or more -DD_<dd name>= parameters are converted using the conversion of ascii_LF,

To specify a different conversion for each file, define on the -DD_<dd name>= line:

```
-DD_UPLOAD=uploadfile; CONVERSION=ebcdic  
-DD_DOWNLOAD=downloadfile; CONVERSION=ascii_LF;
```

12.6.3.4 Comments

Lines in the SYSIN DD card can be commented by keying an asterisk (*) in column 1. In the example below, the preactions keyword will not be executed.

```
//SYSIN DD *  
-PARMS
```



```

Login=mike
Password=pass
* preaction="listfilel /a/b/c"
-DD_UPLOAD=/a/b/c
oformat=ascii_LF

```

12.6.3.5 More Syntax Rules

Values that contain multiple words must be double quoted. An example of this is shown in the “preaction” section below.

Lines should be kept short for readability. No line should run past column 71 or it may not be properly evaluated. KEYWORD=VALUE’s can be placed on separate lines keeping in mind words cannot be split. Below is a complex example of valid syntax:

```

000054 -PARM destination=g7ftp10g parallel=1
000055 login=programmer
000056 password=password
000057 emailall="users@example.com"
000058 preaction="LISTFILEL testfile, LISTFILEL
000060 'test print133 data'"
000061 postaction="listfilel
000062 'test print133 data'"
000063 -DD_FBADOWN="'test print133 data'"
000064 conversion=ascii_CRLF
000065 ocode=nostrip
000066 -DD_FBAUP=
000067 "'test print133 data'" conversion=ebcdic_npc

```

12.6.3.6 Examples

The SAMPLIB partitioned data set (PDS) provided with your MDI solution contains JCL examples that include a variety of KEYWORDS and VALUES for the -PARM and -DD_ parameters in the SYSIN DD card. Please refer to this PDS for more examples.

12.6.4 XPDS Parameters

12.6.4.1 preaction

The preaction parameter allows for a command to be executed on the file system. This action is performed before any file transfer. The parameters consist of keywords and arguments. The arguments always include a path or filename. Below are some keywords and their actions:

- LISTFILE – Shows the existence of a file
- LISTFILEL – Shows the “long” information of a file including permissions and size.
- TESTFILE – Tests if a file exists. If not, the job will fail.
- REMOVEFILE – Will remove a file
- MAKEDIR – Will make a directory
- MAKEDIRP – Will make a multiple directory tree
- REMOVEDIR – Will remove a directory
- MOVEFILE – Will move a file from one path to another

In all cases, the user must have access to the directories involved in the action.

The examples below assume the user has access to the directory /luminex/storage/XPDSstore.

The following shows information on file: abc123:

```
LISTFILE /luminex/storage/XPDSstore/abc123  
LISTFILE abc123
```

To make the directory /luminex/storage/XPDSstore/temp:

```
MAKEDIR /luminex/storage/XPDSstore/temp  
MAKEDIR temp
```

Contact Luminex for more details on the available actions that can be performed.

12.6.4.2 postaction

The postaction parameter is identical to the preaction parameter except that it is executed after the file transfer. See preaction for the syntax of this action.

12.6.4.3 login

Optionally a user name (or login) can be used in the JCL. These users must be defined on each MDI Platform. The user name limits the locations, or directories, in which data can be transferred. Each user has a directory in which they have access. This directory is also predefined on each MDI. If no login is provided in the JCL, a default one is used. This default is assigned a directory in which data can be transferred to or from. NOTE: Upon installation, the default login and password and its associated directory must be created by Luminex Support.

12.6.4.4 password

The password is associated with the login described above.

12.6.4.5 permissions and usergroup

When XPDS transfers a file to NFS mounted storage, the files are transferred with root ownership. Permissions need to be granted to allow users (user groups) to access the data. The keywords *permissions=* and *usergroup=* are coded in the JCL after the -PARM parameter to provide permissions to user groups. These KEYWORD=VALUE pairs cause a “chmod” (change file mode bits) and “chown” (change owner) commands to be executed on the server to change the permissions of the file.

An example of these commands is provided below:

```
-PARM permissions=0664;usergroup=bob:engr
```

The chmod numerical format accepts up to four octal digits. The three rightmost digits refer to permissions for the file user, the group and others. The optional leading digit, when 4 digits are given, specifies the special setuid, setgid, and sticky flags. Each digit of the three rightmost digits represents a binary value, which its bits control the read, write, and execute respectively,

where 1 means allow and 0 means don't. Refer to your network administrator for more information and support for setting up permissions.

#	Permission	rwX	Binary
7	Read,write,and execute	rwX	111
6	Read and write	rw-	110
5	Read and execute	r-X	101
4	Read only	r--	100
3	Write and execute	-wX	011
2	Write only	-w-	010
1	Execute only	--X	001
0	None	---	000

12.6.4.6 oformat Keyword

This parameter, and the synonyms keyword “conversion” are used to define the output format for data being PUT, or written, to the File system. One or more values are available. They are defined as follows:

- native (default) – for variable block data, the blocking information is not removed from the blocks
- strip– for fixed block input data, no operation is performed; for variable block data, the block size (Block Descriptor Word) information is removed from the data
- ascii – this converts the data from EBCDIC to ASCII

For example, to convert to ASCII with a Unix line-feed added to the end of each record:

```
offormat=ascii_LF
```

or

```
conversion=ascii_LF
```

This setting is optional. The default setting is native.

Using one of the examples above, below is the way to convert a file to ascii:

```
-DD_COPYFILE1=report.txt;offormat=ascii_LF
```

or

```
-DD_COPYFILE1=report.txt;conversion=ascii_LF
```

12.7 File Path Conventions

The file to be accessed, read or written, is determined by the -DD_ statement in the SYSIN argu-

ments. Each user is assigned a permissible directory in which data can be accessed. If no user or login is provided in the JCL, a default directory is used. This user/directory information is stored in the profile configuration file. A user can also be granted read/write or only read access. By convention, if the file name contains no directory information, the user's assigned directory will be used. Below is a sample `-DD_` statement with no path, or directory, information:

```
-DD_COPYFILE=prod.master.file.new
```

Sub-directories can also be included in the file name. For example, if the file to be accessed is in the sub-directory "backup", the `-DD_` statement would be:

```
-DD_COPYFILE=backup/prod.master.file.new
```

Note that the sub-directory must already exist. This action can be performed in the "preaction" section. The pre/post actions also follow the rules of accessible user directories.

Full pathing can also be used but the full path must include part of the user's accessible directory. For example, if the user's directory is: `/home/storage/myfiles`, the following `-DD_` statement is valid:

```
-DD_COPYFILE=/home/storage/myfiles/backup/prod.master.file.new
```

email_error_list

This optional parameter can only be defined in the profile configuration on the MDI Platform.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output

email_list

This optional parameter can only be defined in the profile configuration on the MDI Platform.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output.

emailonerror

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output. Multiple emails must be comma separated and double quoted. For example:

```
emailonerror="errors@example.com,alerts@luminex.com"
```

emailall

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receive an email of all job results. The body of the email contains identical information as the job SYSTPRINT output. Multiple emails must be comma separated and double quoted. For example:

```
emailall="user1@example.com,alerts@luminex.com"

EMAILALL="alerts@luminex.com, user@example.com, support@example.com"
```

Handling Emails

Emails can be defined in both JCL and the profile configuration on the MDI Platform. If defined in both places, the lists are merged and duplicates removed. If an error occurs, the email addresses defined in: emailonerror (JCL), emailall (JCL), email_error_list (configuration), and email_list (configuration) are merged and duplicates removed.

12.8 Sample JCL – PUT

The following JCL is a simple example that transfers a single file to a file system host.

Description of the important parameters will be described below.

```
1  //MDITAPE  JOB (ACCT),'NFS NEW DATASET',CLASS=H,
2  // CLASS=A,NOTIFY=&SYSUID
3  //*
4  //* IEBGENER COPIES A DATASET TO MDI/CGX TAPE
5  //*
6  //*
7  // SET OUTDEV=MDITAPE      UNITNAME FOR MDI CGX CONTROLLER
8  // JCLLIB ORDER=(LUMENG.MDI.JCL)  JCLLIB CONTAINING LUMXPROC
9  //*
10 /*
11 //STEP010  EXEC PGM=ICEGENER
12 /*
13 /* SYSUT1  INPUT FILE CAN BE ON DISK OR ANY TAPE
14 /* SYSUT2  OUTPUT MUST BE MDI/CGX TAPE
15 /*
16 //SYSUT1   DD  DISP=OLD,DSN=PROD.MASTER.FILE      INPUT MASTER FILE
17 //SYSUT2   DD  DSN=PROD.MASTER.FILE.MDI,          OUTPUT COPY
18 //          DCB=*.SYSUT1,                          USE SAME DCB AS INPUT
19 //          VOL=(,RETAIN),                          DON'T DISMOUNT TAPE
20 //          DISP=(NEW,CATLG),
21 //          UNIT=&OUTDEV,                          MDI OUTPUT UNIT MUST BE CODED
22 //          RETPD=0                                RETAIN OUTPUT TAPE ZERO DAYS
23 //SYSPRINT DD  SYSOUT=*
24 /*
25 /*
26 /* USE MDI TO TRANSFER FILE THE TAPE JUST CREATED
27 /* INPUT FILE IS THE FILE JUST CREATED
28 /* OUTPUT FILE ON THE SERVER IS PROD.MASTER.FILE.NEW
29 /*
30 // SET      PROFILE=NFS2HOSTA      PROFILE IS PREDEFINED TO PROCESS FILE
31 /*
32 //STEP020  EXEC  LUMXPROC,PROFILE=&PROFILE
33 /*
34 //COPYFILE DD  DISP=OLD,DSN=PROD.MASTER.FILE.MDI,
35 //          UNIT=&OUTDEV          MUST SPECIFY MDI UNIT
```

```

36 //SYSIN      DD *
37 -PARM destination=labbox1;login=User1;password=secret1;
38      postaction="LISTFILEL prod.master.file.new";
39 -DD_COPYFILE=prod.master.file.new
40 /*
41 /*

```

12.8.1 Dataset Transfer to Virtual Tape

Line 11 creates the output tape using an application program or utility to copy the dataset to a tape. In this case IEBGENER is used.

12.8.2 LUMXPROC

Line 32 shows the execution of the LUMXPROC program. This step must follow the transfer of the dataset(s) to MDI.

12.8.3 PROFILE

Line 30 sets the profile name. In this case the name is NFS2HOSTA. This profile must be pre-defined and configured on the MDI Platform.

12.8.4 DD Name

Line 34 defines the DD name and associated dataset parameters. The DD name of COPYFILE is important and is utilized later in the SYSIN to associate a file name to the dataset.

12.8.5 SYSIN

Line 36 is the start of the SYSIN statement. The information associated with SYSIN is continued on subsequent lines. Line 37 contains the –PARM parameters. A semi-colon delimited list of arguments follows. Additional available parameters are described later in this document. Another keyword in the –PARM parameters is the –DD_<DD name>. In this example, the keyword is –DD_COPYFILE. The value following this keyword is the file name to be associated with the dataset defined in the DD COPYFILE statement.

12.9 Sample JCL – GET

The following JCL transfers a file from a file system host to the mainframe. The file resides as a dataset on a MDI owned virtual tape.

```

1  //MDIGET   JOB (ACCT),'UPLOAD NEW FILE',CLASS=H,
2  // CLASS=A,NOTIFY=&SYSUID
3  /*
4  /* THIS UPLOADS A SERVER FILE AND CREATES A NEW DATASET
5  /* ON A MDI/CGX TAPE
6  /*
7  /* AFTER THE TRANSFER THE TAPE CAN BE READ BY ANY AUTHORIZED JOB
8  /*
9  /*
10 // JCLLIB ORDER=(LUMENG.MDI.JCL)    JCLLIB CONTAINING LUMXPROC
11 /*
12 /*
13 // SET     PROFILE=NFSFROMHOSTA      PROFILE IS PREDEFINED TO PROCESS FILE
14 /*
15 //STEP100 EXEC  LUMXPROC,PROFILE=&PROFILE
16 /*

```

```

17 /** OUTPUT DCB MUST MATCH BETWEEN THE SERVER AND JCL
18 /**
19 //NEWDS      DD DISP=(NEW,CATLG),DSN=PROD.UPLOAD.FILE,
20 //              DCB=(LRECL=1024,BLKSIZE=32768,RECFM=FB),
21 //              UNIT=MDITAPE      MUST SPECIFY MDI UNIT
22 //SYSIN      DD *
23 -PARM destination=labbox1;
24     login=User1;
25     postaction="LISTFILEL  prod.master.file.new";
26 -DD_NEWDS=prod.master.file.new
27 /**

```

12.9.1 Virtual Tape Dataset

Lines 19-21 define the scratch VOLSER and dataset that receives the corresponding NFS data. Access to the dataset associated with the tape is external to this JCL. It can be accessed by any authorized program or utility. The tape created with the dataset is cataloged and can be accessed by the dataset name.

12.9.2 LUMXPROC

Line 15 shows the execution of the LUMXPROC program.

12.9.3 PROFILE

Line 15 also includes the profile name. In this case the name is NFSFROMHOSTA. This profile must be pre-defined and configured on MDI.

12.9.4 DD Name

Line 19 defines the DD name and associated dataset parameter. The DD name of NEWDS is important and is utilized later in the SYSIN to associate a file name to the dataset.

12.9.5 SYSIN

Lines 22-27 are the SYSIN statement. The information associated with SYSIN is continued on subsequent lines. Line 23 contains the –PARM parameters. A semi-colon delimited list of arguments follows. Additional available parameters are described later in this document. Another keyword in the SYSIN parameters is the –DD_<DD name>. In this example, the keyword is –DD_NEWDS. The value following this keyword is the file name to be associated with the dataset defined in the DD NEWDS statement.

12.10 Executing a Test Using XPDS

After installation and configuration is complete, it's recommended to execute a PUT and GET operation, from one or more servers that you want to exchange data with, to test the product and ensure that all security provisions have been put in place. Using the JCL examples provided above, create the JCL required to execute a test. Submit the job(s) and check the job's SYSOUT for zero return codes. A non-zero return code indicates that there was a problem with the job's execution. See [Appendix A: Message Codes](#) for non-zero return code information.

If you would like assistance creating the JCL and executing the tests, or resolving a non-zero return code, contact the Luminex Support team.

12.11 Job Results

Any failure to transfer data to or from the MDI Platform results in an ABEND code associated with that transfer program. The entire batch job abends if any step fails. If the LUMXPROC step fails, an MDI message-code is returned, and the appropriate ABEND code delivered. See [Appendix A: Message Codes](#) for non-zero return code information and help resolving the issue. If you would like assistance resolving a non-zero return code, contact the Luminex Support team.

12.12 Migration/JCL Conversion/Professional Services

Luminex provides Professional Services to help clients create/convert their JCL quickly and easily. Luminex JCL Conversion Utility can convert hundreds of JCL decks in just minutes. The conversion utility works with all Managed File Transfer solutions, FTP, FTPS or SFTP. For more information on Luminex JCL Conversion Professional Services, please contact your Luminex Sales Representative.

12.13 Scratch Tape Management

The Scratch List Update Utility or LSCRUP is a package provided to you by the Luminex Support Team. This package contains a LOAD module and sample JCL (see MDI Mainframe Software Installation and Configuration, Scratch Update Processing for more information about installing the LSCRUP load module). LSCRUP is used to extract a list of scratch VOLSERs from your existing tape management report. A subsequent ICEGENER step then writes that scratch list to the MDI Platform. For more information about executing the LSCRUP scratch utility, see [Appendix C: LSCRUP Scratch List Update Utility](#).

12.14 RACF Profile Security

The use of the mainframe program LUMXPROC to execute the Cross-Platform Data Sharing profile or profiles requires the setup of RACF permissions. The Security Administrator at your installation needs to define a FACILITY class profile or profiles to your security server and permit user ID(s) to the profile(s) in order to successfully execute LUMXPROC on your system. XPDS uses the class name OXP for security profiles. The direction of PUT, GET or BOTH is also required.

The use of PDQ and File Watch requires a started task and security setup on the mainframe. See [Section 7: Mainframe Security Setup](#) for detailed information on security requirements.

12.15 MDI Default User Requirements

A default user of MDIUSER must be created on each MDI Platform. This user name is given limited access to the directory defined for the data. For example, if the directory of use is an NFS mount and is at /luminex/storage/NFSHostA/MDIUSER, the following needs to be created with the password “luminex”:

```
useradd -d /luminex/storage/NFSHostA/MDIUSER MDIUSER
```

If other users with access to other directories is required, a Linux administrator is required to meet these specific requirements. The Luminex Support can provide assistance if needed.

12.16 Profile Licensing

The Mainframe Data Integration solutions are licensed by MDI Platform. The use of more than one MDI Platform requires additional product licenses per Platform. Multiple product profiles can be deployed on a single MDI platform at no additional cost.

13. MDI BigData Transfer (MDI:BDT)

13.1 Introduction

Mainframe Data Integration Big Data Transfer (MDI:BDT) provides a way to transfer data securely from the mainframe to an Apache Hadoop Distributed File System (HDFS) cluster. Apache Hadoop is an open-source software framework used for distributed storage and processing of data sets of big data using the MapReduce programming model.

The term *Hadoop* has come to also refer to the *ecosystem*, or collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Phoenix, Apache Spark, Apache ZooKeeper, Cloudera Impala, Apache Flume, Apache Sqoop, Apache Oozie, and Apache Storm.

Also supported is the Azure Data Lake, a scalable data storage and analytic service for big-data analytics workloads that require developers to run massively parallel queries and Azure HDInsight, a big data relevant service, that deploys Hortonworks Hadoop on Microsoft Azure, and supports the creation of Hadoop clusters using Linux with Ubuntu.

MDI:BDT transfers data from the mainframe to the HDFS cluster using a FICON connected Linux server referred to as the MDI Platform. Software on the mainframe and on the MDI Platform work together to transfer data from the mainframe, over secure FICON channels, to the MDI Platform where either the HDFS script provided by Apache Hadoop or a webHDFS REST API ingests the data into Hadoop. Support for Microsoft Azure is provided via webHDFS.

The advantages to using MDI:BDT over traditional TCP/IP methods include:

- FICON transfers data using significantly less system overhead than TCP/IP
- Large files can be transferred off the mainframe faster than FTP/SFTP and similar technologies
- Use of the FICON channel as opposed to TCP/IP is more secure than using unsecure TCP/IP
- An audit trail of all file transfers including data/time, file size and success or failure indicators
- Automatic email notification options for alerting interested parties that file transfers have completed
- Error reporting in the mainframe SYSOUT as well as in the MDI:BDT GUI for problem determination and resolution
- Existing Luminex Mainframe Virtual Tape users can leverage the existing infrastructure for MDI:BDT file transfers.

13.1.1 BigData Transfer Components

Mainframe

- LUMXPROC Load module
- LUMXPROC Procedure

- Batch JCL
- SAF Interface

MDI Platform

- Server Code
- MDI:BDT Software
- Client's customized profile definitions

Storage

- Storage available on the MDI Platform (limited)
- Client provided Open Systems storage
- Optionally, Open Systems storage purchased with MDI BigData Transfer

MDI Reporting Interface

For reporting, a Graphical User Interface (GUI), is provided with MDI:BDT. The GUI provides a dashboard of various reports including:

- Transmissions in Process
- Performance Reports
- Network Traffic
- MDI Put Bytes Transferred
- MDI Get Bytes Transferred
- Storage Consumption and Availability

Also available via the GUI is the ability to generate reports that show trending over a period of time:

- Performance Reports
- MDI Put Time
- MDI Get Time
- MDI Put Bytes Transferred
- MDI Get Bytes Transferred
- Storage Consumption and Availability

13.1.2 BigData Transfer Planning Overview

The software components for the MDI Solutions provide for communication over the FICON channel between the mainframe and the MDI Platform. The MDI Platform appears as a tape device to the mainframe, allowing for the transfer of mainframe data simply by writing data to MDI owned tape. For communication and data movement over FICON to occur, the batch interface components must be installed on the mainframe. MDI product code (BigData Transfer profile) is also installed and configured for your site on the MDI Platform by the Luminex Support team.

The following table describes the tasks necessary to begin using MDI BigData Transfer in your environment:

	Task:	Refer to Section in this Guide:
1	Product Installation on the mainframe	Section 6: MDI Mainframe Software Installation and Configuration
2	Setup and execution of LSCRUP scratch update	Appendix C: LSCRUP Scratch List Update Utility
3	Understanding and configuring Security Requirements	Section 6: MDI Mainframe Software Installation and Configuration ; Section 7: Mainframe Security Setup
5	BigData Transfer Profile Configuration	Section 4: Planning for MDI ; Section 5: MDI Profile Creation and the Installation Workbook ; Section 13.1.3: Profile Configuration
6	Configuring Azure	Section 13.2: Configure Azure for Use with BigData Transfer
7	Understanding Data Conversion Options	Section 9: MDI Data Conversions
7	Understanding JCL Syntax Rules	Section 10: Syntax Rules for Parameters and KEYWORD=VALUE Pairs
8	Creating the JCL	Section 13.4: BigData Transfer JCL Section 13.4: BigData Transfer JCL
9	Executing a test	Section 13.7: Executing a Test Using BigData Transfer Section 13.7: Executing a Test Using BigData Transfer
10	Converting existing JCL	Section 13.9: Migration/JCL Conversion/Professional Services
11	Interpreting Messages and Codes	Appendix A: Message Codes

13.1.3 Profile Configuration

When a BigData Transfer profile is configured, operational arguments such as user name, user password, host name or IP, can be configured in the profile or provided by the mainframe JCL. When operational arguments are configured in the profile, the same arguments must not be coded in the JCL. When operational arguments are found both in the JCL and in the BigData Transfer profile on the MDI Platform, an error is returned indicating that the argument(s) must be removed from the JCL.

Providing operational arguments in the JCL provides flexibility in the client's environment; several users can use the same profile with a variety of options. However, if the client's goal is to mandate certain operational arguments, hard-coding options in the BigData Transfer profile on the MDI Platform achieves that goal.

The section, MDI:BDT JCL Parameters and KEYWORD=VALUE Pairs, describes the parameters specific to the BigData Transfer profile.

13.2 Configure Azure for Use with BigData Transfer



The steps for configuring Azure are described on the Microsoft Azure web site. In particular:

<https://docs.microsoft.com/en-us/azure/data-lake-store/data-lake-store-service-to-service-authenticate-using-active-directory>.

If you do not have an existing Data Lake Store, create a Data Lake Store from the **New Data Lake Store** blade: **New** → **Storage** → **Data Lake Store**.

Enter a unique name for your Data Lake Store and either select or create a resource group. Then select **Create** to create the Data Lake store.

Clicking on the name of your Data Lake Store from the **Dashboard** will give you the URL to use for **hdfs_host**

 Delete	 Data explorer
<hr/>	
Essentials ^	
<hr/>	
Resource group (change)	Pricing tier
luminexmdi	Pay-as-You-Go
Status	URL
Running	https://luminexmdi.azuredatalakestore.net
Location	ADL URI
East US 2	adl://luminexmdi.azuredatalakestore.net
Subscription name (change)	Subscription ID
Pay-As-You-Go	5296d5ba-236e-4e91-ac9d-fe71222b4edf

13.2.1 Create an Active Directory Web Application

The steps for configuring an Active Directory Web Application are described on the Microsoft Azure web site. In particular:




<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal>

You must have permissions to create and register an Active Directory web application. From the Azure portal, select Azure Active Directory → App registrations

Select “**New application registration**” and fill in the **Create** form. Make sure to select “**Web app / API**” for the Application type. The Sign on URL is not used by MDI:BDT, you can set it to **http://localhost**.

13.2.2 Finding Your Application ID

MDI
Registered app

 Settings  Manifest  Delete

Display name

MDI

Application type

Web app / API


Home page

https://localhost

Application ID

17aa927a-2d63-4ac5-a531-ff90d2a8ab0c

Object ID

1aea0fa0-8e70-4840-9dad-058f21c76723 

Managed application in local directory

MDI

From the Azure portal, select **Azure Active Directory** → **App registrations** and select the application you previously created.



The application ID is listed on the resulting blade underneath **Application ID**. The application id is of the form 17aa927a-2d63-4ac5-a531-ff90d2a8ab0c. Use that string for `hdfs_azure_application_id`.


13.2.3 Generating and Finding Your Authentication Key

From the **Settings** blade, select **Keys**.

Provide a description and duration for the key and then **Save**.

Copy the **VALUE** field and save as `hdfs_azure_application_key`. Note that you will not be able to retrieve the key once you close the blade. Save the application key someplace where you won't lose it.

 Save  Discard  Upload Public Key



 Copy the key value. You won't be able to retrieve after you leave this blade.

Passwords

DESCRIPTION	EXPIRES	VALUE
MDI Key	1/23/2019	O3YfXufrmavmk9R3XPFRUm8LfCbakN8siNrtmhURBpM= ...
<input type="text" value="Key description"/>	<input type="text" value="Duration"/> 	<input type="text" value="Value will be displayed on save"/> ...

13.2.4 Find Tenant ID

Select Azure Active Directory → Properties.

 Save  Discard

*** Name**

Default Directory


Country or region

United States

Location


United States datacenters

Notification language

English 

Global admin can manage Azure Subscriptions

Directory ID

379655af-8aae-4fd3-ba1f-a6386ce4dca5 

The `hdfs_azure_tenant_id` is listed under **Directory ID**.

13.2.5 Assign Application to Role

Select **Subscriptions** (search for it by selecting **More services** from the left hand menu).

Select the appropriate subscription and then select **Access Control (IAM)**.

Select **Add** and choose a role for your application. Make sure that the role has both read and write permissions.

Search for your application and then select **Save**.

13.2.6 Configuring a Firewall

A Firewall needs to be configured so that the MDI server has https access to `login.microsoftonline.com` and to the `hdfs_host` for their data lake (i.e. `luminexmdi.azuredatalakestore.net`).

Those connects can be tested using curl from the command line:

```
curl https://login.microsoftonline.com
```

should output:

```
<html><head><title>Object moved</title></head><body>  
<h2>Object moved to <a href="https://www.office.com/login">here</a>.</h2>  
</body></html>
```

And (replace luminexmdi.azuredatalakestore.net with your actual hdfs_host):

```
curl https://luminexmdi.azuredatalakestore.net
```

should output:

```
{“error”: {“code”: “ResourceNotFound”, “message”: “The request path  
https://luminexmdi.azuredatalakestore.net/” is not found. Trace: d3e5030c-1c3d-4f9f-b3e0-  
15fdca67ae43 Time: 2018-06-08T10:05:39.4342058-07:00”}}
```

13.3 Data Conversion

Data from the mainframe is in EBCDIC format. In some cases, data can be moved into the Hadoop file system with no conversion as the data is converted later by another process. In some cases, it is necessary to convert the data before sending it to the Hadoop file system. Data Conversion keywords and syntax can be found in the MDI Data Conversions Guide.

13.4 BigData Transfer JCL

MDI:BDT uses JCL on the mainframe to copy the file to the MDI:BDT Platform, via virtual tape, and transmit the file to a destination. The JCL typically contains a two-step process.

Step 1 is copying the data set to be transferred from disk to MDI owned virtual tape. It’s recommended that a copy utility, such as ICEGENER or if you are licensed for SYNCSORT, SYNC-GENER be used in place of IEBGENER. These recommended copy utilities use buffering to execute faster and use less system overhead.

Step 2 Executes the LUMXPROC program passing parameters coded in the SYSIN DD card to the MDI Platform.

Note: If the data to be transferred has already been copied to MDI owned virtual tape, Step 1 can be eliminated.

Section 6: MDI Mainframe Software Installation and Configuration contains general information about installing the LUMXPROC program that the JCL executes.

Section 10: Syntax Rules for Parameters and KEYWORD=VALUE Pairs describe general syntax rules for coding parameters and keywords/value pairs in the SYSIN DD card in the JCL. Please review this section carefully as many of the values provided in the JCL must be coded in lower case or mixed case, as they are transferred to distributed platforms for processing. **It is recommended that the CAPS OFF command be issued in your ISPF editor when working with MDI related JCL to prevent lower case or mixed case values being automatically converted to upper case.**

13.5 BigData Transfer JCL Parameters and KEYWORD=VALUE Pairs

Parameters specific to the BigData Transfer profile are designated in the SYSIN DD section of the JCL.

13.5.1 The -PARM Parameter

The -PARM parameter is required in the JCL. It is an indicator for the MDI BigData Transfer profile configured on the MDI Platform that keywords and associated values are following.

The -PARM indicator is followed by a space or equal (=) sign.

There are 2 types of information provided after the -PARM parameter:

- 1) KEYWORD=VALUE pairs to define specific parameters for the BigData Transfer profile.
- 2) -DD_<dd name>= parameter describes the actions specific to the file associated with the DD statement.

13.5.2 -DD_<DD name> Parameter

The -DD_<DD name> parameter provides the file name on the server for PUT operations or on the mainframe for GET operations. This file name provides input or output for a data transfer. The filename immediately follows the -DD_<DD name>= value. A -DD_<DD name>= value is required for each DD statement in the JCL. This defines either the input or output file name. For example, if two files are to be transferred, the DD statements associated with the transfers could be:

```
//COPYFIL1 DD DISP=OLD,DSN=PROD.MASTER.FILE1.MDI,
//          UNIT=&OUTDEV
//*
//COPYFIL2 DD DISP=OLD,DSN=PROD.MASTER.FILE2.MDI,
//          UNIT=&OUTDEV
```

The associated -DD_ values are:

```
-DD_COPYFIL1=syslog_020316
-DD_COPYFIL2=report.txt
```

Directory paths must be included with the filename.

Each user, whose user name is the login name, has a “home” directory on the SFTP host. If only the file name is included in the -DD_<DD name>=value, then that file is transferred to or from the user’s home directory.

If a directory is included with the file name, there are several syntaxes that specify whether the directory is a sub-directory off the user’s home directory, or an explicitly defined directory. For example, if the user’s home directory is /home/Bob, the following transfers the file, report.txt, to or from /home/Bob:

```
-DD_COPYFILE1=report.txt
```


The following copies report.txt into a sub-directory, /pub, off /home/Bob. Note that there is no leading / in front of pub.

```
-DD_COPYFILE1=pub/report.txt
```

If the user has permissions to access other directories on the SFTP host, the following syntax can be used. As an example, the user has permissions to access /Documents/reports. To transfer the file, report.txt, to or from /Documents/reports:

```
-DD_COPYFILE1=/Documents/reports/report.txt
```

Note that there is a / before Documents. This implies an explicitly defined directory that is not associated with the user's home directory. The user must have permissions to access this directory.

Sub-directories are not automatically created. If a sub-directory is to be created, the use of the preaction parameter is required. For example, to create a sub-directory, mysubdir, off the user's home directory:

```
preaction="MAKEDIR mysubdir"
```

To create a sub-directory, /myfiles, off /Document/pub:

```
preaction="MAKEDIR /Document/pub/myfiles"
```

Note: If multiple layers of sub-directories are to be created, MAKEDIRP is required. For example, to create the sub-directory, /myfiles/092216, off the user's home directory:

```
preaction="MAKEDIRP myfiles/092216"
```

13.5.3 hdfs_azure_application_id

The Azure Application ID generated when you created your Active Directory Web Application (see Finding Your Application ID). This parameter can either be specified in the profile configuration file or JCL. There is no default value.

When specified in the configuration file:

```
<hdfs_azure_application_id type="String">application_id</hdfs_azure_application_id>
```

When specified in the JCL, add the following to the -PARMS:

```
hdfs_azure_application_id=application_id
```

13.5.4 hdfs_azure_application_key

The Azure Application Key used to authenticate with an Azure Directory Web Application (see [Section 13.2.3: Generating and Finding Your Authentication Key](#) [Section 13.2.3: Generating and Finding Your Authentication Key](#)). This parameter can either be specified in the profile configuration file or JCL. There is no default value.

When specified in the configuration file:

```
<hdfs_azure_application_key type="String">application_key</hdfs_azure_application_key>
```

When specified in the JCL, add the following to the -PARMS:

```
hdfs_azure_application_key=application_key
```

NOTE: you must save your authentication key when you generate the key. You cannot retrieve the authentication key from the Azure portal later.

13.5.5 hdfs_azure_tenant_id

The Azure Tenant ID is the string that identifies your organization's space in the Azure cloud (see [Section 13.2.4: Find Tenant ID](#) *Section 13.2.4: Find Tenant ID*). This parameter can either be specified in the profile configuration file or JCL. There is no default value.

When specified in the configuration file:

```
<hdfs_tenant_id type="String">tenant_id</hdfs_tenant_id>
```

When specified in the JCL, add the following to the -PARMS:

```
hdfs_tenant_id=tenant_id
```

13.5.6 hdfs_host

The URL to use for the webHDFS REST API. If you are using Azure, you will find the `hdfs_host` on the Azure Portal dashboard for your data lake store (see [Section 13.2: Configure Azure for Use with BigData Transfer](#) *Section 13.2: Configure Azure for Use with BigData Transfer* [Section 13.2: Configure Azure for Use with BigData Transfer](#)). If you are using a private webHDFS cluster then your system administrator will provide you with the hostname.

This parameter can either be specified in the profile configuration file or JCL. There is no default value. But this parameter is required if **hdfs_protocol** is **webhdfs**.

If specified in the configuration file:

```
<hdfs_host type="String">hdfs_host</hdfs_host>
```

If specified in the JCL, add the following to your -PARMS:

```
hdfs_host=hdfs_host
```

13.5.7 hdfs_protocol

You can use the **hdfs** script provided by Apache Hadoop or the webHDFS REST API to transfer files. This parameter can be specified in the profile configuration file or JCL. This parameter is required and must be either **direct** or **webhdfs**.

13.5.8 hdfs_port

The port number to use for the webHDFS REST API. This parameter can either be specified in the profile configuration file or JCL. There is no default value, though Apache Hadoop usually uses 50070. If your site changed the default Apache Hadoop configuration then you will need to get the port value from your system administrator. `Hdf_port` is not used when the **hdfs_protocol** is **direct** or you are using Azure. This parameter is required if **hdfs_protocol** is **webhdfs**.

If specified in the configuration file:

```
<hdfs_port type="Int32">port_number</hdfs_port>
```

If specified in the JCL, add the following to your -PARMS:

```
hdfs_port=port_number
```

13.5.9 hdfs_path

The path to the **hadoop** script when using the **direct** protocol. This parameter is specified in the profile configuration file and depends on where Apache Hadoop is installed on the MDI host.

This parameter is required if **hdfs_protocol** is **direct**.

```
<hdfs_path type="String">path_to_hadoop_script</hdfs_path>
```

13.5.10 hdfs_directory

Subdirectory of the user directory in the HDFS cluster where the data is written. This parameter can either be specified in the profile configuration file or JCL. There is no default value.

If specified in the configuration file:

```
<hdfs_directory type="String">path_to_user_directory</hdfs_directory>
```

If specified in the JCL, add the following to your -PARMS:

```
hdfs_directory=path_to_user_directory
```

13.5.11 hdfs_java

The location where Java is installed and depends on the version of Java installed on the MDI server. This parameter is specified in the profile configuration file. This parameter is required if **hdfs_protocol** is **direct**.

```
<hdfs_java type="String">path_to_java</hdfs_java>
```

13.5.12 login

The user name to use with webHDFS (not used with Azure). Your system administrator can provide this information. This parameter can either be specified in the profile configuration file or JCL. There is no default value. This parameter is required if **hdfs_protocol** is **webhdfs**.

If specified in the configuration file:

```
<login type="String">user_name</login>
```

If specified in JCL, add the following to your -PARMS:

```
Login=user_name
```

13.5.13 bucket

Google Cloud Storage (GCS) uses a “bucket” to indicate where data will be stored. The bucket is pre-defined by the client environment and a bucket name is assigned.

The bucket keyword is used to indicate the bucket name to transfer data to. It is specified as one of the JCL parameters.

To have the flexibility to use the same BDT profile configuration on the server and be able to indicate different bucket names in the JCL, the bucket parameter in the BDT profile configuration on the MDI platform must be empty. Conversely, if it's convenient to create a BDT profile configuration for a single bucket name, it can be specified in the BDT profile. If the bucket name is indicated in the BDT profile and also in the JCL, the program will return an error.

If specified in JCL, add the following to your -PARMS:

```
bucket=bucketname
```

13.5.14 compression

When compression is desired, the JCL keyword, `Compress`, can be used. When set to `TRUE`, the data being passed is sent with gzip compression and the file extension `.gz` is automatically added to the resulting file. When compression is not desired, set the keyword value to `FALSE`.

To ensure compression is always used, it can be indicated in the BDT profile configuration on the MDI platform. Conversely, if compression is useful in some cases and not in others, the JCL keyword, `compress=value`, can be used. If used in the JCL, the `Compress` keyword in the BDT profile on the MDI platform should be blank. If the bucket name is indicated in the BDT profile configuration on the MDI platform and also in the JCL, the program will return an error.

```
compress=TRUE or FALSE
```

13.5.15 Passing the credentials file using a JCL DD statement

In lieu of passing credentials in clear text in the JCL, credentials can be specified in a dataset indicated by a separate DD statement in the LUMXPROC job step.

```
//KEYFILE DD DISP=OLD,DSN=KEYFILE,  
// UNIT=&DEV
```

When using this method to pass credentials, the keyword `DD_KEYFILE=value` must be placed in the -PARM section of the LUMXPROC SYSIN statement. The matching DD statement in the parameters must specify the `type=keys` keyword and it must be placed before the DD statement for the data.

The example below is using a key file named `GCS.KEYS`, is converting the keyfile to ASCII with the `oformat=value` keyword and indicates that this is a key file by using the `type=keys` keyword.

```
-DD_KEYFILE=GCS.KEYS oformat=ascii type=keys
```

Below is a JCL example showing a sample credentials dataset, named `GCS.KEYS`, being passed into the job and the associated `DD_KEYFILE=value`, the `oformat=ASCII` keyword and the `type=keys` keyword being specified:

```
//XPROC1 EXEC LUMXPROC,PROFILE=&PROFILE
```

```
//XPROCLOG DD SYSOUT=*
//KEYFILE DD DISP=OLD, DSN=GCS.KEYS,
// UNIT=&DEV
//FILE1 DD DISP=OLD, DSN=DEMO.DATA,
// UNIT=&DEV
//SYSIN DD *,SYMBOLS=JCLONLY
-PARM
emailall="me@example.com";
bucket="luminextest"
compress="TRUE"
-DD_KEYFILE=GCS.KEYS oformat=ascii type=keys
-DD_FILE1=DEMO.DATA oformat=ascii_crlf
/*
```

13.5.16 email_error_list

This optional parameter can only be defined in the profile configuration on the MDI Platform.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output

13.5.17 email_list

This optional parameter can only be defined in the profile configuration on the MDI Platform.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output.

13.5.18 emailonerror

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output. Multiple emails must be comma separated and double quoted. For example:

```
emailonerror="errors@example.com,alerts@luminex.com"
```

13.5.19 emailall

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receive an email of all job results. The body of the email contains identical information as the job SYSPRINT output. Multiple emails must be comma separated and double quoted. For example:

```
emailall="user1@example.com,alerts@luminex.com"
EMAILALL="alerts@luminex.com, user@example.com, support@example.com"
```

13.5.20 Handling Emails

Emails can be defined in both JCL and the profile configuration on the MDI Platform. If defined in both places, the lists are merged and duplicates removed. If an error occurs, the email addresses defined in: emailonerror (JCL), emailall (JCL), email_error_list (configuration), and email_list

(configuration) are merged and duplicates removed.

13.6 JCL Examples

13.6.1 Direct Protocol JCL Example

When `hdfs_protocol` is “direct” then files are transferred using the `hdfs` script provided by Apache Hadoop.

```
//HDFS1      JOB   (ACCT),'HDFS',CLASS=A,MSGCLASS=W,
//*   RESTART=LUMXPROC.HDFS,
//   NOTIFY=&SYSUID
//*
//*
//* 1) copy disk file to mditape
//* 2) HDFS profile copies file to Hadoop cluster
//* 3) retrieve data back to mainframe
//*
//*
// SET TAPE=MDITAPE           MDI tape
// SET DSN=TEST.BIG.DATA      disk dataset to be processed
//*
//COPY      EXEC PGM=ICEGENER      copy disk file to MDI tape
//SYSPRINT DD  SYSOUT=*
//SYSUT1    DD  DSN=&DSN,DISP=OLD
//SYSUT2    DD  DSN=&DSN..&TAPE,
//              DISP=(NEW,PASS),UNIT=&TAPE,RETPD=0
//SYSIN     DD  DUMMY
//*
//*
// SET PROFILE=HDFS          Hadoop
//*
//HDFS      EXEC LUMXPROC,PROFILE=&PROFILE
//DOWN1     DD  DSN=&DSN..&TAPE,           dataset sent as file
//              DISP=OLD,UNIT=&TAPE
//*
//UP1       DD  DSN=&DSN..&TAPE..UP,        uploaded file
//              DCB=(LRECL=133,RECFM=FB,BLKSIZE=0),
//              DISP=(NEW,CATLG),UNIT=&TAPE
//SYSIN     DD  *
-PARM=hdfs_protocol="direct";
emailall="me@example.com";
-DD_DOWN1=download.file;CONVERSION=BINARY;
-DD_UP1=download.file;CONVERSION=BINARY;
//*
```

In the JCL example above:

- The MDI:BDT tape esoteric is defined as MDITAPE
- The file to be sent from the mainframe is TEST. BIG.DATA
- The utility to be used is ICEGENER as it performs more efficiently than IEBGENER
- The retention period for the tape is zero days; the tape can be scratched as soon as the file transfer is complete
- The profile name is HDFS
- The DD name “DOWN1” refers to the data to be transferred to the MDI Platform and subse-

quently to the HDFS.

- The DD name “UP1” refers to output information to be brought back to the mainframe.
- The -PARM= parameter is coded in upper case and the value after the equals sign is coded in lower case. This parameter determines the protocol to be used for the ingestion into the HDFS. The value is surrounded by double quotes. This statement ends with a semicolon.
- The “emailall” keyword allows for email notifications to be sent. It is coded in lowercase as well as the value (email address) following the equals sign. This statement ends with a semicolon. Notice that the email address is surrounded by double quotes.
- The -DD_DOWN1= parameter is coded in uppercase and the value is the distributed name of the file once it has been moved to the MDI Platform, coded in lowercase. This statement ends with a semicolon followed by the CONVERSION keyword and value which is coded in uppercase. This statement also ends with a semicolon.
- The -DD_UP1= parameter is coded in uppercase and the value is the distributed name of the file on the MDI Platform to be moved to the mainframe, coded in lowercase. This statement ends with a semicolon followed by the CONVERSION keyword and value which is coded in uppercase. This statement also ends with a semicolon.

13.6.2 WebHDFS Protocol JCL Example

When **hdfs_protocol** is “**webhdfs**” then files are transferred using the **webHDFS REST API**.

```
//HDFS1      JOB   (ACCT),'HDFS',CLASS=A,MSGCLASS=W,
/*   RESTART=LUMXPROC.HDFS,
/*   NOTIFY=&SYSUID
/*
/*
/* 1) copy disk file to mditape
/* 2) HDFS profile copies file to Hadoop cluster
/* 3) retrieve data back to mainframe
/*
/*
// SET TAPE=MDITAPE           MDI tape
// SET DSN=TEST.BIG.DATA      disk dataset to be processed
/*
//COPY      EXEC PGM=ICEGENER      copy disk file to MDI tape
//SYSPRINT DD  SYSOUT=*
//SYSUT1    DD  DSN=&DSN,DISP=OLD
//SYSUT2    DD  DSN=&DSN..&TAPE,
//              DISP=(NEW,PASS),UNIT=&TAPE,RETPD=0
/*
//SYSIN     DD  DUMMY
/*
/*
// SET PROFILE=HDFS          Hadoop
/*
//HDFS      EXEC LUMXPROC,PROFILE=&PROFILE
//DOWN1     DD  DSN=&DSN..&TAPE,          dataset sent as file
//              DISP=OLD,UNIT=&TAPE
/*
//UP1       DD  DSN=&DSN..&TAPE..UP,      uploaded file
//              DCB=(LRECL=133,RECFM=FB,BLKSIZE=0),
//              DISP=(NEW,CATLG),UNIT=&TAPE
//SYSIN     DD  *
-PARM=hdfs_protocol="webhdfs";
emailall="me@example.com";
```

```
-DD_DOWN1=download.file;CONVERSION=BINARY;
-DD_UP1=download.file;CONVERSION=BINARY;
/*
```

In the JCL example above:

- The MDI:BDT tape esoteric is defined as MDITAPE
- The file to be sent from the mainframe is TEST. BIG.DATA
- The utility to be used is ICEGENER as it performs more efficiently than IEBGENER
- The retention period for the tape is zero days; the tape can be scratched as soon as the file transfer is complete
- The profile name is HDFS
- The DD name “DOWN1” refers to the data to be transferred to the MDI Platform and subsequently to the HDFS.
- The DD name “UP1” refers to output information to be brought back to the mainframe.
- The -PARM= parameter is coded in upper case and the value after the equals sign is coded in lower case. This parameter determines the protocol to be used for the ingestion into the HDFS. The value is surrounded by double quotes. This statement ends with a semicolon.
- The “emailall” keyword allows for email notifications to be sent. It is coded in lowercase as well as the value (email address) following the equals sign. This statement ends with a semicolon. Notice that the email address is surrounded by double quotes.
- The -DD_DOWN1= parameter is coded in uppercase and the value is the distributed name of the file once it has been moved to the MDI Platform, coded in lowercase. This statement ends with a semicolon followed by the CONVERSION keyword and value which is coded in uppercase. This statement also ends with a semicolon.
- The -DD_UP1= parameter is coded in uppercase and the value is the distributed name of the file on the MDI Platform to be moved to the mainframe, coded in lowercase. This statement ends with a semicolon followed by the CONVERSION keyword and value which is coded in uppercase. This statement also ends with a semicolon.

13.6.3 Azure JCL Example

Azure uses the **webhdfs** protocol but includes values for **hdfs_azure_application_key**, **hdfs_azure_application_id**, and **hdfs_azure_tenant_id**. In this example, the JCL is the same as HDFS, except, the appropriate Azure values are specified in the configuration file.

The Azure parameters can be specified in JCL. However, the application keys can be quite long and may lead to a line that is longer than 80 characters. If you have problems with the line length in your JCL, put the Azure parameters in the configuration file on the MDI host.

```
//HDFS1      JOB  (ACCT),'HDFS',CLASS=A,MSGCLASS=W,
/*  RESTART=LUMXPROC.HDFS,
//  NOTIFY=&SYSUID
/*
/*
/* 1) copy disk file to mditape
/* 2) HDFS profile copies file to Hadoop cluster
/* 3) retrieve data back to mainframe
/*
```



```

/*
// SET TAPE=MDITAPE           MDI tape
// SET DSN=TEST.PRINT.DATA     disk dataset to be processed
/*
//COPY      EXEC PGM=ICEGENER   copy disk file to MDI tape
//SYSPRINT DD  SYSOUT=*
//SYSUT1    DD  DSN=&DSN,DISP=OLD
//SYSUT2    DD  DSN=&DSN..&TAPE,
//           DISP=(NEW,PASS),UNIT=&TAPE,RETDP=0
/*
//SYSIN     DD  DUMMY
/*
/*
// SET PROFILE=AZURE          Hadoop
/*
//HDFS      EXEC LUMXPROC,PROFILE=&PROFILE
//DOWN1     DD  DSN=&DSN..&TAPE,           dataset sent as file
//           DISP=OLD,UNIT=&TAPE
/*
//UP1       DD  DSN=&DSN..&TAPE..UP,        uploaded file
//           DCB=(LRECL=133,RECFM=FB,BLKSIZE=0),
//           DISP=(NEW,CATLG),UNIT=&TAPE
//SYSIN     DD *
-PARM=hdfs_protocol="webhdfs";
emailall="me@example.com";
-DD_DOWN1=download.file;CONVERSION=BINARY;
-DD_UP1=download.file;CONVERSION=BINARY;
/*

```

In the JCL example above:

- The MDI:BDT tape esoteric is defined as MDITAPE
- The file to be sent from the mainframe is TEST.BIG.DATA
- The utility to be used is ICEGENER as it performs more efficiently than IEBGENER
- The retention period for the tape is zero days; the tape can be scratched as soon as the file transfer is complete
- The profile name is AZURE
- The DD name “DOWN1” refers to the data to be transferred to the MDI Platform and subsequently to the HDFS.
- The DD name “UP1” refers to output information to be brought back to the mainframe.
- The -PARM= parameter is coded in upper case and the value after the equals sign is coded in lower case. This parameter determines the protocol to be used for the ingestion into the HDFS. The value is surrounded by double quotes. This statement ends with a semicolon.
- The “emailall” keyword allows for email notifications to be sent. It is coded in lowercase as well as the value (email address) following the equals sign. This statement ends with a semicolon. Notice that the email address is surrounded by double quotes.
- The -DD_DOWN1= parameter is coded in uppercase and the value is the distributed name of the file once it has been moved to the MDI Platform, coded in lowercase. This statement ends with a semicolon followed by the CONVERSION keyword and value which is coded in uppercase. This statement also ends with a semicolon.
- The -DD_UP1= parameter is coded in uppercase and the value is the distributed name of the file on the MDI Platform to be moved to the mainframe, coded in lowercase. This statement

ends with a semicolon followed by the `CONVERSION` keyword and value which is coded in uppercase. This statement also ends with a semicolon.

13.7 Executing a Test Using BigData Transfer

After installation and configuration is complete, it's recommended to execute a PUT and GET operation, from one or more servers that you want to exchange data with, to test the product and ensure that all security provisions have been put in place. Using the JCL examples provided above, create the JCL required to execute a test. Submit the job(s) and check the job's SYSOUT for zero return codes. A non-zero return code indicates that there was a problem with the job's execution. See [Appendix A: Message Codes](#) for non-zero return code information.

If you would like assistance creating the JCL and executing the tests, or resolving a non-zero return code, contact the Luminex Support team.

13.8 Results

Any failure to transfer data to or from the MDI Platform results in an ABEND code associated with that transfer program. The entire batch job abends if any step fails. If the LUMXPROC step fails, an MDI message-code is returned, and the appropriate ABEND code delivered. See [Appendix A: Message Codes](#) for non-zero return code information and help resolving the issue.

If you would like assistance resolving a non-zero return code, contact the Luminex Support team.

13.9 Migration/JCL Conversion/Professional Services

Luminex provides Professional Services to help clients create/convert their JCL quickly and easily. Luminex JCL Conversion Utility can convert hundreds of JCL decks in just minutes. The conversion utility works with all Managed File Transfer solutions, FTP, FTPS or SFTP. For more information on Luminex JCL Conversion Professional Services, please contact your Luminex Sales Representative.

13.10 Scratch Tape Management

The Scratch List Update Utility or LSCRUP is a package provided to you by the Luminex Support Team. This package contains a LOAD module and sample JCL (see [Appendix C: LSCRUP Scratch List Update Utility](#)) for more information about installing the LSCRUP load module. LSCRUP is used to extract a list of scratch VOLSERS from your existing tape management report. A subsequent ICEGENER step then writes that scratch list to the MDI Platform. For more information about executing the LSCRUP scratch utility, see [Appendix C: LSCRUP Scratch List Update Utility](#).

13.11 BigData Transfer Security Requirements

The use of the mainframe program LUMXPROC to execute the BigData Transfer profile or profiles requires the setup of RACF permissions. The Security Administrator at your installation needs to define a FACILITY class profile or profiles to your security server and permit user ID(s) to the profile(s) in order to successfully execute LUMXPROC on your system. See the Mainframe Security Setup section of this guide for detailed information on security requirements.

13.12 Licensing

The Mainframe Data Integration solutions are licensed by MDI Platform. The use of more than one MDI Platform requires additional product licenses per Platform. Multiple product profiles can be deployed on a single MDI platform at no additional cost.

14. MDI SAS Language Processor (MDI:SLP)

14.1 Introduction

This section describes the Mainframe Data Integration SAS Language Processor (MDI:SLP). This solution allows clients to process programs written in the SAS language off the mainframe (off-host) while maintaining the mainframe as the “system of record” for SAS language processing. The solution uses mainframe JCL to move SAS language programs and input data to the MDI Platform where the SAS language programs are executed. The output can be returned back to the mainframe or it can be distributed locally from the MDI Platform.

MDI:SLP supports various types of SAS language workloads. A common use case is off-hosting Merrill Expanded Guide (MXG) processing. SAS language workloads that consume large amounts of cycles on the mainframe processor are good candidates for off-hosting.

Benefits include:

- Reducing or eliminating mainframe overhead for SAS language processing
- Reducing SAS language processing on the mainframe during peak processing times
- Providing for additional concurrent processing during heavy processing windows when mainframe cycles are constrained
- Reducing latency for data availability from SAS language workloads
- Providing an opportunity for simplifying and modernizing SAS language workloads
- Reducing mainframe software licensing costs for SAS language compilers

MDI:SLP uses FICON as the network and virtual tape as the API between the mainframe and the MDI:SLP Platform. No TCP/IP is used to transfer data from or to the mainframe. Storage that is made available to the MDI:SLP Platform is used to house data that is transferred. In some cases, such as with off-hosting MXG, storage is required to house one or more MXG Performance Database (PDB) and SMF data.

Using virtual tape as the API requires a tape range defined in your tape management system for tape VOLSERs used exclusively by MDI:SLP. It may also be convenient to assign a unique tape esoteric to your system for ease-of-use. In some use cases, the tape data is transient, meaning virtual tapes can have a short expiration period. Once expired, the virtual tapes can be re-used.

An ICEGENER or similar high-speed copy utility or program is used to copy the data to the virtual tape that is used by the MDI:SLP Platform for input. The Luminex mainframe batch program, LUMXPROC, opens the tape and passes information to MDI:SLP Platform for processing. Software on the MDI:SLP Platform copies the data to a Linux file. Other parameters=value pairs tell MDI:SLP how to process the file.

Output data is written back to the mainframe via virtual tape. Parameters in the JCL tell MDI:SLP which file or files to write back to the mainframe.

After all files are processed, the tapes are closed in the LUMXPROC step. The virtual tape labels

are in Standard Label (SL) format and may be processed as any other tape. With the exception of standard opening and closing, all reads or writes are processed on the MDI:SLP Platform, eliminating I/O and CPU overhead on the mainframe.

Any errors detected in the process forces the LUMXPROC step to ABEND with an informative code and detailed messages. The output should be analyzed, and the job restarted as necessary.

14.2 SAS Language Processor Components

14.2.1 Mainframe

- LUMXPROC load module
- LUMXPROC procedure
- Batch JCL
- SAF interface

14.2.2 MDI Platform

- Server code
- MDI:SLP software
- Client's customized profile definitions
- ASCII version of MXG software (on the MDI Platform if SAS compiler/interpreter resides there)
- SAS language programs (on the MDI Platform if SAS compiler/interpreter resides there)
- SAS language compiler (optionally SSH to a server with SAS language interpreter)

14.2.3 Optional SAS Language Server

- SAS language compiler/interpreter
- Client provided Open Systems storage for PDB build and SMF data virtual tape storage
- Optionally, Open Systems storage purchased with MDI:SLP

14.2.4 Storage

- Storage available on the MDI Platform (limited)
- Client provided Open Systems storage for PDB build and SMF data virtual tape storage
- Optionally, Open Systems storage purchased with MDI:SLP

14.2.5 MDI Reporting Interface

For reporting, a Graphical User Interface (GUI), is provided with MDI:SLP. The GUI provides a dashboard of various reports including:

- Data Movement (PUTs and GETs) in Process
- Performance Reports
- MDI Put Bytes Transferred
- MDI Get Bytes Transferred
- Storage Consumption and Availability

Also available via the GUI is the ability to generate reports that show trending over a period of

time:

- Performance Reports
- MDI Put Time
- MDI Get Time
- MDI Put Bytes Transferred
- MDI Get Bytes Transferred
- Storage Consumption and Availability

14.3 Planning for SLP

14.3.1 SLP Planning Overview

The software components for the MDI Solutions provide for communication over the FICON channel between the mainframe and the MDI Platform. The MDI Platform appears as a tape device to the mainframe, allowing for the transfer of mainframe data simply by writing data to MDI owned tape. For communication and data movement over FICON to occur, the batch interface components must be installed on the mainframe. MDI product code (SAS language profile) is also installed and configured for your site on the MDI Platform by the Luminex Support team.

The SAS language compiler/interpreter can reside on the MDI Platform or on a client provided server with SSH access to the MDI Platform. If on the MDI Platform, SAS programs, including MXG, must also reside on the MDI Platform. If on a client provided server, the SAS language programs and MXG need to reside on the client provided server.

When licensing a SAS language compiler/interpreter for the MDI Platform, a 16 Core Linux server license is required.

The following table describes the tasks necessary to begin using MDI SAS Language Processor in your environment:

	Task:	Refer to Section in this Guide:
1	Discovery	Section 14.3.2: Discovery Process Phase I; Section 14.3.3: Discovery Process Phase II
2	Sizing for Virtual Tape VOLSERS and Drives	Section 14.3.4: Sizing for Virtual Tape VOLS- ERS and Tape Drives
3	Architectural Design	Section 14.3.4.4: Architectural Consider- ations
4	Establishing Naming Conventions and Standards	Section 14.3.5: Naming Conventions for Di- rectories and Dataset Naming Standards
5	Product Installation on the mainframe	Section 6: MDI Mainframe Software Installa- tion and Configuration
6	Setup and execution of LSCRUP scratch update	Appendix C: LSCRUP Scratch List Update Utility

7	Understanding and configuring Security Requirements	Section 6: MDI Mainframe Software Installation and Configuration
8	Remote Execution	Section 14.3.6: Remote Execution of the SAS Language Compiler/Interpreter
9	SLP Profile Configuration	Section 14.3: Planning for SLP ; Section 5: MDI Profile Creation and the Installation Workbook ; Section 14.3.8: Profile Configuration
10	Understanding Data Conversion Options	Section 9: MDI Data Conversions
11	Understanding JCL Syntax Rules	Section 14.5.3.3.3: More Syntax Rules
12	Creating the JCL	Section 14.4: SLP JCL
10	Executing a test	Section 14.7: Executing a Test using SLP
11	Interpreting Messages and Codes	Appendix A: Message Codes

14.3.2 Discovery Process Phase I

Discovery is the process of identifying the SAS language programs in the client's environment and understanding which production/non-production jobs are being executed. The first phase of discovery is performed using an SMF analysis.

The SMF records needed for the first phase of discovery are type 30, 70, and 72's from all licensed LPARs. It's best to get a minimum of 3 weeks of these records during a busy SAS execution time period as an example; the week before month-end or quarter-end, the week of and the week after. Luminex performs a study that shows where SAS is being executed and creates CSV files that show the SAS jobs being executed.

In addition to the SMF data, the client needs to provide the following information for the MXG Use Case:

- Are you using the SMF logger to dump SMF records?
- How often do you dump SMF records?
- Are you segregating CICS, DB2, IMS, MVS, and other SMF records?
- How much SMF gets dumped, on average, during normal operations? In GB or TB
- How much SMF gets dumped during peak processing? In GB or TB
- What is your Service Level Agreement to have the MXG output available?
- Approximately how many GB are your PDB files on the mainframe and how long do you retain them?
- How long do you retain raw SMF data?
- Do you replicate the raw SMF data?
- What are the total storage requirements for other SAS language processing related data sets on your mainframe?

If your use case includes non-MXG related SAS language processing, the following information

is needed:

- What types of workloads are using SAS to produce output?
- Do any of your SAS programs update live DB2 or VSAM databases?
- What types of output are you producing? CSV, PDF, PDS files on the mainframe, etc.
- Approximately how many SAS language programs do you have?
- Approximately how many SAS users?

14.3.3 Discovery Process Phase II

Luminex provides a Professional Services offering to perform an in-depth assessment of the SAS language environment, convert SAS language programs to execute on the SLP co-processor and assist in the installation, setup and testing of the SAS language Processor solution. At a high-level, this service provides:

- Identification of all SAS language related assets, workloads, users, data libraries, code libraries and products.
- Installation, configuration and initial testing of the MDI:SLP environment.
- Reporting of system resource usage of SAS language related workloads in your environment.
- Identification of wasted resources that can be recovered through re-engineering or modernization
- Prioritization of jobs for migration; identification of jobs that can be migrated immediately
- Assistance in the migration design and project planning to achieve minimal impact on the production environment
- Testing assistance for all programs to be migrated.
- Constant monitoring during the production migration to ensure success

14.3.4 Sizing for Virtual Tape VOLSERs and Tape Drives

14.3.4.1 Number of Tape Drives to Define

The number of virtual tapes needs to be at least four times larger than initially expected. This is due to the fact that all input to the program and output from the program returned to the mainframe is via virtual tape. To size the virtual tape pool appropriately you need to determine the following metrics.

- How many production SAS jobs are executed daily?
- How many ad-hoc jobs are executed daily?
- How many input and output tapes need to be kept for some period of time after the job executes?

After you get the sum of the Production and ad-hoc numbers then multiple that number by 4. Most jobs have an input data set, a SASLIST output and a SASLOG output and possibly one or more additional inputs. Once you get the sum of all jobs multiple that by the number of days before you run a scratch update on your tape catalog +1. If you do the scratch update every day then multiple by at least two, if you do the scratch update every two days then multiple by at least 3, etc.

The absolute minimum number of virtual tapes needed = (sum of prod SAS jobs+ sum of ad-hoc jobs) * 4 * (number of days between scratch update +1) * # of days retention

Remember, virtual tapes only cost a UCB address so oversize rather than undersize. The absolute smallest number should be 16 per LPAR running SAS, but more is recommended.

14.3.4.2 Device Type

Tape devices must be defined as 3590 to accommodate all file sizes. Multi-volume tapes are not supported as input or output.

If a dataset exceeds the standard capacity of a 3590, the data size of the virtual tape can be enlarged in the MDI GUI. There may be situations where the tape must be split into multi volumes and processed individually.

Multi-file tapes are not supported either. The JCL must be compatible with (1,SL)

14.3.4.3 Device Esoteric

Assign a unique tape esoteric, such as MDITAPE, to be used in the JCL to direct mounts to the MDI:SLP co-processor. This assignment is part of the I/O Gen changes that occur at the time the MDI:SLP co-processor is installed and configured.

14.3.4.4 Architectural Considerations

The installation of the MDI:SLP solution allows for some consolidation of SAS language processing, however consideration needs to be made for how the input data and SMF data is transferred to the MDI:SLP Platform via virtual tape.

When virtual tape devices are installed on a LPAR within a collection of LPARs that share ICF catalogs, the movement of data is greatly simplified.

If, however, your production environment has non-shared ICF catalogs in a multi-LPAR environment, then defining the MDI:SLP Platform to share virtual tape devices on all non-shared LPARs simplifies the data movement.

In the case of separate physical locations such as a West Coast and East Coast data center, it's preferable that MDI:SLP Platforms are installed at each of the physical locations to handle local SAS language workloads and MXG PDB creation.

When consolidated reporting is required, clients typically find it easier to transfer the daily MXG PDB instead of transferring all the SMF data. This is a decision that must be made based on business need and bandwidth.

When there is a high volume of ad-hoc reporting, clients should consider an additional MDI:SLP Platform dedicated to this workload to keep unscheduled demand from potentially impacting regularly scheduled production jobs.

14.3.5 Naming Conventions for Directories and Dataset Naming Standards

Data sets are handled a little differently in Linux vs z/OS and these differences need to be considered before a migration takes place.

In z/OS datasets are named MYDATA.YOURDATA.OURDATA while in Linux that would be changed to /mydata/yourdata/ourdata.txt using directories instead of high level and midlevel name qualifiers.

For information on naming MXG subdirectories, please see and follow MXG's instructions.

For your SAS input and datasets, we recommend you plan the naming convention according to your business needs.

Below is an example of a two LPAR environment that keeps the data separate from the SAS code.

```
/sas/lpar1/code  
/sas/lpar1/data  
/sas/lpar2/code  
/sas/lpar2/data
```

The above structure stores the data apart from the SAS code. However, it allows for data to be easily aggregated as needed for whole datacenter reports.

When naming these directories, keep in mind you will need to type these names frequently, so it is recommended that you are distinct but brief. Links can be setup for important directories to make switching directories easier.

Within your data directory you may find it easier to keep CICS, DB2 and other z/OS data separate. An example of that may be:

```
/sas/lpar1/data/c/_forCICS  
/sas/lpar1/data/d/_forDB2  
/sas/lpar1/data/z/_fortherestofz/OSdata
```

Layout your data separation requirements carefully and then look for areas of simplification while still making it easier to understand.

14.3.6 Remote Execution of the SAS Language Compiler/Interpreter

Remote execution provides the ability to utilize a SAS language compiler/interpreter on another server. This server can be an existing Linux server with licensed software, or a new physical or virtual Linux server the client procures.

The size of the Linux server depends on how many SAS programs execute in parallel and how much data those programs process. While some SAS programs can parallelize some operations;

most SAS operations are executed serially. Typically, a single core per SAS program is used.

If performance issues are experienced, increasing memory may solve this problem. Larger memory allows more of the data set to be held in RAM. Otherwise, data is spilled to disk (or the server will start swapping). Going to disk slows down sorting and other operations that operate over the entire data set.

The MDI:SLP Platform requires access to the server with the licensed software. For remote execution, a user account and password on the system that runs the SAS compiler/interpreter is required. The user name and password is used to SSH to the SAS server used to execute the programs.

The MDI:SLP Platform must share an NFS mount with the SAS server that is executing the SAS code. This can be disk from the MDI shared with the SAS server or the SAS server sharing an NFS mount to the MDI.

14.3.7 Moving Partitioned Data Set (PDS) to the MDI:SLP Platform

SAS language programs and others are often contained in Partitioned Data Sets (PDS) on the mainframe. To move these PDS data sets to the MDI:SLP Platform, it's recommended to use the MXG SOURCLIB member JCLZERO to create a Physical Sequential (PS) file (of the PDS) on a MDI:SLP virtual tape and then use the MXG SOURCLIB member PROCSRCE to read the PS file on the MDI:SLP Platform.

14.3.8 Profile Configuration

MDI:SLP profiles are setup by the Luminex Support Team to provide desired functionality. Multiple profiles are supported for a variety of processing options. The mainframe JCL references the desired profile by profile name. Profile names are determined by the installation site.

When a SAS Language Processor profile is configured, operational arguments, such as the SAS language program, are provided by the mainframe JCL or configured in the profile on the MDI Platform. When operational arguments are configured in the profile, the same arguments must not be coded in the JCL. When operational arguments are found both in the JCL and in the SAS language profile on the MDI Platform, preconfigured parameters have priority over parameters set in the JCL.

Providing operational arguments in the JCL provides flexibility in the client's environment; several users can use the same profile with a variety of options. However, if the client's goal is to mandate certain operational arguments, hard-coding options in the SAS Language Processor profile on the MDI Platform achieves that goal.

The SAS Language Processor JCL section of this guide describes the parameters specific to the MDI:SLP profile.

14.3.9 Data Conversion from the Mainframe

Converting data from EBCDIC to ASCII or ASCII to EBCDIC is supported by the MDI:SLP

solution. Data written from the mainframe to the MDI:SLP platform has block and record format information associated with the data. The MDI:SLP data conversion program utilizes this information when performing the conversion. For information on Data Conversion keywords and value pairs, see the Data Conversion section of this manual.

14.4 SLP JCL

MDI:SLP uses JCL on the mainframe to copy the file(s) to the MDI:SLP Platform, via virtual tape. The JCL typically contains a two-step process.

Step 1 is copying the data set to be transferred from disk to MDI owned virtual tape. It's recommended that a copy utility, such as ICEGENER or if you are licensed for SYNCSORT, SYNCGENER be used in place of IEBGENER. These recommended copy utilities use buffering to execute faster and use less system overhead.

Step 2 Executes the LUMXPROC program passing parameters coded in the SYSIN DD card to the MDI Platform.

Note: If the data to be transferred has already been copied to MDI owned virtual tape, Step 1 can be eliminated.

Section 10: Syntax Rules for Parameters and KEYWORD=VALUE Pairs describe general syntax rules for coding parameters and keywords/value pairs in the SYSIN DD card in the JCL. Please review this section carefully as many of the values provided in the JCL must be coded in lower case or mixed case, as they are transferred to distributed platforms for processing. **It's recommended that the CAPS OFF command be issued in your ISPF editor when working with MDI related JCL to prevent lower case or mixed case values being automatically converted to upper case.**

14.4.1 Referencing SAS Language Programs in the JCL

A SAS language program consists of a series of steps which are executed for each row of the data set. Usually, DATA steps are first which define the structure of the data sets for the program.

A data set in SAS language is a table of "variables" and "observations". Variables are the columns of the table: i.e. each variable is a field in a record from the input. Observations are the rows of the table: i.e. each observation is a record from the input. The INFILE statement specifies the name of the input file and how to split the file into observations and variables.

The MDI:SLP profile looks for the INFILE statement on PUT jobs and copies the tape file to the filename specified in the INFILE statement. If there is no INFILE statement, then the SAS language program is executed but no data from the mainframe is copied.

PROC steps define what to do with the defined data set. A PROC step may contain an OUTPUT statement where the program output is written. On GET jobs, the MDI profile copies the output file specified in the OUTPUT statement to the virtual tape.

14.5 SLP JCL Parameters and KEYWORD=VALUE Pairs

The JCL can designate one or more files, or data sets, that can be transferred between the mainframe and the MDI:SLP Platform.

Additional parameters are passed to the MDI:SLP profile via the SYSIN DD * card in the mainframe JCL.

14.5.1 Parameters Set in the SLP Profile on the MDI Platform

Parameters set only in the MDI:SLP profile are as follows:

- slp_compiler – the path where the SAS language compiler/Interpreter is installed.
- Email address – One or more email addresses to send alert information
- Parallel Operation Count – The number of simultaneous profile actions per individual job. This value should always be 1.
- Direction – the data transfer direction is either a PUT or GET from the mainframe perspective.

Parameters that can be supplied by the JCL or pre-configured in the MDI:SLP profile are:

- slp_program – the path to a .sas file containing the SAS language program.

Preconfigured parameters have priority over parameters set in the JCL.

MDI:SLP profiles are installed and configured by the Luminex Support team in collaboration with the end user.

14.5.2 The -PARM Parameter

The -PARM parameter is required in the JCL. It is an indicator for the MDI SAS language profile configured on the MDI Platform that keywords and associated values are following.

The -PARM indicator is followed by a space or equal (=) sign.

There are 2 types of information provided after the -PARM parameter:

- 1) KEYWORD=VALUE pairs to define specific parameters for the MDI:SLP profile.
- 2) -DD_<dd name>= parameter describes the actions specific to the file associated with the DD statement.

-DD_<DD name> Parameter_<DD name>= parameter provides the name of the file(s) on the MDI:SLP server used to either write the file to (input) or send the file back (output) to the mainframe for the corresponding DD name.

The filename immediately follows the -DD_<DD name> value.

A -DD_ entry is required for each DD statement in the JCL. This defines either the input or output file name.

As an example, when two files are to be transferred in and out of the MDI:SLP platform for processing, the DD statements associated with the transfers could be:

```
//FILEIN DD DISP=OLD,DSN=PROD.MASTER.FILE1.MDI, dataset to be copied to Linux
// UNIT=&MDITAPE
//*
//FILEOUT DD DISP=(NEW,CATLG),DSN=PROD.REPORT.MDI, report retrieved after program
// UNIT=&MDITAPE executes
```

The associated -DD_ values are:

```
-DD_FILEIN=master.file1
-DD_FILEOUT=reporta.lst
```

14.5.3 Keywords & Values

All parameters are specified in the NAME=VALUE format, where NAME is a keyword that defines the VALUE.

Most keywords are unique to each MDI solution. Keywords can be coded in upper case, lower case or a mixture of upper and lower case. KEYWORD=VALUE pairs can be delimited by a space or semi-colon.

Every KEYWORD= must have a value. KEYWORD=VALUE cannot have spaces before or after the equal sign (=).

Below are all valid syntax:

```
login=userid;password=pass
LOGIN=userid PASSWORD=pass
Login=userid; PassWord=pass
```

Values

Values are the variables entered after the equals (=) sign of the KEYWORD=VALUE pair.

Values that have spaces must be double quoted. As an example:

```
PREACTION="REMOVEFILE /a/b/c"
```

Values with multiple entries require double quotes and, in some cases, comma separators. As an example:

Below are additional available keywords that are provided by SYSIN and after the -PARM keyword.

14.5.3.1.1 Special Handling of Conversion Keyword

The CONVERSION= value keyword is supported by all MDI solutions. It can be globally

defined for all files if coded after the -PARM parameter. Alternatively, each file can be converted differently by including this KEYWORD=VALUE on the -DD_<dd name> line.

For example, to globally define a conversion for all files, put the KEYWORD=VALUE on the -PARM line:

```
-PARM CONVERSION=ascii_LF;  
-DD_UPLOAD=uploadfile;  
-DD_DOWNLOAD=downloadfile;
```

Files associated with one or more -DD_<dd name>= parameters are converted using the conversion of ascii_LF,

To specify a different conversion for each file, define on the -DD_<dd name>= line:

```
-DD_UPLOAD=uploadfile; CONVERSION=ebcdic  
-DD_DOWNLOAD=downloadfile; CONVERSION=ascii_LF;
```

14.5.3.2.2 Comments

Lines in the SYSIN DD card can be commented by keying an asterisk (*) in column 1. In the example below, the preactions keyword will not be executed.

```
//SYSIN DD *  
-PARMS  
Login=mike  
Password=pass  
* preaction="listfile1 /a/b/c"  
-DD_UPLOAD=/a/b/c  
oformat=ascii_LF
```

14.5.3.3.3 More Syntax Rules

Values that contain multiple words must be double quoted. An example of this is shown in the “preaction” section below.

Lines should be kept short for readability. No line should run past column 71 or it may not be properly evaluated. KEYWORD=VALUE pairs can be placed on separate lines keeping in mind words cannot be split. Below is a complex example of valid syntax:

```
000054 -PARM destination=g7ftp10g parallel=1  
000055 login=programmer  
000056 password=password  
000057 emailall="users@example.com"  
000058 preaction="LISTFILEL testfile, LISTFILEL  
000060 'test print133 data'"  
000061 postaction="listfile1  
000062 'test print133 data'"  
000063 -DD_FBADOWN="'test print133 data'"  
000064 conversion=ascii_CRLF  
000065 ocode=nostrip  
000066 -DD_FBAUP=  
000067 "'test print133 data'" conversion=ebcdic_npc
```

14.5.3.4.4 slp_program

This parameter defines the name of the .sas file containing a SAS language program. The value of slp_program is passed to the compiler/interpreter using the command line interface (CLI).

```
slp_program=<sas file>
```

For example, given a SAS language program called jobs.sas:

```
slp_program=jobs.sas
```

This setting is mandatory in the JCL or MDI:SLP profile. If defined in the SLP profile, the value provided in the JCL is ignored.

14.5.3.5.5 slp_compiler

This parameter defines where the SAS language compiler/interpreter is installed.

```
slp_compiler=<path>
```

For example, if the SAS language compiler was installed in /opt/compiler-3.2.2:

```
slp_compiler=/opt/compiler-3.2.2/bin/compiler
```

This setting is mandatory in the MDI:SLP profile.

14.5.3.6.6 slp_user

This parameter defines the user to run the SAS program.

This is an optional parameter when the SAS program is run on the MDI server.

14.5.3.7.7 slp_group

This parameter defines the group to run on the SAS program under.

This is an optional parameter when the SAS program is run on the MDI server.

14.5.3.8.8 slp_default_memsize

This optional parameter can only be defined in the profile configuration on the MDI Platform.

The parameter defines the default amount of memory that the SAS compiler will use to run a job before spilling to disk.

We recommend not using more than 80% of the servers memory for SAS. This means that if you have 32GB installed on the server running the SAS programs and you will run up to 4 jobs concurrently then slp_default_memsize should be set to (32GB*80%)/4, or around 6G i.e.:

```
<slp_default_memsize type="String">10G</slp_default_memsize>
```

Note: the G is required and specifies GB. You can also use M in the memsize parameter to specify MB.

14.5.3.9.9 slp_memsize

This optional parameter can only be defined in the JCL.

The parameter is used to override slp_default_memsize. All recommendations listed above for slp_default_memsize applies here as well.

14.5.3.10.10 MDI:SLP Remote Parameters

This section describes the parameters for remote execution of SAS programs under MDI:SLP. These allow SAS programs to be executed on a different host than the MDI server.

If any of the parameters are missing – other than slp_remote_share – then SAS programs are executed on the MDI Platform. If the SAS language compiler/interpreter is not licensed for the MDI Platform, the execution fails.

14.5.3.11 slp_remote_host

The hostname or IP address where SAS language jobs are run. This parameter can be in JCL or the configuration file.

In the configuration file:

```
<slp_remote_host type="String">mysashost</slp_remote_host>
```

In the JCL:

```
-PARM slp_remote_host=mysashost
```

14.5.3.12 slp_remote_user

The user name to use when logging into the remote host. This parameter can be in JCL or the configuration file.

In the configuration file:

```
<slp_remote_user type="String">sasuser</slp_remote_user>
```

In the JCL:

```
-PARM slp_remote_user=sasuser
```

14.5.3.13 slp_remote_passwd

The password to use when logging into the remote host. This parameter can be in JCL or the configuration file. However, care must be taken with this parameter so that the password cannot be used by unauthorized users.

In the configuration file:

```
<slp_remote_password type="String">xxxxxxx</slp_remote_password>
```

In the JCL:

```
-PARAM slp_remote_password=xxxxxxx
```

14.5.3.14 slp_remote_wdir

The working directory on the remote host where job inputs and outputs are temporarily stored. This may be different than the MDI server working directory. This value can be in the configuration file or JCL.

In the configuration file:

```
<slp_remote_wdir type="String">/home/users/sasuser/work</slp_remote_wdir>
```

In the JCL:

```
-PARAM slp_remote_wdir="/home/users/sasuser/work"
```

14.5.3.15 slp_remote_share

A boolean value indicating whether or not the slp_remote_wdir and workDir (the MDI server working directory) refer to the same directory on shared storage. This can improve performance by removing the need to copy data between the MDI server and the remote SAS server. The default value is FALSE.

In the configuration file:

```
<slp_remote_share type="Boolean">TRUE</slp_remote_share>
```

In the JCL:

```
-PARAM slp_remote_share=TRUE
```

14.5.4 Email Setup

Emails can be defined in both JCL and the profile configuration on the MDI Platform. If defined in both places, the lists are merged and duplicates removed. If an error occurs, the email addresses defined in: emailonerror (JCL), emailall (JCL), email_error_list (configuration), and email_list (configuration) are merged and duplicates removed.

14.5.4.1 email_error_list

This optional parameter can only be defined in the profile configuration on the MDI Platform.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output

14.5.4.2 email_list

This optional parameter can only be defined in the profile configuration on the MDI Platform.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output.

14.5.4.3 emailonerror

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output. Multiple emails must be comma separated and double quoted. For example:

```
emailonerror="errors@example.com,alerts@luminex.com"
```

14.5.4.4 emailall

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receive an email of all job results. The body of the email contains identical information as the job SYSTPRINT output. Multiple emails must be comma separated and double quoted. For example:

```
emailall="user1@example.com,alerts@luminex.com"
```

```
EMAILALL="alerts@luminex.com, user@example.com, support@example.com"
```

14.5.4.5 slp_email_files

This optional parameter.

This is a comma separated, list of files that can be emailed upon successful completion of the SAS job. For example, if you are running a SAS job called job.sas which produces an output file called results.csv then you can email both the SAS LST and results.csv by including the following in your JCL:

```
slp_email_files="job.lst,results.csv"
```

14.5.4.6 slp_email_recipients

This optional parameter.

This is a comma separated, list of email addresses where slp_email_files will be sent. The syntax is the same as all of the other email parameters mentioned in the log.

14.5.5 Sample JCL for the MXG Use Case

The following JCL example shows how to dump SMF records from the system logger. Typically, the installation site has this process in place.

This is the first step in testing or using the MDI:SLP solution as data must first be made available to be executed on the MDI:SLP platform. If your use case is not MXG related, see the Data Migration Example JCL and PROCS section later in this guide.

```
/*  
// SET MDITAPE= MDI ESOTERIC  
/*  
/* DUMP SMF LOGSTREAM TO GDG FOR ARCHIVE  
/* DUMP SECOND COPY FOR MDI SLP  
/* DB2 OR CICS CAN BE COPIED TO THEIR OWN TAPES IN SIMILAR FASHION  
/*
```

```

//STEP1      EXEC PGM=IFASMF DL,REGION=0M
//OUTDD1     DD DSN=SMFDUMP.DUMP1(+1),
//            UNIT=PRODTAPE,          STANDARD PRODUCTION TAPE
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=32760,RECFM=VBS,BLKSIZE=0)
//*
//OUTDD2     DD DSN=SMFDUMP.SLP, THIS COULD BE GDG
//            LABEL=EXPDT=99000,
//            UNIT=&MDITAPE,          MDI TAPE
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=32760,RECFM=VBS,BLKSIZE=0)
//*
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD *
LSNAME (IFASMF.SYS1.DEFAULT,OPTIONS (ARCHIVE))
OUTDD (OUTDD1,TYPE (0:255))
OUTDD (OUTDD2,TYPE (0:255))
RELATIVEDATE (BYDAY,1,1)
/*

```

14.5.5.1 JCL Example to Build the MXG PDB

This sample uses JCL symbolics as introduced in z/OS 2.1 so parameters can be passes in SY-
SIN.

```

//SLPPDB JOB
//*
//
/* build pdb, email SYSPRINT report, print saslist and saslog
/*
/*
// JCLLIB ORDER=(MDI PROCLIB) change as necessary
/*
// EXPORT SYMLIST=(CUST,SMF) save variables
/* symbolics are used to simplify filenames
/*
// SET PROFILE=SLP
/*
// SET UNIT=MDITAPE change esoteric name as necessary
/*
// SET CUST=SAMPLE used to create filenames
/*
// SET SMF=dumped.smf input SMF on MDI TAPE
/*
/*
//SLP EXEC LUMXPROC,PROFILE=&PROFILE
/*
//XPROCLOG DD SYSOUT=* debugging messages
/*
//SMF DD DSN=&SMF,
// DISP=(OLD,DELETE),UNIT=&UNIT
/*
//SASLOG DD DISP=(,CATLG,CATLG), data can be valid
// DSN=TEST.&CUST..SASLOG, even if step abends
// LABEL=EXPDT=99000, keep as long as it is cataloged
// DCB=(LRECL=255,RECFM=VB,BLKSIZE=0),
// UNIT=&UNIT MDI esoteric
/*
//SASLIST DD DISP=(,CATLG,CATLG), data can be valid
// DSN=TEST.&CUST..SASLIST, even if step abends

```

```

//          LABEL=EXPDT=99000, keep as long as it is cataloged
//          DCB=(LRECL=255,RECFM=VB,BLKSIZE=0),
//          UNIT=&UNIT          MDI esoterics
// *
//SYSIN      DD  *,SYMBOLS=JCLONLY    pass parms from SET statement
*
*   program is the MXG PDB build code stored on server
-PARM slp_program=pdb.sas
*
*   always email results to programmer
*   this will be same output as in SYSPRINT
emailall="programmer@customer.com"
*
*   email results to operations in case of abend
emailonerror="operations@customer.com"
*
-DD_SMF=&SMF conversion=binary
*
-DD_SASLOG=&CUST..log conversion=ebcdic
-DD_SASLIST=&CUST..lst conversion=ebcdic
/*
// *
//PRINTLST EXEC PGM=ICEGENER,COND=EVEN    always print SASLIST
//SYSPRINT DD  DUMMY
//SYSIN      DD  DUMMY
//SYSUT1     DD  DSN=TEST.&CUST..SASLIST,
//            DISP=(OLD,DELETE),UNIT=&UNIT
// *
//SYSUT2     DD  SYSOUT=*
// *
// *
//PRINTLOG EXEC PGM=ICEGENER,COND=EVEN    always print SASLOG
//SYSPRINT DD  DUMMY
//SYSIN      DD  DUMMY
//SYSUT1     DD  DSN=TEST.&CUST..SASLOG,
//            DISP=(OLD,DELETE),UNIT=&UNIT
// *
//SYSUT2     DD  SYSOUT=*
// *

```

An alternative method emails all output including SASLOG and SASLIST. Tapes will not be required for SASLOG and SASLIST.

```

//SLPPDB JOB
// *
// *
// *   build pdb, email SYSPRINT , SASLOG and SASLIST
// *   to reduce number of tapes drives needed
// *
// JCLLIB ORDER=(MDI PROCLIB)    change as necessary
// *
// EXPORT SYMLIST=(CUST,SMF)    save variables
// *   symbolics are used to simplify filenames
// *
// SET      PROFILE=SLP
// *
// SET      UNIT=MDITAPE    change esoteric name as necessary
// *
// SET      CUST=SAMPLE    used to create filenames

```

```

/*
// SET      SMF=dumped.smf input SMF on MDI TAPE
/*
/*
//SLP      EXEC LUMXPROC,PROFILE=&PROFILE
/*
//XPROCLOG DD  SYSOUT=*          debugging messages
/*
//SMF      DD  DSN=&SMF,
//          DISP=(OLD,DELETE),UNIT=&UNIT
/*
//SYSIN    DD  *,SYMBOLS=JCLONLY   pass parms from SET statement
*
*  program is the MXG PDB build code stored on server
-PARM slp_program=pdb.sas
* files to email - SASLOG & SASLIST
slp_email_files="&CUST..lst,
&CUST..log"
*
slp_email_recipients="programmer@customer.com,
user1@customer.com"
*
* always email results to programmer
* this will be same output as in SYSPRINT
emailall="programmer@customer.com"
*
* email results to operations in case of abend
emailonerror="operations@customer.com"
*
-DD_SMF=&SMF conversion=binary
*
/*
/*

```

14.5.5.2 JCL Example to Execute a Report using MXG

JCL Example to Execute a Report using the Previously Created MXG PDB:

```

/* Request MXG report to run against previously created PDB
/* then print SASLIST and SASLOG
/* No data from the mainframe is required
/*
// SET  PROFILE=SLP          profile name
// SET  TAPE=MDI            MDI tape controller
/*
//XPROC  EXEC LUMXPROC,PROFILE=&PROFILE
/*
//SASLOG  DD DISP=(,CATLG),
//          LABEL=EXPDT=99000,   keep as long as cataloged
//          DSN=TEST.MDI.DAILY1.SASLOG,
//          DCB=(LRECL=255,RECFM=VB,BLKSIZE=0),
//          UNIT=&TAPE
//SASLIST DD DISP=(,CATLG),
//          LABEL=EXPDT=99000,   keep as long as cataloged
//          DSN=TEST.MDI.DAILY1.SASLIST,
//          DCB=(LRECL=255,RECFM=VB,BLKSIZE=0),
//          UNIT=&TAPE
//SYSIN   DD *
* program is daily1.sas
* it references PDB name

```

```

-PARM slp_program=/root/sas/daily1.sas
*
* name of SASLIST must match program name (daily1)
* upload file and convert to EBCDIC for printing
*
-DD_SASLIST=daily1.lst oformat=ebcdic
*
* name of SASLOG must match program name( daily1)
* upload file and convert to EBCDIC for printing
*
-DD_SASLOG=daily1.log oformat=ebcdic
/*
/*
//SASLIST EXEC PGM=ICEGENER,COND=EVEN print SASLIST
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
/*
//SYSUT1 DD DSN=TEST.MDI.DAILY1.SASLIST,
// DISP=(OLD,DELETE),UNIT=&TAPE
/*
//SYSUT2 DD SYSOUT=*
/*
//SASLOG EXEC PGM=ICEGENER,COND=EVEN Print SASLOG
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
/*
//SYSUT1 DD DSN=TEST.MDI.DAILY1.SASLOG,
// DISP=(OLD,DELETE),UNIT=&TAPE
/*
//SYSUT2 DD SYSOUT=*
/*

```

14.5.6 Data Migration Examples JCL and PROCs

14.5.6.1 SAS Language Program DATA SET

To get the SAS language program dataset ready for transmission to the MDI:SLP Platform, use the PROC CPORT job as shown below. This process allows you to use your existing PBD data by migrating PDBs on the mainframe to MDI.

This job has a cleanup step (IEFBR14) in the event it needs to be rerun, followed by the CPORT process that copies the SAS language program data set onto a virtual tape, then the CIMPORT job is executed to write the SAS language program data set into a directory on the MDI:SLP Platform.

Once you are comfortable with the execution of these jobs they can be combined into a single job or PROC and executed together.

```

//Your Job Card here
//CPORT EXEC SAS
//PDB DD DSN=HILVL.PDB.TEST(0),
// DISP=SHR
//CPORT DD DSN=HILVL.LUMINEX.SASPDB.CPORT,
// DISP=(,CATLG),
// UNIT=MDITAPE CHANGE THIS TO YOUR TAPE ESOTERIC FOR MDI
//SYSIN DD *

```

```
PROC CPORT LIB=PDB FILE=CPORT;
```

Once the CPORT job is complete and the data set is cataloged on MDI:SLP virtual tape you can execute the following CIMPORT job to create the SAS language program file on the MDI:SLP Platform.

```
//YOUR JOBCARD HERE
/* This is the CIMPORT process to read the data created by a CPORT job and
/* turn it back into a SAS dataset on the MDI:SLP.
/* The output of the previous
/* CPORT data must be on a MDI tape dataset
/*
/* 1) uncatalog datasets from prior runs
/* 2) copy source dataset from MDI tape
/* 3) exec LUMXPROC SLP to run SAS source
/* 3) print SASLOG output
/*
//PROCS JCLLIB ORDER=HILVL.LUMINEX.PROCLIB
/*
// SET  PROFILE=SLP
// SET  UNIT=CART90L                      MDI ESOTERIC
// SET  SASLOG=HILVL.LUMINEX.SASPDB.CPORTLOG LOG OUTPUT
// SET  SOURCE=HILVL.LUMINEX.SASPDB.CPORT   SOURCE INPUT
/*
/*
//CLEANUP  EXEC PGM=IEFBR14
//SASLOG   DD DISP=(MOD,DELETE),
//          DSN=&SASLOG,                      OUTPUT LOG
//          DCB=(LRECL=255,BLKSIZE=0,RECFM=VB),
//          UNIT=&UNIT
/*
//XPROC    EXEC LUMXPROC,PROFILE=&PROFILE
//XPROCLOG DD SYSOUT=*
//SOURCE   DD DISP=(OLD,KEEP),
//          DSN=&SOURCE,                      INPUT SOURCE
//          UNIT=&UNIT
/*
//SASLOG   DD DISP=(,CATLG,CATLG),
//          DSN=&SASLOG,                      OUTPUT LOG
//          EXPDT=99000,
//          DCB=(LRECL=255,BLKSIZE=0,RECFM=VB),
//          UNIT=&UNIT
//SYSIN    DD *
* cimutil.sas executes cimport and builds a PDB
-PARM slp_program=/luminex/storage/mxg/userid/cimutil.sas
slp_parms='lparname:HILVL.PDB.MYSASLIB!lpar1'
*
***LPAR1 is your LPAR name and the HILVL.PDB.MYSASLIB is whatever you want to name
**your dataset
*
-DD_SOURCE=cimutil.cpt
*
-DD_SASLOG=cimutil.log oformat=ebcdic
/*
/*
//SASLOG   EXEC PGM=ICEGENER,COND=EVEN
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=&SASLOG,
//          UNIT=&UNIT,
```



```
//          DISP=(OLD,DELETE)
//*
//SYSIN      DD DUMMY
//*
//SYSUT2     DD SYSOUT=*
//*
```

After the CPORT job is complete the PDB data is written to a virtual tape. To complete the PDB build run a cimport.sas job on MDI. Below is a template of the cimport.sas program referenced in the above JCL. This is a starting point for you to use to customize it to your business requirements if you are migrating a lot of existing SAS databases into the SLP appliance.

cimport.sas

```
-----
options source source2 mprint;
%macro utilcimp(
  dataset=&lparname,
);
  %let len=%eval(%length(&lparname)-2);
  %let dataset=%substr(%str(&lparname),2,&len);
  %let plex=%scan(%str(&dataset),2,%str(!));
  %let dataset=%upcase(%scan(%str(&dataset),1,%str(!)));
  %if %sysfunc(fileexist(/luminex/storage/&plex/&dataset))=0
    %then %do;
      X mkdir /luminex/storage/&plex/&dataset;
    %end;
  libname pdb "/luminex/storage/&plex/&dataset";
  proc cimport lib=pdb file='cimutil.cpt';
  proc datasets ddname=pdb;
  run;
%mend utilcimp;
-----
```

14.6 Important Locations on the SLP Platform

Use the following command to find the release level of the MDI:SLP Platform code:

```
ls /opt/luminex/cgx
```

Use the following command to view the SASLOG information from jobs executed on the MDI:SLP Platform.

```
/opt/luminex/MDITASK/current/var/logs
```

14.7 Executing a Test using SLP

After installation and configuration is complete, it's recommended to execute a test to ensure that all security provisions have been put in place. Using the JCL examples provided above, create the JCL required to execute a test. Submit the job(s) and check the job's SYSOUT for zero return codes. A non-zero return code indicates that there was a problem with the job's execution. See Appendix A. Messages and Codes for non-zero return code information.

If you would like assistance creating the JCL and executing the tests, or resolving a non-zero

return code, contact the Luminex Support team.

14.8 Results

Any file copied from the MDI:SLP Platform to a mainframe virtual tape data set can be processed as though it had been created by a mainframe application.

Any failure to transfer data to or from the MDI Platform results in an ABEND code associated with that transfer program. The entire batch job abends if any step fails. If the LUMXPROC step fails, an MDI message-code is returned, and the appropriate ABEND code delivered. See Appendix A. Message Codes for non-zero return code information and help resolving the issue.

If you would like assistance resolving a non-zero return code, contact the Luminex Support team.

14.9 SLP Requirements

The Security Administrator at your installation needs to define a FACILITY class profile or profiles to your security server and permit user ID(s) to the profile(s) in order to successfully execute LUMXPROC on your system. See the Mainframe Security Setup section of this guide for detailed information on security requirements.

14.10 Licensing

The MDI:SLP solution supports the SAS Institute interpreter/compiler which is licensed separately. Multiple MDI:SLP platforms may be necessary for high-volume processing and/or high availability and disaster recovery requirements. Each Platform requires MDI:SLP licensed software and SAS compiler/interpreter. MXG software is licensed by Merrill Consultants. If the installation site is licensed for MXG in the enterprise, there is no additional license required for the ASCII version of the software.

14.11 References

Lora D. Delwiche and Susan J. Slaughter. 2003. *The Little SAS Book*, 2nd Edition. SAS Publishing. Cary, North Carolina.

15. MDI zKonnnect

15.1 Introduction

The Mainframe Data Integration zKonnnect solution (zKonnnect) allows clients to send data, as a producer, from the mainframe to any Kafka topic. Clients can also send data from any Kafka topic to the mainframe using the File Watch and Pending Data Queue (PDQ) features described in section [Section 8: MDI File Watcher and Pending Data Queue \(PDQ\)](#).

Apache Kafka is an open-source stream-processing software platform that enables its clients to build real-time data pipelines and streaming applications. With an Apache Kafka streaming platform deployed, your organization can integrate data from a multitude of computing platforms into a single source for data consumers. The benefits of using Kafka are that it is horizontally scalable, fault-tolerant and provides excellent performance. Use of Kafka as a streaming platform provides for:

1. Publishing and subscribing to streams of data messages from multiple platforms
2. Storing and managing the data in a durable, fault-tolerant way
3. Processing of data streams continually, as they occur
4. Appending messages sent by any producer to any Kafka topic
5. Keeping order of the records so that consumers see the records in the correct order
6. Deployment of an n+1 server redundancy for recovery due to any type of outage

zKonnnect v1.0 supports writing files using a batch program, LUMXPROC, to a specified Kafka topic.

Benefits include:

- Reducing mainframe overhead for moving data from the mainframe to Kafka
- Moving data faster, at wire speed, over the FICON channel vs. TCP/IP protocols
- Providing for secure data movement from the mainframe to the Kafka topic
- Easy-to-Use and familiar batch utility program on the mainframe enables quick adoption
- Provides data movement tracking and auditing

15.2 Components

Mainframe

- LUMXPROC load modules
- LUMXPROC procedure
- Batch JCL
- SAF interface

MDI Platform

- Server code
- zKonnnect software

- Client's customized zKconnect profile(s)
- Kafka connector

Storage

- Client provided Open Systems storage assigned to Kafka cluster
- Storage for the MDI platform either provided by the client or purchased with zKconnect

MDI Reporting Interface

For reporting, a Graphical User Interface (GUI) is provided with zKconnect. The GUI provides a dashboard of various reports including:

- Data Movement in process
- Performance reporting
- MDI Bytes Transferred (PUT)
- MDI Bytes Received (via PDQ)
- Storage Consumption and Availability

15.3 Planning

The software components for the MDI Solutions provide for communication over the FICON channel between the mainframe and the MDI Platform. The MDI Platform appears as a tape device to the mainframe, allowing for the transfer of mainframe data simply by writing data to MDI owned tape. For communication and data movement over FICON to occur, the batch interface components must be installed on the mainframe. MDI product code (zKconnect profile) is also installed and configured for your site on the MDI Platform by the Luminex Support team.

The following table describes the tasks necessary to begin using MDI zKconnect in your environment:

	Task	Refer to Section in this Guide
1	Collect information on the Kafka topic(s) to send data to	Section 15.4: Collect Information on the Kafka Topic(s)
2	Collect information on the number of partitions per topic	Section 15.5: Collect Information on the Number of Partitions per Topic
3	Collect information on the IP address and port number for Zookeeper	Section 15.6: Collect Information on the IP address and Port Number for Zookeeper
4	Collect information on the IP addresses and port numbers for the brokers	Section 15.7: Collect Information on the IP addresses and Port Numbers for the Kafka Brokers
5	Collect information on the authentication requirements with Kafka; specifically, what scheme is to be used	Section 15.8: Collect Information on Authentication Requirements

	Task	Refer to Section in this Guide
6	Product installation on the mainframe	Section 6: MDI Mainframe Software Installation and Configuration
7	Planning for Virtual Tape Usage	Section 15.9: Planning for Virtual Tape Usage
8	Setup and execution of LSCRUP scratch update	Appendix C: LSCRUP Scratch List Update Utility
9	Understanding and configuring Security Requirements	Section 6: MDI Mainframe Software Installation and Configuration
10	zKconnect Profile Configuration	Section 5: MDI Profile Creation and the Installation Workbook ; Section 15.10: Profile Configuration
11	Understanding Data Conversion Options	Section 9: MDI Data Conversions
12	Understanding JCL Syntax Rules	Section 10: Syntax Rules for Parameters and KEYWORD=VALUE Pairs
13	Creating the JCL	Section 15.11: zKconnect JCL
14	Executing a test	Section 15.13: Executing a Test Using zKconnect
15	Interpreting Messages and Codes	Appendix A: Message Codes

15.4 Collect Information on the Kafka Topic(s)

Data from the mainframe is sent to a specific Kafka topic. The topic name is typically specified in the batch JCL to facilitate sending data to multiple Kafka topics. However, if the solution is being deployed to send data to 1 topic, it may be prudent to define the topic name in the zKconnect profile on the MDI Server.

During initial setup, one or more topics should be identified to test the setup/configuration. Once data is successfully being sent to the test topic and the process is verified, JCL can be used to send data to any Kafka topic.

An example of this parameter, typically indicated in the JCL is: `topic=mytopic`

A topic name can contain letters, digits, the underscore and periods. The topic must already exist. zKconnect does not create topics.

15.5 Collect Information on the Number of Partitions per Topic

Contact the Kafka administrator to determine if custom partitions are being used. This advanced feature is used for special balancing. If custom partitions are used, configure the zKconnect profile on the MDI Platform to indicate so. This parameter is seldom used.

An example of this parameter: `partition=3`

15.6 Collect Information on the IP address and Port Number for Zookeeper

Contact the Kafka administrator to determine the host name and port number where the instance for the Kafka cluster can be contacted. This is necessary to get the list of defined topics.

This keyword=value pair is typically defined in the zKconnect profile on the MDI Platform.

An example of this KEYWORD=VALUE pair: `zookeeper type=zookeeper1:2181`.
Where *zookeeper1* is the host name and 2181 is the port number.

15.7 Collect Information on the IP addresses and Port Numbers for the Kafka Brokers

Contact the Kafka administrator to collect the list of bootstrap Kafka brokers. The `broker_list` parameter is a comma separated list of HOST:PORT pairs. Generally, this list is defined in the zKconnect profile configuration on the MDI Platform.

An example of this KEYWORD=VALUE pair: `broker_list="kafka1:9092,kafka2:9092,kafka3:9092"`
Where *kafka1*, *kafka2* and *kafka3* are the broker names and 9092 is the port number.

15.8 Collect Information on Authentication Requirements

Contact your Kafka administrator to determine which authorization schema is in place on your Kafka cluster.

The most common schemas are 2-way TLS or SASL authentication. Kafka ACLs may also be in use.

For 2-way TLS, the zKconnect profile needs to be provided with the proper trust and keystores. Each MDI platform needs a copy of the trust and java key stores (.jks) files to be installed locally.

For SASL, the type of authentication needs to be defined, such as SASL PlainText or Kerberos. Depending on the authentication type, username and passwords, or keytabs and principals is also required.

15.9 Planning for Virtual Tape Usage

The MDI Platform appears as a virtual tape device to the mainframe. As part of the installation of the hardware, an IOGEN is performed on your z/OS system to add the MDI Platform as a peripheral device to the system. zKconnect uses virtual tape to transfer data from the mainframe to a Kafka topic. This requires that virtual tapes be defined to accommodate the data transfer. In addition, a tape scratch program (LSCRUP) must be executed daily to scratch expire tape volumes so they may be reused.

15.9.1 Number of Virtual Tapes and Drives to Define

The number of virtual tapes and tape drives to define is dependent on the number of files to be sent to Kafka daily and how long the data is to be retained. Typically, the data being sent to Kafka is transient and can expire within 24 hours. If the data is to be retained for some period of time, additional tapes will be needed.

Each file that is sent to Kafka uses one (1) virtual tape. If the data is not going to be retained beyond 24 hours, the number of volumes to define can be calculated by determining the number of files to be sent daily, times 2, plus additional tapes for expected growth. If the data is to be retained, the retention period must be factored in.

15.9.2 Device Type

Tape devices must be defined as 3590 to accommodate all file sizes. Multi-volume tapes are not supported as input or output.

If a dataset exceeds the standard capacity of a 3590, the data size of the virtual tape can be enlarged in the MDI GUI. There may be situations where the tape must be split into multi volumes and processed individually.

Multi-file tapes are not supported. The JCL must be compatible with (1,SL)

15.9.3 Device Esoteric

Assign a unique tape esoteric, such as MDITAPE, to be used in the JCL to direct mounts to zKconnect. This assignment is part of the IOGEN changes that occur at the time zKconnect is installed and configured.

15.10 Profile Configuration

zKconnect profiles are setup by the Luminex Support Team to provide desired functionality. Multiple profiles are supported for a variety of processing options. The mainframe JCL references the desired profile by profile name. Profile names are determined by the installation site.

When a zKconnect profile is configured, operational arguments, such as the Kafka topic name, are typically provided by the mainframe JCL. If only 1 topic is to be used, it can be configured in the profile on the MDI Platform. When operational arguments are configured in the profile, the same arguments must not be coded in the JCL. When operational arguments are found both in the JCL and in the zKconnect profile on the MDI Platform, preconfigured parameters have priority over parameters set in the JCL.

Providing operational arguments in the JCL provides flexibility in the client's environment; several users can use the same profile with a variety of options. However, if the client's goal is to mandate certain operational arguments, hard-coding options in the zKconnect profile on the MDI Platform achieves that goal.

The zKconnect JCL section of this guide describes the parameters specific to the zKconnect profile.

15.11 zKonnnect JCL

MDI zKonnnect uses JCL on the mainframe to copy the file to the MDI zKonnnect Platform, via virtual tape, and transmit the file to a destination. The JCL typically contains a two-step process. Step 1 is copying the data set to be transferred from disk to MDI owned virtual tape. It's recommended that a copy utility, such as ICEGENER or if you are licensed for SYNC SORT, SYNC GENER be used in place of IEBGENER. These recommended copy utilities use buffering to execute faster and use less system overhead.

Step 2 Executes the LUMXPROC program passing parameters coded in the SYSIN DD card to the MDI Platform. If the data to be transferred has already been copied to MDI owned virtual tape, Step 1 can be eliminated.

15.12 zKonnnect JCL Parameters and KEYWORD=VALUE Pairs

Parameters specific to the zKonnnect profile are designated in the SYSIN DD section of the JCL. To successfully execute a file transfer using the zKonnnect batch interface (LUMXPROC) the following information must be provided either in the profile definition on the MDI Platform (previously configured by Luminex Support team) or in the LUMXPROC step of the JCL.

15.12.1 The -PARM Parameter

The -PARM parameter is required in the JCL. It is an indicator for the MDI zKonnnect profile configured on the MDI Platform that keywords and associated values are following.

The -PARM indicator is followed by a space or equal (=) sign.

There are 2 types of information provided after the -PARM parameter:

- 1) KEYWORD=VALUE pairs to define specific parameters for the SecureTransfer profile.
- 2) -DD_<dd name>= parameter describes the actions specific to the file associated with the DD statement.

15.12.2 The -DD_<DD name> Parameter

This file name defines the name of the file written to Luminex virtual tape as part of the ICEGENER step. This is the name of the file that contains the data to be sent to the Kafka topic. The filename immediately follows the -DD_<DD name>= value. A -DD_<DD name>= value is required for each DD statement in the JCL. For example, if two files are to be transferred, the DD statements associated with the transfers could be:

```
//COPYFIL1 DD DISP=OLD,DSN=PROD.MASTER.FILE1.MDI
// UNIT=&OUTDEV
//*
//COPYFIL2 DD DISP=OLD,DSN=PROD.MASTER.FILE2.MDI,
// UNIT=&OUTDEV
```

The associated -DD_ values are:


```
-DD_COPYFIL1=PROD.MASTER.FILE1.MDI  
-DD_COPYFIL2=PROD.MASTER.FILE2.MDI
```

15.12.3 Keywords & Values

All parameters are specified in the NAME=VALUE format, where NAME is a keyword that defines the VALUE.

Most keywords are unique to each MDI solution. Keywords can be coded in upper case, lower case or a mixture of upper and lower case. KEYWORD=VALUE pairs can be delimited by a space or semi-colon.

Every KEYWORD= must have a value. KEYWORD=VALUE cannot have spaces before or after the equal sign (=).

Below are all valid syntax:

```
login=userid;password=pass  
LOGIN=userid PASSWORD=pass  
Login=userid; PassWord=pass
```

15.12.4 Values

Values are the variables entered after the equals (=) sign of the KEYWORD=VALUE pair. **Values that have spaces must be double quoted.** As an example:

```
PREACTION="REMOVEFILE /a/b/c"
```

15.12.5 Data Conversion

The CONVERSION= value keyword is supported by all MDI solutions. It can be globally defined for all files if coded after the -PARM parameter. Alternatively, each file can be converted differently by including this KEYWORD=VALUE on the -DD_<dd name> line.

For example, to globally define a conversion for all files, put the KEYWORD=VALUE on the -PARM line:

```
-PARM CONVERSION=ascii_LF;
```

To specify a different conversion for each file, define on the -DD_<dd name>= line:

```
-DD_UPLOAD=uploadfile; CONVERSION=ebcdic;  
-DD_DOWNLOAD=downloadfile; CONVERSION=ascii_LF;
```

15.12.6 Comments

Lines in the SYSIN DD card can be commented by keying an asterisk (*) in column 1. In the example below, the preactions keyword will not be executed.

```
//SYSIN DD *  
-PARMS
```

```

Login=mike
Password=pass
* preaction="listfile1 /a/b/c"
-DD_UPLOAD=/a/b/c
oformat=ascii_LF

```

15.12.7 More Syntax Rules

Values that contain multiple words must be double quoted.

Lines should be kept short for readability. No line should run past column 71 or it may not be properly evaluated. KEYWORD=VALUE pairs can be placed on separate lines keeping in mind words cannot be split. Below is a complex example of valid syntax:

```

000054 -PARM destination=g7ftp10g parallel=1
000055 login=programmer
000056 password=password
000057 emailall="users@example.com"
000063 -DD_FBADOWN="'test print133 data'"
000064 conversion=ascii_CRLF
000065 ocode=nostrip
000066 -DD_FBAUP=
000067 "'test print133 data'" conversion=ebcdic_npc

```

15.12.8 zKonnnect Supported Keywords

15.12.8.1 email_error_list

This optional parameter can only be defined in the profile configuration on the MDI Platform. This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output

15.12.8.2 email_list

This optional parameter can only be defined in the profile configuration on the MDI Platform. This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output.

15.12.8.3 emailonerror

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receives an email of any errors. The body of the email contains identical information as the job SYSPRINT output. Multiple emails must be comma separated and double quoted. For example:

```
emailonerror="errors@example.com,alerts@luminex.com"
```

15.12.8.4 emailall

This optional parameter can only be defined in the JCL.

This is a list of email addresses that receive an email of all job results. The body of the email contains identical information as the job SYSTPRINT output. Multiple emails must be comma

separated and double quoted. For example:

```
emailall="user1@example.com,alerts@luminex.com"  
EMAILALL="alerts@luminex.com, user@example.com, support@example.com"
```

Emails can be defined in both JCL and the profile configuration on the MDI Platform. If defined in both places, the lists are merged and duplicates removed. If an error occurs, the email addresses defined in: emailonerror (JCL), emailall (JCL), email_error_list (configuration), and email_list (configuration) are merged and duplicates removed.

15.12.8.5 zookeeper

The Host:Port pair where the Zookeeper instance for the Kafka cluster can be contacted. This is necessary to get the list of defined topics. Generally, this is defined in the profile configuration file on the MDI Platform. The host name and port number should come from the Kafka administrator.

```
<zookeeper type="String">zookeeper1:2181</zookeeper>
```

15.12.8.6 broker_list

The list of bootstrap Kafka brokers. The **broker_list** is a comma separated list of Host:Port pairs. Generally, this list is defined in the profile configuration file on the MDI Platform. The broker_list should come from the Kafka administrator.

```
<broker_list type="String">"kafka1:9092,kafka2:9092,kafka3:9092"</broker_list>
```

The default Kafka broker port number is 9092 when not using authentication. When using SSL/TLS authentication, the default port number is 9093.

15.12.8.7 topic

The Kafka topic where messages are written. Usually specified in the LUMXPROC JCL:

```
PARM topic="MYTOPIC"
```

A topic name can contain letters, digits, the underscore, and periods.

The topic must already exist. zKconnect cannot create topics that do not exist.

15.12.8.8 partition=

If using custom partitioning, the partition where messages will be written to. The partition value is an integer, often 1,2,3 and so forth. This is an advanced feature that is seldom used.

15.12.9 zKconnect JCL Example

```
//zKCONNECT JOB (YOUR ACCOUNT INFO),'USER INFO',  
// CLASS=X,MSGCLASS=X,  
// NOTIFY=&SYSUID,REGION=0M  
//  
//* SEND OUTPUT TO ZKCONNECT TOPICS USING LUMXPROC
```

```

/*
// SET      UNIT=MDITAPE          <= Unique to Client; Edit for
/*                               your site
// SET      PROFILE=ZKONNECT      <= Unique to Client; Edit for
//                               your site
/*
// JCLLIB ORDER=(SYSTEM.PROCLIB)  <= EDIT or Remove if not
/*                               needed
//GENER      EXEC PGM=ICEGENER
//SYSPRINT DD  SYSOUT=*
//SYSOUT DD  SYSOUT=*
//SYSIN DD  DUMMY
//SYSUT1 DD  DSN=INPUT.DATA.SET.NAME,      <=NAME OF INPUT FILE
//          DISP=SHR
/*
//SYSUT2 DD  DSN=OUTPUT.DATA.SET.NAME,      <= name of output tape
//          unit=&UNIT,
//          DISP=(,PASS),RETPD=001          <= change RETPD if
/*                                         desired
//XPROC      EXEC LUMXPROC,PROFILE=&PROFILE
/*
//XPROCLOG DD  SYSOUT=*
/*
//FILE1 DD  DSN=OUTPUT.DATA.SET.NAME,      <= same as SYSUT2 tape
//          DISP=OLD,
//          UNIT=&UNIT
/* NOTE: REMOVE COMMENTS THAT BEGIN WITH <= IN THE SYSIN BEFORE
/*                                         USE
//SYSIN DD  *
-PARM                                <= Required
TOPIC=zconnect_test_topic          <= Indicate name of KAFKA topic here
* this is a comment - the next PARM line is an example
* of the emailall keyword, it is optional
EMAILALL=" alerts@luminex.com, user@example.com, support@example.com "
-DD_FILE1=file.for.KAFKA          <= name of file used by Kafka topic
/*
//

```

15.13 Executing a Test Using zKconnect

After installation and configuration is complete, it's recommended to execute a test to one or more Kafka topics, to test the product and ensure that all security provisions have been put in place. Using the JCL examples provided above, create the JCL required to execute a test. Submit the job(s) and check the job's SYSOUT for zero return codes. A non-zero return code indicates that there was a problem with the job's execution. See [Appendix A: Message Codes](#) for non-zero return code information.

If you would like assistance creating the JCL and executing the tests, or resolving a non-zero return code, contact the Luminex Support team.

15.13.1 Results

Any failure to transfer data to or from the MDI Platform results in an ABEND code associated with that transfer program. The entire batch job abends if any step fails. If the LUMXPROC step fails, an MDI message-code is returned, and the appropriate ABEND code delivered. See [Appendix A: Message Codes](#) for non-zero return code information and help resolving the issue. If you

would like assistance resolving a non-zero return code, contact the Luminex Support team.

15.14 Migration/Conversion/Professional Services

zKconnect requires JCL, keyword and parameter requirements. If your site is planning on moving a large number of files from the mainframe to Kafka, Professional Services may be helpful to complete a large project. For more information on Luminex JCL Conversion Professional Services, please contact your Luminex Sales Representative.

15.15 Scratch Management

The Scratch List Update Utility or LSCRUP is a package provided to you by the Luminex Support Team. This package contains a LOAD module and sample JCL (see [Scratch Update Processing LSCRUP Load Module](#) in [Section 6.1: Installation Checklist](#) for more information about installing the LSCRUP load module). LSCRUP is used to extract a list of scratch VOLSERs from your existing tape management report. A subsequent ICEGENER step then writes that scratch list to the MDI Platform. For more information about executing the LSCRUP scratch utility, see [Appendix C: LSCRUP Scratch List Update Utility](#).

15.16 Security Requirements

The use of the mainframe program LUMXPROC to execute the zKconnect profile requires the setup of RACF permissions. The Security Administrator at your installation needs to define a FACILITY class profile or profiles to your security server and permit user ID(s) to the profile(s) in order to successfully execute LUMXPROC on your system. See [Section 7: Mainframe Security Setup](#) for detailed information on security requirements.

15.16.1 Security Example for zKconnect Definition

zKconnect security setup is different from the other profiles in that there are only PUTs and no GETs. zKconnect sends data to the Kafka cluster only. FileWatch/PDQ can be configured to send output from Kafka to the mainframe.

In addition, permissions as to who can write to a Kafka topic are handled within the Kafka environment. This means that RACF can only control who can submit the LUMXPROC procedure for any zKconnect profile. As a reminder, profile names are created by the client and configured on the MDI Platform.

As a result, to permit all users to be able to execute LUMXPROC and send data to all Kafka topics the RACF profile setup can be simplified to:

```
LUMXPROC.ZBUS.**
```

If it is desired to protect individual zKconnect profiles so that only certain users can use any particular zKconnect profile, the following RACF profile can be used:

```
LUMXPROC.ZBUS.*.KAFKA1
```

Where KAFKA1 is the name of the profile configured on the MDI Platform (client specified).

The generic use of the asterisk is used in place of PUT, GET or BOTH.

Scenario #1:

MDI:zKonnnect profile TOKAFKA is configured to transfer the site's batch output to Kafka topics. User IDs permitted to this profile can send data from the mainframe to Kafka. The Data Analyst group (DKTAL, DKTKK, DKTYK) and the site's scheduling system (OPER) can use this profile.

RACF Security Definition Examples for Scenario #1:

```
RDEFINE FACILITY LUMXPROC.ZBUS.** OWNER(MDIADMIN) UACC(NONE)
SETR RACLIST(FACILITY) REFRESH
PERMIT LUMXPROC.ZBUS.*.TOKAFKA CLASS(FACILITY) ID(DKTAL) ACCESS(READ)
PERMIT LUMXPROC.ZBUS.*.TOKAFKA CLASS(FACILITY) ID(DKTKK) ACCESS(READ)
PERMIT LUMXPROC.ZBUS.*.TOKAFKA CLASS(FACILITY) ID(DKTYK) ACCESS(READ)
PERMIT LUMXPROC.ZBUS.*.TOKAFKA CLASS(FACILITY) ID(OPER) ACCESS(READ)
```

15.17 Licensing

The Mainframe Data Integration solutions are licensed by MDI Platform. The use of more than one MDI Platform requires additional product licenses per Platform. Multiple product profiles can be deployed on a single MDI platform at no additional cost.

Appendix A. Message Codes

MDI Code	Source	Description	Sample Message	Action
MDI001I	mdi_dispatch	Transaction ID	Transaction ID: 1803122903098711	-
MDI003I	mdi_dispatch	Successful conversion to tape results	VOLSER: AC0022 File: test.print137.data.VBA Conversion Name: binary Tape Format: VB Total Bytes Converted: 20334 Total Blocks Processed: 2 Total Records Processed: 442 Dataset: TEST.PRINT137. DATA.CU9300 DD name: VBAUP Seconds to Convert: 2	-
MDI004I	mdi_dispatch	Successful conversion from tape results	VOLSER: AC0019 File: test.print137.data.VBA. ascii Conversion Name: ascii_LF Tape Format: VB Total Bytes Converted: 19206 Total Blocks Processed: 1 Total Records Processed: 0 Dataset: TEST.PRINT137. DATA.CU9300 DD name: VBADOWN Seconds to Convert: 1	-
MDI005I	mdi_dispatch	Transport time to execute	Put Transport took 1 seconds	-
MDI006I	mdi_dispatch	Transport details	test.print133.data.fba.ascii transferred 1562762 bytes from host ADCDPL.MVS2. LZG2 in 1 seconds	-
MDI007I	Dispatch	Total Time for all transports	Total time: 5 seconds	
MDI100T	Dispatch	JCL parsing error of LUMXPROC arguments	<Unique message from syntax checking>	Call Luminex
MDI101T	SLP	Missing required JCL parameter	Missing JOBID parameter	Call Luminex
MDI102T	SLP	Missing required parameter for profile	<Unique message that will specific missing parameter>	Declare missing parameter in JCL

MDI104T	MDITASK	Profile does not exist or is malformed	<Message on profile problem>	Call Luminex
MDI105T	MDITASK	Profile not licensed	<Message of licensing issue>	Call Luminex for license
MDI106T	MDITASK	Invalid number of parameters from LUMX-PROC	<Message of missing parameters>	Call Luminex
MDI107T	mdi_dispatch	Failed to execute GET dispatch program	Failed to transfer <file name> using <transfer program>	See previous messages.
MDI108T	MDITASK	Unexpected dispatch failure	PROFILE=<profile> Unexpected abort running profile	Call Luminex
MDI109T	SFTP	SFTP required parameters not specified in JCL or configuration	<Description of missing parameter>	Specify missing parameter in JCL
MDI110T	MDITASK	Failed to dispatch profile	PROFILE=<profile> Failed to dispatch profile	Call Luminex
MDI111T	SFTP	lftp installation or configuration error	<Unique message on configuration issue>	Call Luminex
MDI113I	SFTP	SFTP success	<Success messages>	-
MDI114T	SFTP	SFTP transfer failure	<Error Messages>	Look at previous messages for errors in transfer
MDI115T	SFTP	Failure to transfer log file	<Error messages>	Resolve error in message
MDI116I	SFTP	Successful transfer of log file	Successful log update to <directory>	-
MDI200I	SFTP	pre/postaction success results	<Unique success messages>	-
MDI201W	SFTP	pre/postaction warning error results	<Unique Warning messages>	Ignore if expected or Call Luminex
MDI202T	SFTP	pre/postaction failure results	<Unique Error messages>	Call Luminex
MDI203T	mdi_dispatch	Conversion to tape failed	Creation of tape <VOLSER> from <file name> failed	Determine error from messages. Call Luminex
MDI204T	mdi_dispatch	Conversion from tape failed	Conversion of tape <VOLSER> to <file name> failed	Determine error from messages. Call Luminex

MDI210I	SLP	General success messages from profile	<Unique success messages>	-
MDI212T	SLP	General profile execution error	<Unique error message>	Call Luminex
MDI213T	MDITASK	LUMXPROC version not compatible	Lumxproc version <version number> is not compatible with MDI server code	Call Luminex
MDI213T	SFTP	Failed to ping SFTP host	<SFTP host> does not respond to ping	Verify JCL hostname or IP. Verify SFTP host is responding.
MDI980T	LUMXPROC	RACF authorization failure	MDI980T JOB NOT AUTHORIZED FOR PROFILE: MDIX	Check RACF permissions for Profile/ UserID
MDI981T	Dispatch	Unknown Profile	Profile MDIX not found	Check profile declaration in JCL for spelling.
MDI982T	Dispatch	Syntax Error parsing JCL arguments	Syntax Error	Review JCL for syntax in the SYSIN area
MDI983T	Dispatch			
MDI984T	Dispatch	Authorization failed	<Unique message dependent on authorization failure>	Call Luminex
MDI985T	Dispatch	Any fatal error processing	MDI982T Failed to execute profile	See additional MDI codes and messages for failure details
MDI990T	LUMXPROC	LUMXPROC is not compatible with MDI server code	LUMXPROC is back level or incompatible with this equipment	Update LUMXPROC
MDI991T	LUMXPROC	JCL PARM statement error	JCL statement PARM= error	Verify -PARM parameter exists in JCL
MDI992T	LUMXPROC	JCL DD statement error		Verify DD statement in JCL

MDI995T	LUMXPROC	LUMXPROC/MDI server communication failure	MDI Control CCW failed	Verify device is not boxed. Call Luminex
MDI996T	LUMXPROC	LUMXPROC/MDI server communication failure	CONNECTION RESET – NO ENDING STATUS AVAILABLE	Verify device is not boxed. Call Luminex
MDI997T	LUMXPROC	LUMXPROC/MDI server communication failure	NO ENDING STATUS PROVIDED BY PROFILE	Verify device is not boxed. Call Luminex
MDI999I	Dispatch	Execution successful	Success	-

SYSPRINT

Various processing messages and the final status of MDI processing are printed in the SYSPRINT DD output. If the processing is not successful, then the LUMXPROC step ABENDs with a user code that matches the final status.

The final message for successful processing produces a message ID of MDI998I (if non-fatal issues) or MDI999I (success).

The LUMXPROC program has the following reserved message IDs for conditions it detects.

Type of Error	Message ID	User Abend Code
JOB NOT AUTHORIZED FOR PROFILE:	MDI980T	980
LUMXPROC is back-level or incompatible with this equipment	MDI990T	990
JCL EXEC statement PARM= error	MDI991T	991
JCL DD statement error	MDI992T	992
MDI control CCW failed	MDI995T	995
CONNECTION RESET – NO ENDING STATUS AVAILABLE	MDI996T	996
NO ENDING STATUS PROVIDED BY PROFILE:	MDI997T	997

The final message ID for any other processing error must be one of the following.

Type of Error	Message ID	User Abend Code
Unknown/unsupported Profile	MDI981T	981
Parm/syntax error	MDI982T	982
Input-related processing failure	MDI983T	983
External-security failure	MDI984T	984
MDI processing failure	MDI985T	985
External storage failure/noaccess	MDI986T	986

Appendix B. Security Specifications

1. PARMLIB requirements – SYS1.PARMLIB entries
 - a. No SYS1.PARMLIB entries are installed or required by this product other than APF authorization (SYS1.PARMLIB (IEAAPFxx)).
 - b. There is a requirement for a parm file, but it can be placed anywhere that can be read by the started task PROC.
2. Started task requirements
 - a. What attributes?
 - i. No PPT entries are required.
 - ii. The LUMXPDQ program should be TRUSTED(NO) since it honors resource definitions and permissions.
 - iii. The LUMXPDQ program must run AUTHORIZED.
 - iv. The LUMXPDQ program uses RACROUTE REQUEST=VERIFY which may require an entry in a security system authorized caller table.
 - b. What permissions?
 - i. For each request to create a new tape data set, the identity of the requesting user (userid) is securely passed to the MDI server. An associated PassTicket is generated on the server. The userid, PassTicket, and name of the requested new tape data set name is passed to LUMXPDQ on the mainframe. LUMXPDQ uses RACROUTE REQUEST=VERIFY (with a given APPL) to create an ACEE and verify that the user is valid. Then LUMXPDQ uses that ACEE with RACROUTE REQUEST=DEFINE CLASS='DATASET' TYPE=DEFINE GENERIC=YES to verify that the given userid is authorized to create the specified new tape data set. LUMXPDQ then allocates the new tape data set on a Luminex tape drive. The ACEE is only used to verify that the requesting user has permission to create the specified data set name. Therefore, LUMXPDQ itself does not need to be given permission into the profiles governing the requested new tape data set names.
 - ii. No data set permission is needed for the LUMXPDQ program other than access to the PARMFILE, MDIHUB, and STEPLIB data sets described in the next section.
3. Dataset security – What are the product datasets and what are the suggested access requirements.
 - a. PARMFILE: started task PROC needs READ access.

- b. COMMxxxx: One tape communications dataset on each MDI server (xxxx is the device number). The started task PROC should be given UPDATE access to this standard label tape data set. This data set is only used to establish an EXCP environment for communicating with the MDI server. After open processing is completed, the LUMXPDQ program uses non-motion commands to interact with the Luminex MDI server through the allocated tape drive.
- c. STEPLIB: the LUMXPDQ executable module needs to be APF authorized so that it can access the NED information for each tape drive.
- d. PDQ00001-PDQ99999: These are the DD names that are dynamically allocated by PDQ whenever a new tape data set is allocated on the Luminex MDI server. The lowest available DD name is used for each new allocation. The maximum number of allocated tape data sets depends on the list of drives defined in the PDQ parameters and whether those drives are online. The authority to create a particular tape data set is based on the permissions of the requesting userid. If the data set name already exists, the request is rejected prior to allocating a tape DD name.

4. General resource security

- a. Does this product have panels, commands, transactions, or any other resources that need protection?
 - i. This product does not have any foreground commands or programs.
 - ii. LUMXPDQ allows some operator to modify “F” commands.
- a. List the class name(s) and resources along with some recommended permissions.
 - i. The name of the PDQ queue must be defined as a FACILITY profile and permitted READ access by the PDQ task. The first two qualifiers are “LUMXPDQ.CONNECT”. The queue name is the third qualifier. LUMXPDQ will terminate if the profile does not exist or if not permitted to READ this profile. See [Section 7.7.3: Data Set Name](#) for further explanation.
 - ii. New tape data set names that are not controlled by a data set profile will not be allowed unless an additional FACILITY profile is created. The purpose of this profile is simply to allow the use of data sets that have no governing data set profile. At most sites, this would be undesirable. But if LUMXPDQ should allow such data sets to be created, the first two qualifiers of the FACILITY profile must be “LUMXPDQ.DSNOPROF”. The third qualifier must be the PDQ queue name. See [Section 7.7: PDQ Security Definitions](#) for further explanation.
 - iii. PassTicket requires the PTKTDATA class. The first qualifier must be the application ID of the Luminex MDI file watcher program, such as MDIWFIL. The second qualifier must be the userid that is authorized to create data sets on the mainframe. Included with this resource definition must be the SSIGNON(KEYMASKED(___)) parm in which the

16-hex-digit pass ticket token for that userid must be specified. See [Section 7.7.3: Data Set Name](#) for further explanation.

1. Product configuration settings affecting security

- a. What is the dataset/member [of any file that contains settings/flags/parms/options]?
 - i. The only parameter file is defined in the started task PROC and uses the PARMFILE DD statement.
- b. What are the options affecting security [or risks] and their default values [including hidden/un-documented options]?
 - i. There are no security options that can be set or controlled by any input, parm, or command. All security settings are controlled by calls to the SAF router.
 - ii. There are diagnostic commands that can turn on displays of EXCP traffic to and from the PDQ control file. This would include the current pass ticket, which could also be displayed using a GTF IO trace. However, the pass ticket value can only be used once and is immediately used by LUMXPDQ, which effectively prevents further use.

2. Additional Security Considerations

- a. Example, are exits required or optional?
 - i. At this time, there are no EXITs introduced by this product.
 - ii. No system EXITs are used by this product. There are no SVCs introduced by this product. There are no cross-memory services introduced by this product. There are no structures or control blocks created in system memory by this product.
 - iii. The ACEE created by RACROUTE REQUEST=VERIFY is retained in LUMXPDQ local address space memory for subsequent requests. Each subsequent request is also tested with RACROUTE REQUEST=VERIFY using the ACEE that corresponds to the given userid.
- b. Are there any default userid/passwords that should be changed?
 - i. There are no userids/passwords embedded in this product. There are no default passwords and no default userids.
- c. Are there USS requirements not covered by the previous items?
 - i. This product does not use any UNIX System Services in the mainframe.

3. Program Summary

- a. The LUMXPDQ program needs to be APF authorized. It switches to key zero supervisor state only while running the IOSCDR macro. This is needed to look at the Configuration Data Records (NED information) from each tape drive that is to be dynamically allocated. If the tape drive is not Luminex, the allocation is not attempted.
- b. The LUMXPDQ program uses RACROUTE REQUEST=VERIFY to check that the specified userid is allowed for PDQ processing. The application ID must be specified as a “APPID=” parm on the EXEC JCL statement. This application ID must be the same as the application ID in the MDI file watcher program that runs on the MDI server. The Luminex File Watcher uses an application ID of “MDIWFILE”.
- c. After the requesting userid has been authenticated via Pass Ticket, the LUMXPDQ program uses RACROUTE REQUEST=DEFINE CLASS='DATASET' to verify that this userid is authorized to create the specified new tape data set. If authorized, then LUMXPDQ will perform a dynamic allocation of that data set name on a Luminex MDI tape drive. After the data set has been built by the Luminex MDI server, the tape allocation will be freed, and the data set will be cataloged.

Appendix C. LSCRUP Scratch List Update Utility

LSCRUP is used to extract a list of scratch VOLSERs from your existing tape management report. A subsequent ICEGENER step then writes that scratch list to the MDI Platform. If the Platform has been defined as an MTL (manual tape library) and if your tape management system does not update the TDCB, then you must also generate IDCAMS ALTER VOLUMEENTRY statements and update the TCDB. The TCDB is also known as: tape catalog data base, SYS1.VGENERAL, or SMS/OAM VOLCAT. The sample JCL for scratch update without the TCDB update is LSCRVTJ, JCL with the TCDB update is LSCRZTJ.

The package contains a loadlib and samplib. They are provided in AMATERSE PACK format and also in TSO XMIT files so that you can look at them before uploading the samplib to the mainframe.

Both of the sample jobs have an EOJ statement (//) that prevents the actual update steps (ALTER-VOL, SERVUPD) from executing. You must remove this EOJ after you have finished testing the EXTRACT step, or else the scratch list will not be updated.

The MDI Platform default configuration is for a single scratch pool. This scratch pool is updated whenever a list of scratch VOLSERs is writing to a special tape VOLSER that is assigned to that scratch pool. This default special tape VOLSER on the MDI Platform is SCRTAP. If there are multiple scratch pools or your tape management system will not support this VOLSER, then SCRTAP is not appropriate.

One possible convention is to use the lowest VOLSER of the range of VOLSERs defined for the MDI Platform. For example, if VOLSERs 500000 through 699999 were assigned to the MDI Platform, then it would make sense for VOLSER 500000 to be designated as the special scratch update VOLSER. This means that VOLSER 500000 would have to be set aside with a data set that never expires so that the VOLSER never goes scratch. This data set name would then be specified in the SERVUPD SYSUT2 DD statement with a disposition of OLD, KEEP. It is best to set reserve this VOLSER as non-scratch prior to using the new range of VOLSERs on the MDI Platform, just to avoid losing it to some other tape data set.

If you have a disaster recovery plan that has a separate set of output VOLSERs for the DR site, then you might want to set up a separate LSCRUP job for those VOLSERs.

Appendix D. Program Characteristics and Requirements

The LUMXPROC program has the following features, characteristics, requirements, and restrictions.

- The SYSPRINT DD is required so that processing messages and errors can be reported.
- The SYSIN DD is read and any text is passed to the MDI Platform for evaluation and processing. The SYSIN DD may contain parameters and syntax that contains lower case letters.
- The LUMXPROC program ABENDs with a user abend code if the MDI processing was not completed successfully.
- To check the parameters and arguments that are being passed from LUMXPROC to the MDI profile, include an XPROCLOG DD statement (SYSOUT=*) in the LUMXPROC step.
- The TIOT is scanned for DD names. Only DD names associated with MDI Platform virtual tape drives are passed to the MDI processing. If a VOLSER prefix has been specified, then only VOLSERs having that prefix are passed to the MDI processing.
- All MDI tape files must use standard labels (SL).
- Only the first data set on a DISP=OLD virtual tape VOLSER should be specified in the corresponding DD statement of the LUMXPROC step. Support for subsequent data sets on the DISP=OLD VOLSER depends on the MDI profile, but this is not currently supported.
- **DISP=NEW files are opened for mainframe output and then closed, resulting in a null file but valid labels.** The MDI processing uses the (partial) data set name from these labels when generating the DISP=NEW file. After completion of the MDI processing, the DISP=NEW file contains the data created by the MDI processing.
- Multiple data sets on a DISP=NEW virtual tape VOLSER are not supported by MDI. There can only be one file per VOLSER.
- For each DISP=NEW tape data set, a scratch tape is mounted even if the allocated tape drive is not an MDI Platform virtual tape drive. These non-MDI scratch tapes cannot be accessed by the MDI Platform and therefore remain empty.
- The LUMXPROC program does not use any privileged features of z/OS.
- The LUMXPROC program does not use cross memory services.
- There are no TCP/IP or network communications from the LUMXPROC program, just a small amount of FICON channel activity.
- MDI Platform commands (and responses) are passed over the FICON channel to an allocated virtual tape drive via tape control unit commands. One (or more) of the DISP=OLD and/or DISP=NEW files may remain open (in input mode) for the duration of MDI processing.
- Use the normal tape methods to expire and scratch the MDI Platform virtual tape VOLSERs
- Do not use DISP=MOD for an MDI tape file. It is not appropriate for a DISP=OLD file. It is also not appropriate for a DISP=NEW file since the file is completely overlaid with data from MDI Platform processing.

The passing of a mounted DISP=NEW VOLSER from the LUMXPROC step to any subsequent step in the job is not supported. This is because the LUMXPROC unloads the VOLSER during MDI processing and does not remount it.

- All MDI tape files specified by DD statements in the LUMXPROC step remain enqueued for the duration of that step. It's important to understand that these data sets cannot be referenced by or created by any other MDI processing until the step completes. The operating system enqueue on data set name effectively enforces this restriction.

- From the perspective of z/OS batch processing, it appears as though LUMXPROC is processing the data files using zero CPU and zero data I/O. The LUMXPROC program does not read or write tape data blocks.

Appendix E. Restore .XMT File Using TSO Receive

The mainframe software components are shipped in TSO XMIT format via email. To unpack a mainframe file that was sent in TSO XMIT format, do the following.

1. Create an FB LRECL=80 sequential (PS) dataset. We use BLKSIZE=3120 but it does not matter. We typically use 'userid.UPLOAD.XMT' as the upload data set name.
2. Use your PC 3270 emulator to upload one of the XMIT files into this data set.
Be sure to specify NO CRLF conversion, TSO, and NO ASCII/EBCDIC conversion.
You could use FTP in BINARY mode, but we do not provide the corresponding instructions on how to do this.
3. From TSO/ISPF, run the following command using your data set name.
RECEIVE INDSN('userid.UPLOAD.XMT')

You will be prompted for restore parameters. Just press enter and the transmitted data set will be created with your userid as the high-level qualifier. (Example: userid.PDQ.LOADLIB)

If you already have a data set with this name, TSO RECEIVE will ask what you want to do. You can then specify: DSN('the.name.you.choose')

Rename or copy the data set to an appropriate location and name for your site. The receive will have restored the file to its original (or compatible) block size and data set organization.

Appendix F. PDQ - DCB Exit Sample: DCXTYPE

This sample DCB exit is included in the PDQ software package. It makes decisions based on the lowest delimited qualifier of the distributed-systems file name. In addition, some characteristics can be explicitly set by appending “@” onto the file name and then specifying the appropriate keyword attributes.

```
/* REXX *****
PURPOSE: SET DCB CHARACTERISTICS BASED UPON FILEPATH/FILENAME
METHOD: CONVERT FILENAME INTO USER DATA SET NAME
METHOD: SET CHARACTERISTICS BASED ON FILENAME SUFFIX
METHOD: OVERRIDE CHARACTERISTICS BASED ON @ATTRIBUTES
2018.02.12 INITIAL SAMPLE FROM LUMINEX SOFTWARE INC
*****
HERE ARE PARAMETER=VALUE EXAMPLES OF THE INPUTS TO THIS DCB EXIT.
    FULLNAME='/root/dir1/dir2/SYSPLEX1/RSOSM1/Sales.SteveMilton.csv'
    USERID='RSOSM1'
    WATCHER='WATCHER1'
    SERVER='MDIPLAT2'
    BYTES=82880                (NUMBER OF BYTES IN THE INCOMING FILE)
*****
REQUIRED OUTPUT VARIABLES (MUST CONTAIN VALID VALUES):
    DSN          DATA SET NAME THAT IS TO BE CATALOGED
    RECFM         F FB FBS FBA FBM V VB VBS VBA VBM U
    LRECL        1 TO 32760
    BLKSIZE       0 MEANS LET SYSTEM SET IT, OTHERWISE GIVE VALID BLKSZ
OPTIONAL OUTPUT VARIABLES:
    CONVERSION    NATIVE|EBCDIC|<custom>    DEFAULT: NATIVE
    EMAIL         COMMA DELIMITED LIST OF EMAIL ADDRS; CAN BE EMPTY
*****/
PARSE UPPER ARG ARG1 .
PARSE VALUE WITH DSN RECFM LRECL BLKSIZE CONVERSION EMAIL /*CLEAR*/
DTTM=DATE('S') TIME()

DEBUG=0                      /* 1=TRUE 0=FALSE */
CASESENSITIVE=0

IF ARG1='PDQ' THEN DO        /* NORMAL OPERATION */
    SIGNAL DCBEXIT
    SAY 'SHOULD NOT COME BACK FROM SIGNAL'
    EXIT 32; END

/*****/
/* TEST VARIOUS COMBINATIONS TO VERIFY THIS EXEC WORKS CORRECTLY */
/*****/
DEBUG=1                      /* 1=TRUE 0=FALSE */
WATCHER='WATCHTEST'; SERVER='SIMULATED'
DIRS='/root/dir1/dir2/QUEUE1/'

SAY;SAY;SAY;SAY;SAY
SAY COPIES('-',72)
USERID='RSOSM1'
BYTES=82880
CALL TEST 'mcp.warehouse.inventory.movement.txt'
CALL TEST 'Sales.SteveMilton.csv'
CALL TEST 'Sales.TimDiamond.csv'
CALL TEST 'damage.photo.jpeg'

SAY COPIES('-',72)
```

```

EXIT 99

TEST:
  PARSE ARG TESTFN ','
  PARSE VALUE WITH DSN RECFM LRECL BLKSIZE CONVERSION EMAIL /*CLEAR*/
  FULLNAME=DIRS||USERID'/'TESTFN

  CALL DCBEXIT
  SAY '      DSN='DSN' F='RECFM 'L='LRECL 'B='BLKSIZE ,
    'C='CONVERSION 'E='EMAIL
  RETURN

/******
/* SET DSN AND DCB CHARACTERISTICS */
/******
/* FULLNAME='/root/dir1/dir2/SYSPLEX1/RSOSM1/SalesRpt.csv@vb512'
  USERID='RSOSM1'
  WATCHER='WATCHER1'
  SERVER='MDIPLAT2'
  BYTES=82880              (NUMBER OF BYTES IN THE INCOMING FILE)
*/
DCBEXIT:
  IF DEBUG>0 THEN DO
    SAY 'FULLNAME='FULLNAME' USERID='USERID' WATCHER='WATCHER ,
      ' SERVER='SERVER
  END

  /* FIND PATH DELIMITER CHAR FOR THIS FILE SOURCE */
  SLF=POS('/',FULLNAME); IF SLF=0 THEN SLF=99999
  SLB=POS('\',FULLNAME); IF SLB=0 THEN SLB=99999
  PD1=MIN(SLF,SLB) /* USE LEFTMOST SLASH TYPE */
  IF PD1=99999 THEN DO
    SAY 'UNABLE TO FIND PATH DELIMITER IN FULLNAME='FULLNAME
    RETURN 25; END
  PDELIM=SUBSTR(FULLNAME,PD1,1)
  PD2=LASTPOS(PDELIM,FULLNAME)

  IF \CASESENSITIVE THEN ,
    FULLNAME=TRANSLATE(FULLNAME) /* FORCE UPPERCASE FOR SIMPLE MATCHING */
    PATHNAME=SUBSTR(FULLNAME,1,PD2-1) /* DOES NOT INCLUDE TRAIL SLASH */
    BASENAME=SUBSTR(FULLNAME,PD2+1) /* MAY CONTAIN @CONVERT SUFFIX */

  FILENAME=BASENAME
  FILEATTR=''
  /* ALLOW @ IN BASENAME TO TRIGGER SPECIAL CONSIDERATIONS */
  IF POS('@',BASENAME)>0 THEN DO
    PARSE VAR BASENAME FILENAME '@' FILEATTR
  END
  /* PATHNAME='/root/dir1/dir2/SYSPLEX1/RSOSM1/SalesRpt.csv@vb512@ebcdic'
  BASENAME='SalesRpt.csv@vb512@ebcdic'
  FILENAME='SalesRpt.csv'
  FILEATTR='vb512@ebcdic' /* leading @ is omitted */
  */

  IF DEBUG>0 THEN DO
    SAY 'FILENAME='FILENAME' USERID='USERID' PATH='PATHNAME
    IF FILEATTR<>' THEN SAY ' FILEATTR=@'FILEATTR
  END

```

```

/* ----- */
/* CONVERT FILENAME TO USERID.DSN IF ALL QUALS ARE ALLOWED IN DSN */
DQ=SPACE(TRANSLATE(FILENAME,' ','_')) /* REMOVE QUAL DELIMS */
DQD=''
LOWQ=''
DO WHILE DQ<>' ' /* CHECK QUALS FOR USE IN DSN */
  PARSE VAR DQ DQQ DQ
  IF LENGTH(DQQ)<=8 ,
    & VERIFY(LEFT(DQQ,1),'ABCDEFGHIJKLMNOPQRSTUVWXYZ$#@')=0 ,
    & VERIFY(DQQ,'ABCDEFGHIJKLMNOPQRSTUVWXYZ$#@0123456789')=0 THEN DO
      DQD=DQD'.'DQQ
      LOWQ=DQQ
    END
  ELSE DO
    SAY DTTM 'QUALIFIER EXCEEDS 8 CHARS:' DQQ 'IN' FILENAME
    DQ='' ; DQD='' ; LOWQ='' ; LEAVE
  END
END
IF LEFT(DQD,1)='.' THEN DSN=USERID||DQD /* PUT DSN UNDER USERID QUAL*/
/* FILENAME='SalesRpt.csv'
  USERID='RSOSM1'
  DSN='RSOSM1.SALESRPT.CSV'
  LOWQ='CSV'
*/

IF LENGTH(DSN)>44 THEN ,
  SAY DTTM 'DSN EXCEEDS 44 CHARS:' DSN 'FROM' USERID FILENAME

/* ----- */
/* DEFAULT ATTRIBUTES BASED ON DSN LOW QUALIFER */
/*
PARSE VALUE WITH FARECFM FACONV FABLK SZ /* CLEAR DEFAULT ATTRIBS */

IF DEBUG=2 THEN TRACE I
IF LOWQ='ASC' THEN DO; FARECFM='VB1024'; FACONV='EBCDIC'; END
IF LOWQ='ASCII' THEN DO; FARECFM='VB1024'; FACONV='EBCDIC'; END

IF LOWQ='TXT' THEN DO; FARECFM='VB1024'; FACONV='EBCDIC'; END
IF LOWQ='TEXT' THEN DO; FARECFM='VB1024'; FACONV='EBCDIC'; END

IF LOWQ='CSV' THEN DO; FARECFM='VB4096'; FACONV='EBCDIC'; END

IF LOWQ='BIN' THEN DO; FARECFM='U4096'; FACONV='NATIVE'; END
IF LOWQ='BINARY' THEN DO; FARECFM='U4096'; FACONV='NATIVE'; END
IF LOWQ='JPG' THEN DO; FARECFM='U4096'; FACONV='NATIVE'; END
IF LOWQ='JPEG' THEN DO; FARECFM='U4096'; FACONV='NATIVE'; END

TRACE N

/* ----- */
/* USE @ATTRIB'S TO SET/OVERRIDE FILE CHARACTERISTICS */
/*
IF DEBUG=3 THEN TRACE I
ZFAZ=FILEATTR
DO WHILE ZFAZ<>' '
  PARSE VAR ZFAZ FA '@' ZFAZ
  PFA=FA /* ALPHABETIC PREFIX OF FILEATTRIB TOKEN */
  /* SPLIT FRONT ALPHA FROM BACK NUMBERS */
  ZNI=VERIFY(PFA,'1234567890','M') /* FIND 1ST DIGIT IF PRESENT */
  IF ZNI>1 THEN PARSE VAR PFA PFA =(ZNI) .

```

```

IF FIND('FB FBA FBM F FA FM VB VBA VBM VBS VA VM U',PFA)>0 THEN DO
    FARECFM=FA
    ITERATE; END
IF FIND('EBC EBCDIC A2E',PFA)>0 THEN DO
    FACONV='EBCDIC'
    ITERATE; END
IF FIND('NAT NATIVE',PFA)>0 THEN DO
    FACONV='NATIVE'
    ITERATE; END
IF FIND('BLK BLKSZ BLKSIZE',PFA)>0 THEN DO
    FABLSZ=FA
    ITERATE; END
SAY DTTM 'ATTRIB "'FA'" WAS NOT RECOGNIZED.'
END
TRACE N

/* ----- */
/* MANDATORY ATTRIBUTES BASED ON DSN LOW QUALIFER */

IF LOWQ='TRS'      THEN DO; FARECFM='FB1024'; FACONV='native'; END
IF LOWQ='TERSE'    THEN DO; FARECFM='FB1024'; FACONV='native'; END

IF LOWQ='XMT'      THEN DO; FARECFM='FB80';   FACONV='native'; END
IF LOWQ='XMIT'     THEN DO; FARECFM='FB80';   FACONV='native'; END

/* ----- */
/* PARSE FARECFM ... SPLIT FRONT ALPHA FROM BACK NUMBERS */
IF DEBUG=2 THEN TRACE I
IF FARECFM<>' ' THEN DO
    RECFM=FARECFM; RECFMLEN=''
    /* SPLIT FRONT ALPHA FROM BACK NUMBERS */
    ZNI=VERIFY(RECFM,'1234567890','M') /* FIND 1ST DIGIT IF PRESENT */
    IF ZNI>1 THEN PARSE VAR RECFM RECFM =(ZNI) RECFMLEN
    IF \DATATYPE(RECFMLEN,'W') THEN PARSE VALUE WITH RECFM RECFMLEN

    IF FIND('U',RECFM)>0 THEN DO
        IF RECFMLEN<>' ' THEN DO
            LRECL=RECFMLEN
            BLKSIZE=RECFMLEN
            END
        END
    END

    IF FIND('FB FBA FBM',RECFM)>0 THEN DO
        LRECL=RECFMLEN
        IF LRECL='' THEN DO
            END
        ELSE IF LRECL<1 THEN DO
            END
        ELSE IF (BYTES//LRECL)<>0 THEN DO
            SAY ' FILE BYTES='BYTES 'ARE NOT A MULTIPLE OF LRECL='LRECL
            DO I=1 TO RECFMLEN-1
                LRECL=RECFMLEN+I; IF BYTES//LRECL=0 THEN LEAVE
                LRECL=RECFMLEN-I; IF BYTES//LRECL=0 THEN LEAVE
            END
            SAY DTTM 'ADJUSTED LRECL='LRECL 'BASED ON BYTES='BYTES
            END
        END
    END
END

```

```

IF FIND('F FA FM',RECFM)>0 THEN DO
  LRECL=RECFMLEN; BLKSIZE=RECFMLEN
END

IF FIND('VB VBA VBM',RECFM)>0 THEN DO
  LRECL=RECFMLEN
END

IF FIND('V VA VM',RECFM)>0 THEN DO
  LRECL=RECFMLEN; BLKSIZE=RECFMLEN+4
END

END /* FARECFM */
TRACE N

/* ----- */
/* RESOLVE CONVERSION AND BLKSIZE */
IF FACONV<>' ' THEN DO
  CONVERSION=FACONV
END

IF FABLSZ<>' ' THEN DO
  ZNI=VERIFY(RECFM,'1234567890','M') /* FIND 1ST DIGIT IF PRESENT */
  IF ZNI>1 THEN PARSE VAR FABLSZ . =(ZNI) BLKSIZE
END
IF BLKSIZE='' THEN BLKSIZE=0
IF \DATATYPE(BLKSIZE,'W') THEN BLKSIZE=0
IF BLKSIZE<0 THEN BLKSIZE=0

IF CONVERSION='' THEN CONVERSION='NATIVE'

/* ----- */
/* REPORT RESULTS */

IF DSN<>' ' THEN SIGNAL PASS
/* ELSE, ASSUME THIS DCB EXIT DID NOT HAVE ENOUGH INFO TO DECIDE */

FAIL: /* NO DECISION -- ALLOW SERVER DCB TABLE TO BE USED */
  SAY DTTM '--- N='FULLNAME' DSN='DSN' F='RECFM ,
    'L='LRECL 'B='BLKSIZE 'C='CONVERSION 'E='EMAIL
  /* NONZERO MEANS DONT USE THESE DCB CHARACTERISTICS */
  RETURN 8

PASS:
  IF DSN='' | RECFM='' | LRECL='' | BLKSIZE='' THEN SIGNAL XERR
  SAY DTTM 'DCX N='FULLNAME' DSN='DSN' F='RECFM ,
    'L='LRECL 'B='BLKSIZE 'C='CONVERSION 'E='EMAIL
  RETURN 0

XERR: /* PASS DETECTED A MISSING REQUIRED PARM */
  SAY DTTM '-E- N='FULLNAME' DSN='DSN' F='RECFM ,
    'L='LRECL 'B='BLKSIZE 'C='CONVERSION 'E='EMAIL
  /* NONZERO MEANS DONT USE THESE DCB CHARACTERISTICS */
  RETURN 8

```



871 Marlborough Avenue
Riverside, CA 92507

1.888.LUMINEX
1.951.781.4100

www.luminex.com