Optim Data Privacy Providers (ODPP) v11.7

Release Notes

IBM's Optim Enterprise Solution

Version 11 Release 7 Modification 0 (January 2022)
This edition applies to version 11, release 7, modification 0 of IBM InfoSphere Optim Masking on Demand and to all subsequent releases and modifications until otherwise indicated in new editions.

1. Optim Data Privacy Providers (ODPP)

Optim Data Privacy Providers (ODPP) v11.7 includes the following providers:

Provider Name	ID	Description
Age	AGE	Mask date values
Credit Card	CCN	Mask credit card numbers.
National IDs	NID	Mask National IDs, enforcing country NID standards. National IDs of the following countries can be masked: US UK Canada Italy France Spain
Email	EML	Mask Email addresses.
Affinity (Column Transformation)	COL	Mask value, and maintain picture clause (i.e. number and character data are masked, but remain the same data type.)
Hash	HASH	Generates hash-type value from input value(s).
Lookup	LKP	Given a source value, replace with a corresponding value from a lookup-type table.
Data Swapping	DS	Swaps field values between different rows within a set.

2. Platforms and Compilers

ODPP v11.7 provides binaries built for the following platforms with the indicated compilers in the formats specified:

Platform	Compiler	Bit
Red Hat Linux	gcc 4.8.5	64/32
AIX	IBM XL C/C++ Enterprise Edition for AIX, V11.1	64/32
Suse Linux	Built on RHEL	64/32
Windows	Visual Studio 2013	64/32

3. Documentation

Documentation is available from IBM Documentation. IBM Documentation provides the most current content as it can be quickly updated. Refer to the section titled “Optim data privacy provider library” at the following link on IBM Documentation:

https://www.ibm.com/docs/en/iotdm/11.7

4. ODPP Installation

Complete details on installing ODPP is available from IBM Documentation. Refer to the section titled “Overview for installing the Optim data privacy providers” at the following link on IBM Documentation:

https://www.ibm.com/docs/en/iotdm/11.7?topic=installing-overview-optim-data-privacy-components

5. What’s New

5.1 ODPP 11.7.0

5.1.1 Optional Suppression of ODPP trace files

Prior to this release, ODPP created trace files containing warning messages regardless of ODPPTRCCMD settings. To increase performance and save disk space, set the environment variable ODPPTRCCMD=NO to suppress the creation of trace files. Caution: If Optim support requests a trace file to diagnose a problem, this environment variable must then be UNset.

5.1.2 Support for UDFs in Netezza 11.2

UDF’s are now supported for Netezza 11.2. Note that the UDF’s are NOT supported for Netezza 11.x versions prior to 11.2.

5.1.3 CONSTRANDOM option added to the AGE provider

This behaves similarly to the RANDOM method but the result will be the same each time a given date is masked (similar to “REPEATABLE” in other masking providers).

5.1.4 Use Hardware Based Seed for Random Numbers

When RANDOM masking is done in cases where user does not provide a SEED value, ODPP will now generate a random seed from a hardware-based random number generator. Previously, ODPP used the system clock to generate a seed value.

5.2 Folder Summary

The following provides a summary of the contents of each folder in the ODPP-provided .zip-type file. Later sub-sections provide details for each folder.

Folder	Contents
Images32	**ODPP libraries for 32 bit environments**
aix	ODPP AIX libraries
aix_udf_ora	ODPP AIX libraries for Oracle UDFs
rhel	ODPP Red Hat Linux/SUSE libraries
rhel_udf_db2	ODPP Red Hat Linux/SUSE libraries for DB2 UDFs
rhel_udf_nz	ODPP Red Hat Linux/SUSE libraries for Netezza UDFs
rhel_udf_ora	ODPP Red Hat Linux/SUSE libraries for Oracle UDFs
win	ODPP Windows libraries and include files
win_udf_db2	ODPP Windows libraries for DB2 UDFs
win_udf_mss	ODPP Windows libraries for SQL Server UDFs
win_udf_ora	ODPP Windows libraries for Oracle UDFs
win_udf_tera	ODPP Windows libraries for Teradata UDFs

Images64	**ODPP libraries for 64 bit environments**
aix	ODPP AIX libraries
aix_udf_db2	ODPP AIX libraries for DB2 UDFs
aix_udf_ora	ODPP AIX libraries for Oracle UDFs
rhel	ODPP Red Hat Linux/SUSE libraries
rhel_udf_db2	ODPP Red Hat Linux/SUSE libraries for DB2 UDFs
rhel_udf_ora	ODPP Red Hat Linux/SUSE libraries for Oracle UDFs
win	ODPP Windows libraries and include files
win_udf_db2	ODPP Windows libraries for DB2 UDFs
win_udf_mss	ODPP Windows libraries for SQL Server UDFs
win_udf_ora	ODPP Windows libraries for Oracle UDFs
win_udf_tera	ODPP Windows libraries for Teradata UDFs
Replacement Data	**Replacement data in CSV format and DDL for DB2**
Data	Replacement data in the form of CSV-type files and DB2-LUW DDL. This is intended to be used for with the Lookup service provider.
Doc	**Documents and Developer's Guide**
Developer_Guide	ODPP Developer's Guide provides user guide type instructions for C/C++-type programming using the ODPP APIs and the ODPP SPIs.
JavaAPI	Documentation (Javadoc) for the ODPP Java API classes.
ODPPMessages.txt	List of ODPP return codes and messages.
ODPP_v11.7_Release_Notes.pdf	ODPP v11.7 Release Notes
ODPP-Users-Guide.pdf	ODPP Users Guide provides usage-type information on using and licensing ODPP.

Scripts	Symbolic link creation and uninstallation scripts
createODPPsymboliclinks.sh	This shell script must be used to create symbolic links for the ODPP and ICU libraries on UNIX. Usage: sh createODPPsymboliclinks.sh <path> <major_ver> <full_ver> <hybrid version> <cleanonly> where: <path> is the path to the directory containing the ODPP binaries <major_ver> is the major version (e.g. 11.7) <full_ver> is the full version (e.g. 11.7.0.0) <hybrid version> is a hybrid 2 position ODPP version (e.g. 117) <cleanonly> removes symbolic links
removeODPP.bat	Windows-type batch command file to remove ODPP binaries from your system
removeODPP.sh	UNIX/Linux shell script to remove ODPP binaries from your system
Samples	**Sample Code**
App_CCN	A sample C++-type program that details the use of the ODPP APIs for invoking the ODPP CCN service provider.
CMExit_ODPP_CCN	A sample Optim Column Map Exit C language program that details the use of the ODPP APIs for invoking the ODPP CCN service provider. See Optim_CMExit_ODPP_CCN.doc/.mht for complete details on this sample application.
SrvPrv	A sample C language DLL that demonstrates an ODPP Service Provider Interface (SPI) module. This sample may be used as a guide for creating yourr own SPI module which may be plugged into the ODPP framework.
zOS	This folder and sub-folders contain C language and a COBOL lauguage Optim/z Column Map Exits (CMEs) that demonstrates the use of the ODPP APIs from an Optim/z CME.
ODPP_AFFLANGEXIT	Sample Affinity custom language exit
ODPP_HASHEXIT	Sample EXIT for use with the newly added support for SHA-256 hashing algorithm
Include	Contains ODPP common-type header files for the samples
JavaAPI	Sample Java program demonstrating the use of the ODPP Java API. The folder also includes a Readme file with information about the API.

5.3 Images-type folder

These folders contain the ODPP binary images for each supported platform and header files. See the summary table above for a description of each image-type folder.

5.4 Doc

This folder contains the ODPP documentation.

5.4.1 Release Notes

The ODPP v11.7 Release Notes contains information regarding the ODPP release.

5.4.2 ODPP User's Guide

The ODPP User's Guide is a comprehensive usage-type document for ODPP users. It includes full details on installing ODPP. It also includes all of the ODPP licensing information from the former ODPP Licensing Guide.

5.4.3 ODPP Developer's Guide

The ODPP Developer's Guide provides user guide type instructions for C/C++-type programming using the ODPP APIs. It provides details on the ODPP APIs, structures and parameters. It provides a step-by-step guide to
- Initialize and configure ODPP,
- Create the input/output data structures required by ODPP
- Execute the ODPP Service Provider.

The ODPP Developer's Guide also includes details on parameters and their allowable values for each ODPP Service Provider.

To launch the Developer's guide:
1. In the Developer_Guide folder, double-click launch.bat, or
2. In the html folder beneath that, double-click index.html.

5.5 Replacement Data

This folder contains replacement-type data in form of CSV-type files and database-specific DDL that is intended to be used for lookup purposes.

The CSV-type files can be loaded into a target database using the DDL files provided along with the data. At the moment, only DB2-LUW-type DDL is provided.

5.6 Samples

This folder contains ODPP-type sample code.

5.6.1 Sample stand-alone ODPP application

The App_CCN folder contains sample C++-type code for calling ODPP APIs, specific to the credit card number (CCN) service provider. The code demonstrates how to populate various ODPP structures and call the ODPP APIs from initialization through termination. For ease of use, it also contains sample code to read input data from a text file and write the masked data to a text file.

5.6.2 Sample Optim on LUW Column Map Exit using ODPP

The CMExit_ODPP_CCN folder contains sample C-type code for an Optim on LUW Column Map exit that call into the ODPP APIs, specific to the credit card number (CCN) service provider. The code demonstrates how to populate various ODPP structures and call the ODPP APIs from initialization through termination.

5.6.3 Sample Optim/z Column Map Exits using ODPP

The zOS folder and sub-folders contain sample Optim z/OS Column Map Exits (CMEs) that call into the ODPP APIs. They are:
1. A COBOL language standalone program that uses ODPP z/OS for masking Dates.
2. A C language Optim/z Column Map Exit (CME) that uses ODPP z/OS for masking Credit Card Numbers.
3. A C language Optim/z Column Map Exit (CME) that uses ODPP z/OS for providing a hash look from a replacement table.
4. A COBOL language Optim/z Column Map Exit (CME) that uses ODPP z/OS for masking the National Identifier (NID) for the U.S. which is the Social Security Account Number (SSAN).

5.6.4 Sample ODPP Service Provider

The SrvPrv folder contains a sample ODPP Service Provider implemented using the ODPP Service Provider Interface (SPI).

5.6.5 Sample ODPP_AFFLANGEXIT

Sample Affinity custom language exit.

5.6.6 Sample ODPP_HASHEXIT

Sample EXIT for use with the newly added support for SHA-256 hashing algorithm.

5.6.7 Sample Java program

The JavaAPI folder contains a sample Java program that demonstrates the Java API. The folder also includes a Readme file with information about the API.

6. ODPP integration considerations

Applications integrating with ODPP should either link to the ODPP Provider Core library **(…ODPProvider…)** or load it dynamically. All other ODPP-type libraries other than the core library are managed by the ODPP core library.

7. Encoding and code page support

ODPP currently supports input data in single-byte character sets (SBCS), Unicode (UTF16/UTF32), and multi-byte character sets (MBCS). The code pages supported are the same as those supported by Optim on LUW, which are documented here:

https://www.ibm.com/docs/en/iotdm/11.7?topic=configuration-character-formats

8. Limitations

The following are limitations within the current ODPP implementation:

8.1 User-Defined Functions (UDFs)

For ODPP v11.7, UDFs are supported for the following databases:
- DB2 z/OS
- DB2 LUW
- MS SQL Server
- Netezza
- Oracle
- Teradata

For ODPP v11.7., UDFs are supported for the following ODPP-type service providers:
- Affinity (column transformation)
- Age
- Credit Card Numbers
- Email
- Hash
- National Identifiers

For ODPP v11.7, UDFs are supported for the following data types:
- char
- date
- decimal/numeric
- double/float
- integer
- time
- timestamp
- varchar

8.1.1 Netezza

8.1.1.1 Random masking using CCN, EML or NID providers may have duplicates.

During execution of a query, Netezza sends the query to all the data slices present, these data slices execute the query in separate parallel processes. At the end of the query, the output from all of the queries on all of the data slices are merged into a single result set.

From the ODPP perspective, the Random method generates the output using a combination of a random sequence and sequence numbers. This means it always starts from the same point for each query in each Netezza slice thus generating the same set of values in each slice. When the query output is merged together, the result is duplicate values with a rate of duplication equal to the number of data slices.

8.1.2 Oracle

When a UDF is run on an Oracle server, it is actually run within a separate process called EXTPROC.EXE on the Oracle server. At the beginning of the execution of an Oracle UDF (i.e. external process) the external program library (i.e. DLL or shared object) is loaded into memory within the EXTPROC.EXE process. When the Oracle SQL query hosting a UDF execution completes, the external program library (i.e. DLL or shared object) is not unloaded from memory. This means subsequent executions of the same UDF do not require the library to be reloaded.

An unfortunate by-product of this action is that if you are using an ODPP-type UDF that uses the Email-type service provider and you have specified a METHOD=REPEATABLE, the subsequent executions of the UDF with the same inputs will not reproduce the same repeatable outputs. This is because the sequence indicator that is maintained within the ODPP is not reset between executions since the ODPP libraries remain in memory.

You may overcome this Oracle-type limitation by simply stopping and restarting the Oracle Listener service as this will cause the EXTPROC.EXE to unload the libraries when the service is stopped and to reload the libraries when the ODPP-type UDF is subsequently run.

Another alternative might be to use the Email-type service provider with the METHOD=HASH. This may result in some duplicates but there would be consistency between different executions.

8.1.3 Teradata

By default, Teradata UDFs currently do not provide persistent storage across invocations. This means that the ODPP-type UDFs for Teradata are initialized and terminated upon invocation for each row. A side affect of this is that the ODPP-type service providers, Credit Card Number (CCN) and National Identifiers (NID) will not generate masked output data that is truly random as the METHOD=RANDOM algorithm is based upon a counter which is consistent with Optim.

Given the Teradata UDF lack of persistent storage, this counter will be reset to 1 each time thus producing the masked output that is not as random as you would expect. For the ODPP-type UDFs that utilize CCN or NID, you may want to specify the METHOD=REPEATABLE parameter.

8.2 Hash_Lookup Service Provider

The HASH_LOOKUP service provider supports three special values. They are: -1 for NULL, -2 for Space and -3 for Zero Length as a part of the sequence values in the replacement table. In addition to these special values, positive values from 1 to 'n' where 'n' is the maximum value are supported. Further, there must be no gaps in the positive sequence values.

If you happen to use negative values other than those detailed above or you have gaps in the positive sequence values, then the results of the HASH_LOOKUP function are unpredictable.

8.3 Lookup Service Provider

The ODPP Lookup service on Linux, UNIX and Windows supports DB2-LUW v9.1 and beyond, as well as Oracle v10.2 & v11.2. The ODPP Lookup service on z/OS supports DB2 z/OS v8.1 and beyond.

8.4 Lookup limitation with double and float data type

For Real and Double data type columns, in a DB2 replacement table, ODPP supports only fields with Float and Double ODPP data types respectively. Using any other data type for such columns might result in the following:

1. If used in a search field:
   Failure to search the replacement row

2. If used in a replacement field:
   Values in the output typically in the exponential form. (e.g. 1.3000000E01)

8.5 Others

The ODPP_METHOD_DEFAULT option is the only format that is currently supported for the “sMethod” argument in the Provider_Service() API.

9. Known Issues

The following are known issues with the current ODPP implementation. The Work Item ID# is included for reference:

9.1 NID Service provider for French NID

When using the National Identifier (NID) service provider for the French NID, when the Department masking rule (i.e.. ODPP_FR_PARTS_MASK_DEPT) is used; ODPP produces an invalid French NID. This happens only when the Department masking rule is used.
If the Department masking rule is used along with the Commune masking rule (i.e. ODPP_FR_PARTS_MASK_COMMUNE) then the results are correct. When all other masking rules are used, the results are correct. **Work Item ID#: 36233.**

9.2 Affinity (COL) Service provider and Double data type

When using the Affinity (COL) service provider with the Double data type (e.g. ODPPDATATYPE_DOUBLE) the results are different between Linux, UNIX, and Windows. The differences with Double are expected as the implementation of this data type is somewhat different between UNIX, Linux and Windows.

The problem occurs with double-types when they are converted to a string for processing in ODPP. As an example, using the number 13, when it is converted in Windows it becomes 13.0000 and on AIX it becomes 1.3000000E01. **Work Item ID#: 39591.**

9.3 Lookup Service provider and Double data type

For the Lookup (LKP) service provider, when a Double (e.g. ODPPDATATYPE_DOUBLE) data type is used and the source column value is more than 1-digit in length, the provider fails. **Work Item ID#: 40354.**

9.4 HASH Service provider and Double data type

When using the Hash (HASH) service provider with the Double (e.g. ODPPDATATYPE_DOUBLE) data type the results are different between Linux, UNIX, and Windows. The differences with Double are expected as the implementation of this data type is somewhat different between Linux, UNIX, and Windows.

The problem occurs with double-types when they are converted to a string for processing in ODPP. As an example, using the number 13, when it is converted in Windows it becomes 13.0000 and on AIX it becomes 1.3000000E01. Work Item ID#: 40445.

9.5 Netezza UDFs running in fenced mode

The ODPP-provided Netezza UDFs, by default, are defined to run in unfenced mode. If you need to run the ODPP-provided Netezza UDFs in fenced mode you must also ensure you are using the following Netezza versions with the indicated patches applied:
6.0.8 P2
7.0.0 P2
7.0.2

Failure to do so may result in Netezza system crashes. **Work Item ID#: 45084.**

9.6 Oracle UDF Function OptimMaskDate returns incorrect destination value for Age provider

In some cases the OptimMaskDate UDF function returns the incorrect value in the AGE provider for a data type of DATE. This issue will be corrected in an iFix for v9.1.0.4. In the interim the OptimMaskTimestamp function is a valid work around. **Work Item ID#: 52740.**