Data Masking (Deep Dive)

Agenda

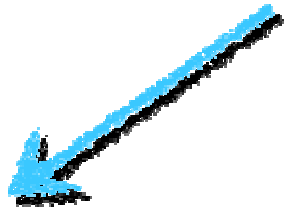
- **Background and Use Cases**
- **Optim Privacy Strategy**
- **Architecture**
- **Optim Data Privacy Providers**
- **Optim**
- **User Defined Functions**
- **Scripting**
- **Big Data Map/Reduce**
- **Guardium Integration Points**
- **Q&A**

Background and Use cases

- **Privacy is (or should be) a concern Gramm-Leach Bliley Act.**
  - Health Insurance Portability and Accountability Act.
  - EU Data Protection Directive.
  - Privacy laws in Canada, Japan, and Australia.
  - Payment Card Industry Data Security Standards.
  - Interagency Guidelines for Safeguarding Customer Information.
  - Basel II operational controls, Sarbanes-Oxley internal controls.

Background and Use cases

Sensitive data exported from the database to external reports.



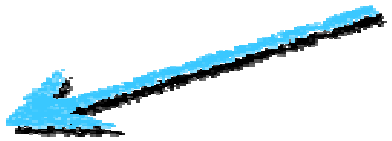
	A	B	C	D	E	F
1	NC003	Olive Oyl	35	F	NorthCentral	MA0081
2	NC005	Mick E. Mouse	50	M	NorthCentral	MA0081
3	NC012	Tooth Fairy	21	F	NorthCentral	MA0081
4	NE003	Peter Pan	50	M	NorthEast	MA0066
5	NE005	Mister Ed	80	M	NorthEast	MA0066
6	NE012	Tinker Belle	18	F	NorthEast	MA0066
7	NW003	Howdy Doody	19	M	NorthWest	MA0015
8	NW005	Darth Vader	45	M	NorthWest	MA0015
9	NW012	P. Wee Herman	30	M	NorthWest	MA0015
10	RP0013	Richard Parente	30	M	Massachusetts	MA0066
11	SC003	Dirt E. Harry	62	M	SouthCentral	MA0021
12	SC005	Leisure Suit Larry	28	M	SouthCentral	MA0021
13	SC012	J. R. Ewing	57	M	SouthCentral	MA0021
14	SE003	Robin Hood	42	M	SouthEast	MA0001
15	SE005	Merrill N. Monroe	29	F	SouthEast	MA0001
16	SE012	Sherlock Holmes	67	M	SouthEast	MA0001
17	SW003	Frank N. Stein	25	M	SouthWest	MA0040
18	SW005	Godzilla Jones	41	F	SouthWest	MA0040
19	SW012	James T. Kirk	37	M	SouthWest	MA0040
20	WE003	Betty Boop	24	F	West	MA0301
21	WE005	Captain Kangaroo	79	M	West	MA0301
22	WE012	Nancy Drew	19	F	West	MA0301

```
SQL:  New  Run  Modify  Use-editor  Output  Choose  Save  Info  Drop  Exit
Run the current SQL statements.

----------- codis@ol_informix1170_1 ------------ Press CTRL-W for Help ------

salesman_id salesman_name                age sex territory      manager_id

NC003       Olive Oyl                     35 F   NorthCentral   MA0081
NC005       Mick E. Mouse                 50 M   NorthCentral   MA0081
NC012       Tooth Fairy                   21 F   NorthCentral   MA0081
NE003       Peter Pan                     50 M   NorthEast      MA0066
NE005       Mister Ed                     80 M   NorthEast      MA0066
NE012       Tinker Belle                  18 F   NorthEast      MA0066
NW003       Howdy Doody                   19 M   NorthWest      MA0015
NW005       Darth Vader                   45 M   NorthWest      MA0015
NW012       P. Wee Herman                 30 M   NorthWest      MA0015
RP0013      Richard Parente               30 M   Massachusetts  MA0066
SC003       Dirt E. Harry                 62 M   SouthCentral   MA0021
SC005       Leisure Suit Larry            28 M   SouthCentral   MA0021
SC012       J. R. Ewing                   57 M   SouthCentral   MA0021
SE003       Robin Hood                    42 M   SouthEast      MA0001
SE005       Merrill N. Monroe             29 F   SouthEast      MA0001
SE012       Sherlock Holmes               67 M   SouthEast      MA0001
SW003       Frank N. Stein                25 M   SouthWest      MA0040
SW005       Godzilla Jones                41 F   SouthWest      MA0040
SW012       James T. Kirk                 37 M   SouthWest      MA0040
WE003       Betty Boop                    24 F   West           MA0301
WE005       Captain Kangaroo              79 M   West           MA0301
WE012       Nancy Drew                    19 F   Mest           MA0301
```

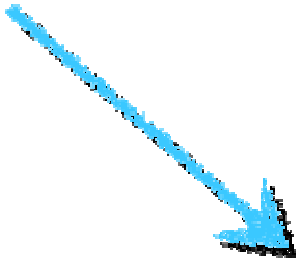
Background and Use cases

SQL queries rendering sensitive information.

```
EXEC SQL DECLARE customers CURSOR FOR
   SELECT cust_id, custname
   INTO :cust_id, :custname
   FROM customers;
EXEC SQL OPEN customers;
int i;
for (i = 1;; i++)
{
    EXEC SQL FETCH customers;
    if (strncmp(SQLSTATE, "00", 2) != 0) {
       break;
    }
   printf("Row %05d: %s '%s'\n", i, cust_id, custname);
}
if (strncmp(SQLSTATE, "02", 2) != 0) {
    printf("SQLSTATE after fetch is %s\n", SQLSTATE);
}
EXEC SQL CLOSE customers;
EXEC SQL FREE customers;
```



```
DISPLAY:  Next  Restart  Exit
Display next page of results.

----------- codis@ol_informix1170_1 -----------

cust_id         00242
custname        Popcorn Videos
address         Aramingo Place
city            Kooskooskie
state           WA
zip             70800
ytd_sales       486.00
salesman_id     NW003
phone_number    2069761210

cust_id         00243
custname        Pick-a-Flick
address         120 Central Avenue
city            Dusty
state           WA
```

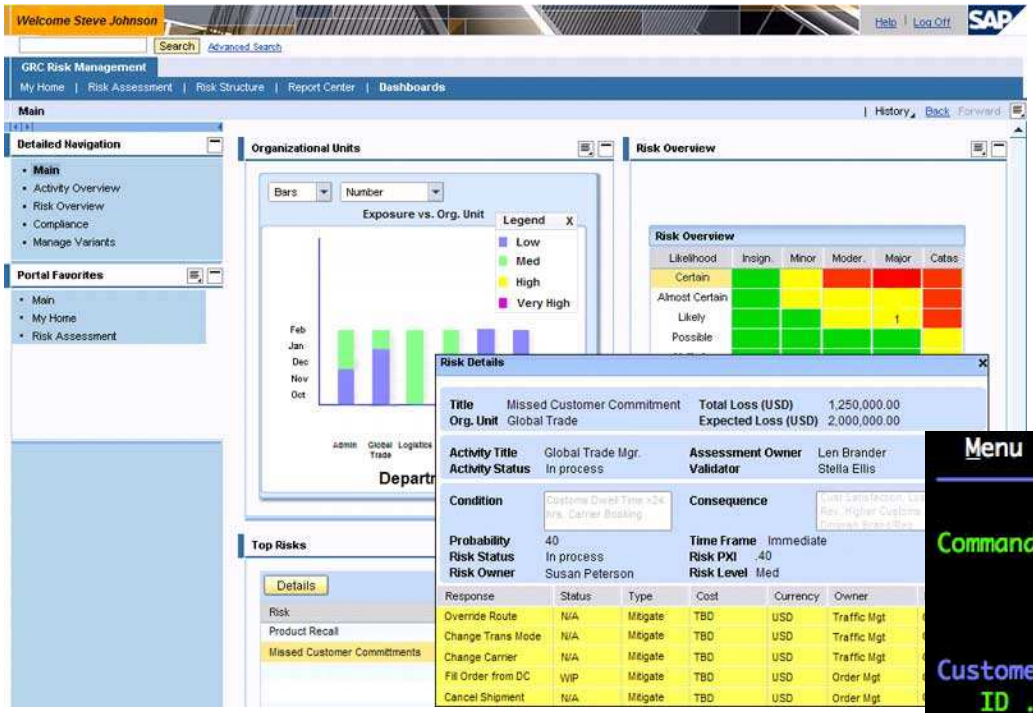


All Output

```
Row 00001: 00242 'Popcorn Videos      '
Row 00002: 00243 'Pick-a-Flick        '
Row 00003: 00244 'Cinemagic           '
Row 00004: 00245 'Movies Galore       '
Row 00005: 00246 'Rick's Flicks       '
Row 00006: 00247 'Movies-R-Us         '
Row 00007: 00248 'Select-A-Movie      '
Row 00008: 00249 'Popcorn             '
Row 00009: 00250 'Popcorn Videos      '
Row 00010: 00251 'Prime Time Video    '
Row 00011: 00252 'Prime Tyme          '
Row 00012: 00253 'Reely Great Videos  '
Row 00013: 00254 'Replay Videos       '
```

Background and Use cases

Applications extracting sensitive data




```
Menu  Customers Orders Details Items Sales  Help
                        Edit Customer
Command ===> _____________________________________________________________
                                                          More:     +

                                                   Row 00001 of 00704
Customer:
   ID . . . . . . . 00242
   Name . . . . . . Popcorn Videos
   Address  . . . . Aramingo Place
   State  . . . . . WA
   Zip  . . . . . . 70800
   YTD Sales  . . . $486.00
   Salesman ID  . . NW003
   Phone  . . . . . 206-976-1210


 F1=Help      F2=Lookup     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Left      F11=Right     F12=Cancel
```

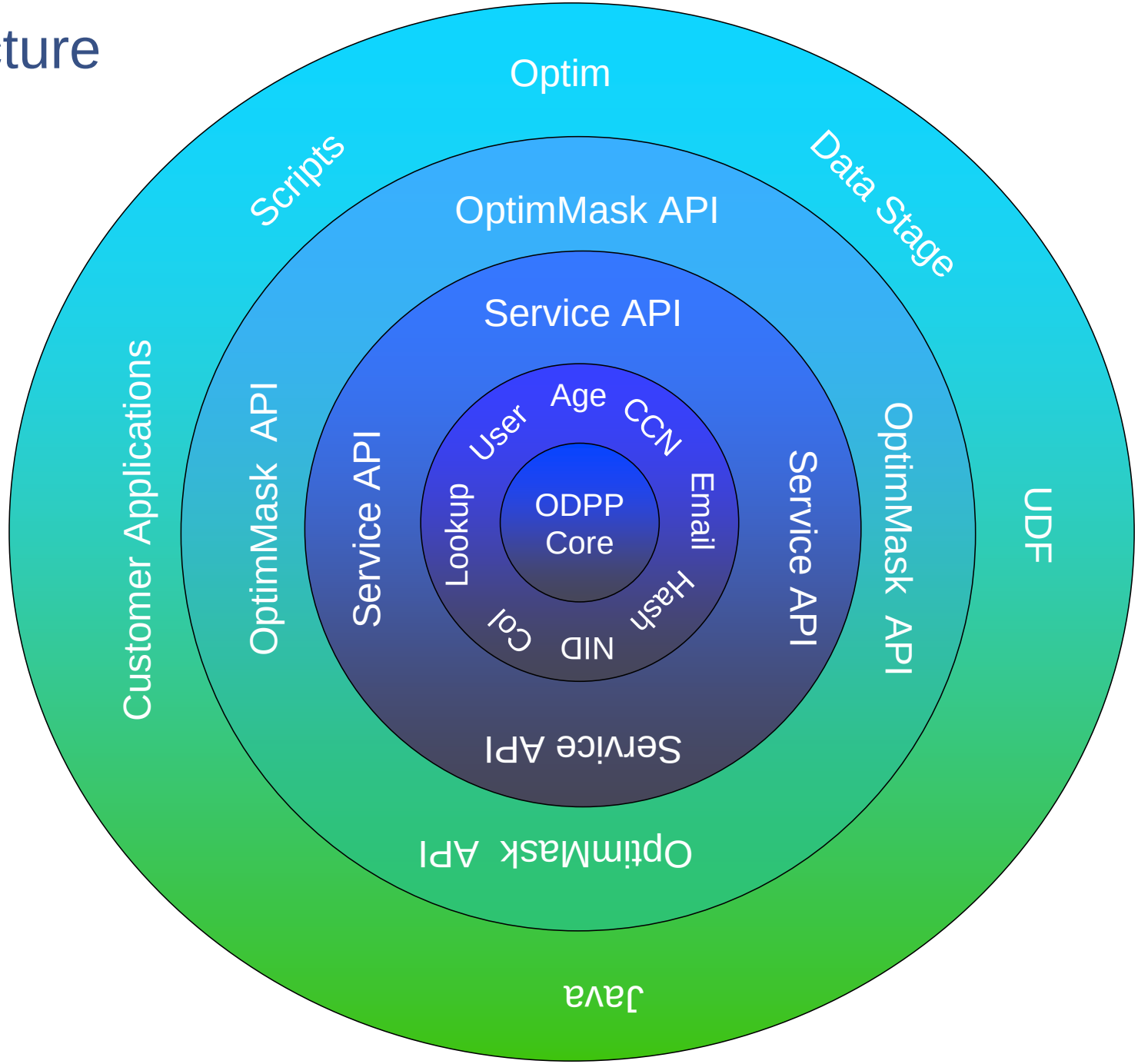
Optim Privacy Strategy

- **Provide consistent masking across all environments - platforms, data sources and use cases**
  - Mask “whatever”, “wherever”, “whenever”
    - At rest or in flight.
    - Relational data, flat files and data sets, IMS, VSAM, etc.
    - During query, load, display, processing, etc.
    - On Linux, UNIX, Windows and z/OS.
  - Consistent behavior
    - Across products, platforms and locales (including customer applications).
  - Repeatable behavior
    - Same inputs and masking parameters yield same outputs.

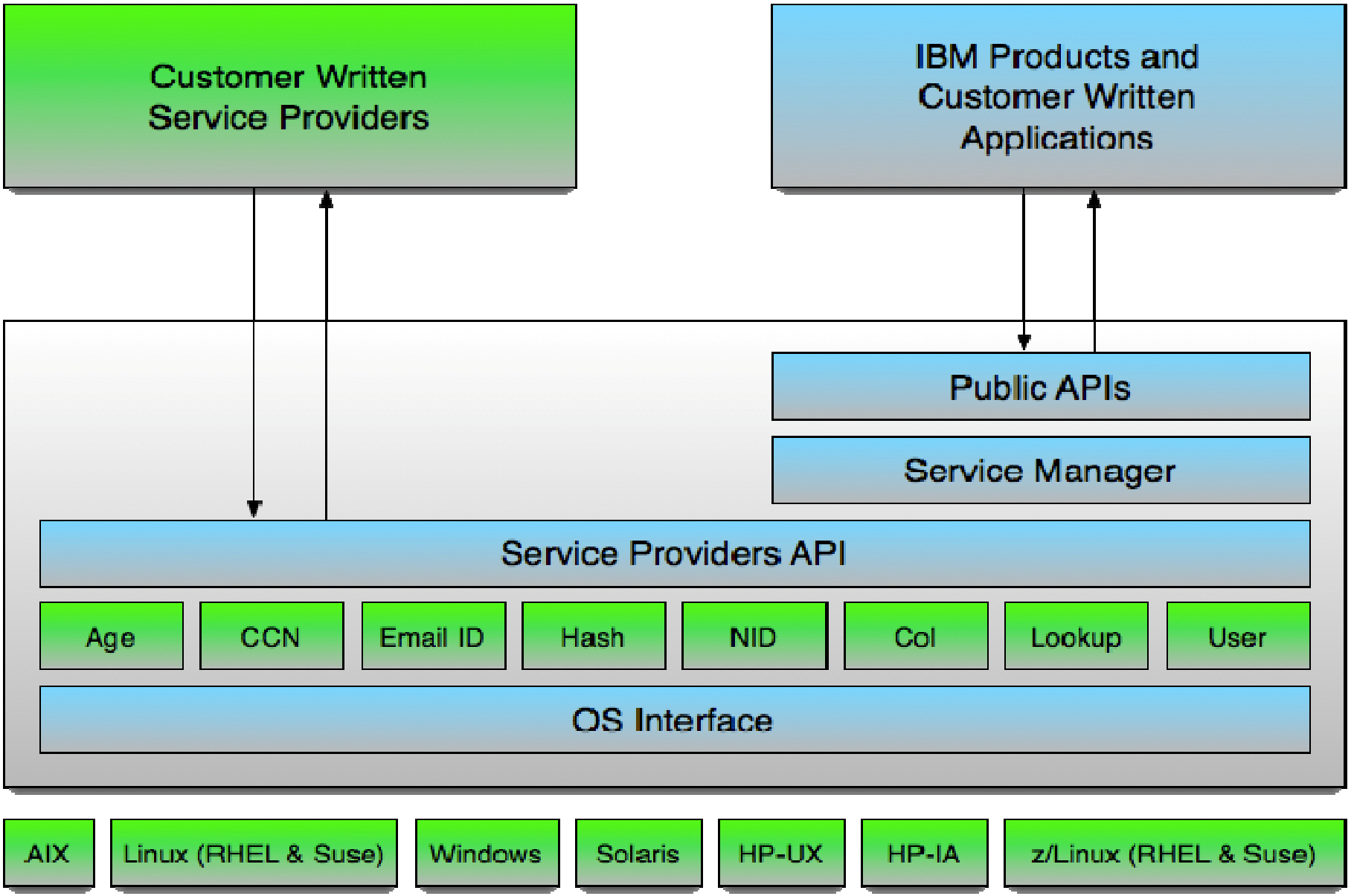
Optim Privacy Strategy

- **Optim Data Privacy Providers**
  - Set of out-of-the-box privacy algorithms referred to as “providers”
  - Can be extended to include user-written providers
  - Simple yet powerful API.
  - Consistent behavior across platforms.
  - Can be used in IBM products and customer applications.
  - Has data source independent design.
  - Provides dynamic invocation of masking services.
  - Written in cross-platform, ANSI C/C++.
    - Interfaces with other languages that support the C calling conventions.
    - Provides platform abstraction services.
  - Is locale and character set aware.
    - Supports SBCS, MBCS and Unicode.
  - Deploys as a number of shared libraries.

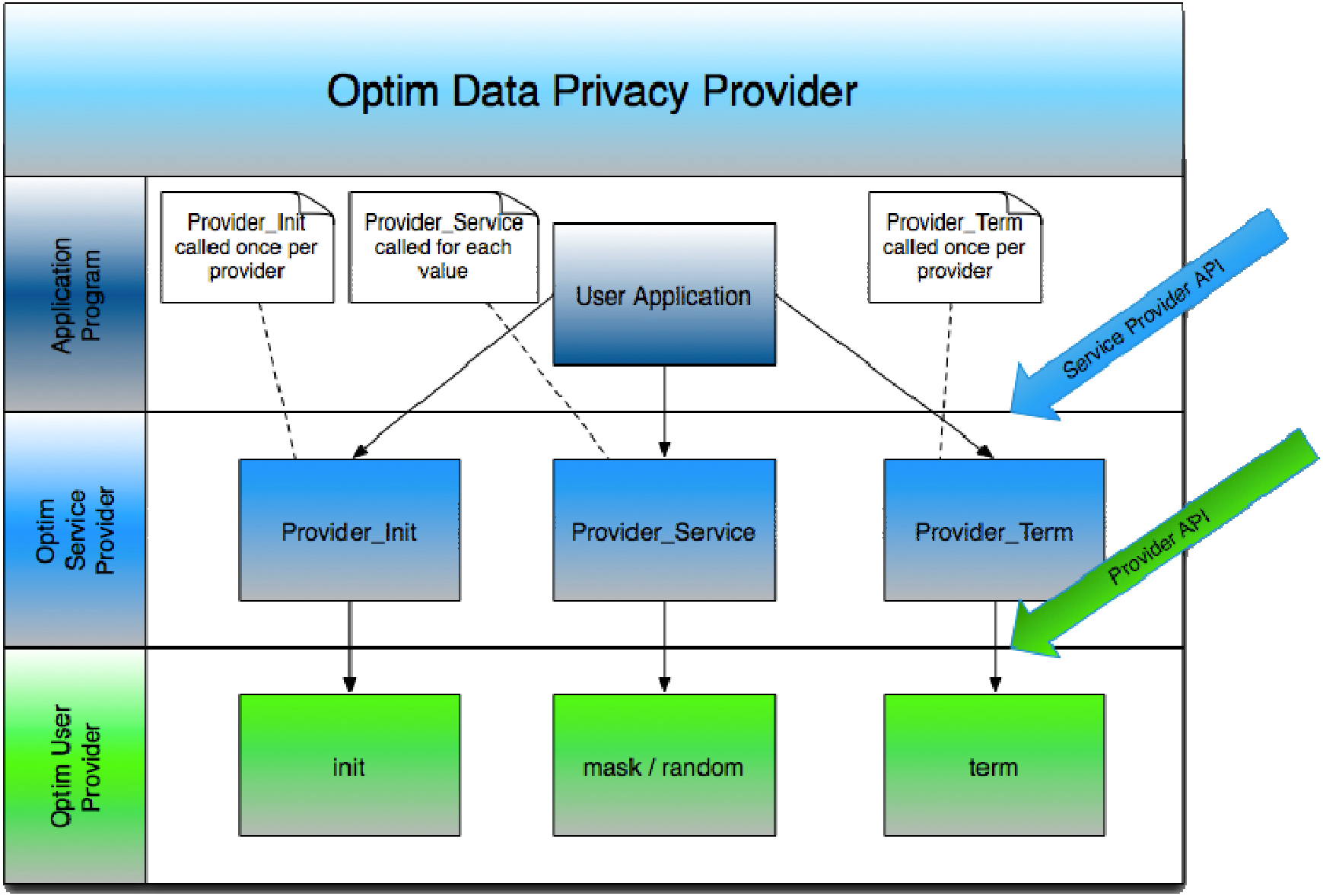
Architecture



Architecture



Architecture



Optim Data Privacy Provider Core Features

•Single API

- Referred to as the Optim Data Privacy Provider API for Data Privacy Services.
  A flexible and extensible API providing an interface that can fit into existing data masking services as well as those that may be developed in the future.
- A command based approach for a broader range of control.
- The API allows applications to connect to the Optim Data Privacy Service Manager which, in turn, will invoke the requested service provider.

•Extensible

- A generic intermediate Data Privacy Service API allows users/application developers to develop their own data masking services and use them in the Optim Data Privacy Provider framework.
- Alternatively, masking services can be directly accessed using the Data Privacy Service API. New data masking services can be added to the existing suite of masking services extending those services to other users.

Optim Data Privacy Provider Core Features

•Dynamic Invocation

- The intermediate Data Privacy Service API provides a very modular design including a plug-n-play approach for service providers.
- New Data Privacy Service Providers can be added dynamically without the need to shut down or recompile running applications.

•Modular Design

- Everything from the Service Manager to the Service Provider, including data conversions, are implemented in separate modules connected to each other by a loosely coupled generic API.
- This provides greater flexibility for plugging in additional external libraries to add new features or enhance existing ones.
- Allows application developers to write their own implementation of the Service Manager (also called Optim Data Privacy Provider Framework) and/or service providers.

•Usability across the Products

- The API has been designed to be used in products that are built in a variety of languages.

Optim Data Privacy Provider Core Features

- **Batch Processing**
  - The API supports batch processing with a user-defined batch size
- **Multi-Platform Support**
  - Currently supports AIX, Linux (Red Hat & Suse), Sun Solaris, HP-Unix, HP Itanium, zLinux (RHEL & Suse), Windows, z/OS
- **Data Source Independent Design**
  - The purpose of the API is to handle data, not the data source.
  - Data source independence provides the flexibility of supporting unlimited data sources as data is extracted and presented to the API by the calling application.
- **Simple and Standard Representation of Data**
  - Structures simulate data as rows and columns of a database
  - Standard data types are used to represent various types of data (e.g. integer, char, null terminated strings, date and time etc.,)

Service Provider: Age

- **Used to age date values in source columns**
- **Aging is a process of incrementing or decrementing a date value**
- **Aging can be specific to a number of years, months, weeks or days**
  - Optionally, may be a combination of these units.
- **Aging can also be based upon a specific 4-digit year value**

Service Provider: Column Transformation

- **Provides data masking of undifferentiated or dynamically-formatted values**
- **Undifferentiated value is where there are no parts that have significance therefore all parts of the value are candidates for masking**
  - **(e. g. 123456, Gizmo, CDE9874)**
- **Dynamically-formatted value has one or more portions that have significance and cannot be altered without affecting the validity of the value**
  - **(e. g. 12-3456789, ItemCode Gizmo, CDE-9874)**
  - Options are provided to specify which portions of the masked value should be unchanged

Service Provider: Credit Card Number

- **Used to generate a valid and unique Credit Card Number (CCN)**
- **By default, it generates a consistently altered CCN based on a source CCN**
  - Uniqueness is guaranteed only for unique input values
- **A CCN is defined by ISO 7812 which consists of:**
  - 6-digit issuer identifier
  - Variable length account number
  - Single check digit as the final number
  - Check digit is verified against the Luhn algorithm
  - Maximum length = 19 digits
- **Two methods for masking:**
  - Mask
  - Random

Service Provider: Credit Card Number

- **Mask Method:**
  - Includes the 1st 4-digits of the source issue identifier
  - Alters the remaining 2-digits of the issue identifier and account number based upon the source CCN
  - A valid check digit is assigned
- **Random Method:**
  - Generates a CCN that may include the 1st 4-digits of the source issue identifier
  - It alternatively, uses an issuer identifier number assigned by:
    American Express, Discover, MasterCard, or Visa
  - A valid check digit is assigned
  - If the 1st 4-digits of source issue identifier are included,
    then, the 1st account number based on those digits will begin with a 1
    and for each additional CCN that uses those digits, the number will be incremented by 1

Service Provider: Email

- **Used to generate an email address consisting of:**
  - User name
  - Separator - usually the at-sign '@'
  - Domain name
- **User name is based upon either:**
  - Destination data
    - or -
  - Literal concatenated with a sequential number
    - or –
  - User-supplied name values using two name-type columns
- **Domain name is based upon:**
  - Email address in the source data
    - or -
  - Literal value
    - or -
  - Randomly selected from a list of email service providers

Service Provider: Hash

- **Used to return a numeric hash-type value based upon an input source value**
- **Multiple source values, of the same or different data type are supported**
  - All source values are converted to a UTF-8 string and then hashed
- **Output hash values may not be unique even when the input is unique but is repeatable based upon a given input**
  - i.e. the same output hash value is generated for the same input hash value
- **Repeatable hash values require a constant seed value for a given input**
- **Hash values for the same input values will vary when the seed is changed**

Service Provider: Lookup

•Masking uses replacement data that is looked-up from a data source based upon a key value

•Required when some types of data (e.g. names, addresses) cannot be generated using arithmetic logic

•Replacement data is typically provided as a set of rows, with a key column, in a database

•Lookup via the key column(s) of a replacement type-table are based upon:

- Values in the key columns of the original input data
  - or -
- Hash-type value generated from original input data columns

Service Provider: Lookup

- **Three types of key lookups are supported:**
  - Basic/plain lookup
  - Hash lookup
  - Random lookup
- **Basic Lookup:**
  - Key source column(s) are used to find matching rows in lookup data
  - There must be a one-to-one mapping between the source key data and the replacement key data
  - Supports single- and multiple-type column lookups
- **Hash Lookup:**
  - Generates a hash value from single- or multiple-type source columns which are used as a key value
    to lookup via a sequence-type column in the replacement table
  - Hash lookup is case sensitive
  - Supports single- and multiple-type column lookups

Service Provider: Lookup

- **Random Lookup:**
  - Selects a value at random from a specified replacement table.
  - Provider generates a random number between 1 and the replacement table row limit supplied by the caller
  - When the replacement table row limit is not supplied, then all rows from the replacement table are read and then uses the total row count as the maximum value for generating a random number
  - The random number becomes the row subscript into the replacement table
  - Supports single- and multiple-type column replacements

Service Provider: National ID

- **Used to generate valid and unique National Identifiers (NIDs) for:**
  - U.S. = Social Security Number (SSN)
  - U.K. = National Insurance Number (NINO)
  - Canada = Social Insurance Number (SIN)
  - France = Institute for Statistics and Economic Studies (INSEE)
  - Italy = Fiscal Code (CF)
  - Spain = Fiscal Identification Number (NIF) / Foreign Identification Number (NIE)

Service Provider: National ID

- **Properties of all the NID routines:**
  - Generates a valid and unique NID for each unique input
  - Two methods:
    - Mask
    - Random
- **Mask:**
  - Algorithm-based generated destination NID based upon a source NID
- **Random:**
  - Randomly generated NID when the source does not have a NID value or when there is no need to transform the source NID in a consistent manner

Optim

- **Adopting Optim Data Privacy Providers as the underlying masking infrastructure**
- **Customer-written privacy providers available via Optim runtime**
- **Customer-written privacy providers discoverable by the Optim user interface**

User Defined Functions

- **Optim Data Privacy service providers invoked via UDFs**
- **Enable sites to mask data before data leaves the database**
- **Mask test environments in place – No need to re-extract**
- **Same masking algorithms used by Optim across all DB(s) and platforms**
- **Increased efficiencies while reducing masking complexity**

User Defined Functions

- **Mask social security numbers before they leave the database**

```
SELECT FIRST_NAME, LAST_NAME,
   OptimMask(SSN, "Provider=NID, method=mask")
   FROM EMPLOYEES WHERE ...
```

- **Mask employee email addresses as they are added to the database**

```
INSERT INTO EMPLOYEES (FIRST_NAME, LAST_NAME, E_MAIL, ...)
   VALUES ("John", "Doe",
         OptimMask(E_MAIL, "Provider=EML, method=random"),
         ...)
```

- **Mask the existing employee account numbers in the database**

```
UPDATE EMPLOYEES SET ACCT_NUM = (SELECT
   OptimMask(ACCT_NUM, "Provider=Variant,seed=@VAR1,method=MASK")
   FROM EMPLOYEES WHERE ...)
```

Scripting

- **Today Customers Write Exits**
  - Assembler, C/C++, COBOL or PL/1 on z/OS
  - C/C++ on Linux, UNIX and Windows
  - Exits are different on Linux, UNIX, Windows vs. z/OS
- **Optim Basic only available on Windows**

Scripting

- **Introducing Lua as Optim's Scripting Language**
  - Powerful, fast, lightweight, embeddable
  - Runs on all platforms
  - Used extensively in industry applications
    - Netezza, Adobe, Ginga, ...
- **Enhanced by IBM**
  - Character set support:
    - SBCS, MBCS, Unicode
  - Broader data type support:
    - Unicode strings, decimals

Scripting

- **Modeled on Optim Basic Column Map invocation pattern**
  - cm_load (optional, invoked when script has been loaded)
  - cm_start_table (optional, invoked at the start of table processing)
  - cm_transform (required, invoked for each row)
  - cm_end_table (optional, invoked at end of table processing)
  - cm_unload (optional, invoked prior to script being unloaded)

- **Implemented using Lua standards and conventions**
  - Source and target presented to script as Lua tables
  - Information about source and target available through table methods.

Scripting

```
-- Load
function cm_load(source, target)
    local dbalias, creatorid
    dbalias = source.getdbalias()
    creatorid = source.getcreatorid()
    report = io.open('/<some path>/report.txt', 'w')
end

-- Start of table
function cm_start_table(source, target)
    local table = source.gettablename()
    report:write('Table processing for ''' .. table .. ''' started at ' .. os.date(), '\n')
end

-- Transform
function cm_transform(source, target)
    local table = source.gettablename()
    -- Handle transform
end

-- End of table
function cm_end_table(source, target)
    local table = source.gettablename()
    report:write('Table processing for ''' .. table .. ''' ended at ' .. os.date(), '\n')
end

-- Unload
function cm_unload()
    report:write('Processed ' .. #tables .. ' tables', '\n')
    report:write('Optim processing ended at ' .. os.date(), '\n')
    io.close(report)
    report = nil
end
```

Scripting

Example:

Swap gender while increasing the age of males and decreasing the age of females.

```
local sex, age
sex = source.column.getvalue('sex')
age = source.column.getvalue('age')
if sex == 'F' then
    sex = 'M'
    if age < 65 then
        age = age + 2
    else
        age = age + 1
    end
else
    sex = 'F'
    if age < 21 then
        age = age - 1
    else
        age = age - 2
    end
end
target.column.setvalue('sex', sex)
target.column.setvalue('age', age)
```

Scripting

Example:

Change zip codes, swap states and alter YTD sales.

```
local zip, state, ytd_sales
 zip = source.column.getvalue('zip')
 state = source.column.getvalue('state')
 ytd_sales = source.column.getvalue('ytd_sales')
 if zip == '66100' then
    zip = nil
 else
    zip = generate_zip()
 end
 if state == 'WA' then
    state = 'NJ'
 end
 local percentage
 if ytd_sales >= 1000 then
    percentage = -10.0
 else
    percentage = 7.50
 end
 ytd_sales = ytd_sales + (ytd_sales * percentage / 100)
 if ytd_sales < 10.0 then
    ytd_sales = 25.0
 end
 target.column.setvalue('zip', zip)
 target.column.setvalue('state', state)
 target.column.setvalue('ytd_sales', ytd_sales)
```

Scripting

Example:

Mask social security numbers and credit card numbers using the ODPP providers NID and CCN.

```
local ssn, ccn
ssn = source.column.getvalue('ssn')
ccn = source.column.getvalue('ccn')
ssn = optimmask(ssn, 'Provider=NID, method=mask')
target.column.setvalue('ssn', ssn)
ccn = optimmask(ccn, 'Provider=CCN, method=random')
target.column.setvalue('ccn', ccn)
```

Big Data

- **Data Masking for IBM BigData**
  - Integrate Optim Data Privacy Providers into Hadoop.
    - Provide Java interface for masking.
      - Incidentally, this works anywhere Java and the privacy providers work.
    - Provide MapReduce base classes and helpers.
      - Configuration.
        - Distributed cache.
        - Shared libraries, license files, etc.
      - Use of masking in Mappers.
      - Use of masking in Reducers.
    - Provide ready-to-run out-of-box configurable Mapper and Reducer.
      - Sub-setting.
      - Masking.

Big Data

- **Example:**

  24 Lines of Java - would be 400+ lines of C

```
FrameworkDefinition frameworkDefinition = new FrameworkDefinition();
frameworkDefinition.getInitializationOperands().add(new InitializationOperand(Constants.ODPP_OPR_ERRORFILE_PATH, "/<path>/"));
frameworkDefinition.getInitializationOperands().add(new InitializationOperand(Constants.ODPP_OPR_LIC_FILES_PATH, "/<path>/"));
odpp.initializeFramework(frameworkDefinition);
ServiceDefinition serviceDefinition = new ServiceDefinition();
serviceDefinition.getInitializationOperands().add(new InitializationOperand(Constants.ODPP_OPR_SWITCH_NA, 0));
serviceDefinition.getInitializationOperands().add(new InitializationOperand(Constants.ODPP_OPR_SOURCE_COLINDEX, 0));
serviceDefinition.getInitializationOperands().add(new InitializationOperand(Constants.ODPP_OPR_METHOD,
        Constants.ODPP_METHOD_MASK));
serviceDefinition.getInitializationOperands().add(new InitializationOperand(Constants.ODPP_OPR_CCN_FLAGS,
        Constants.ODPP_FLAG_CCN_ISSUER6));
serviceDefinition.getFieldDefinitions().add(new FieldDefinition(DataType.WVARCHAR_SZ, 0, 0, 0, "COL_1"));
int serviceToken = odpp.initializeProvider("CCN", serviceDefinition);
Rowset rowset = new Rowset();
Row row = new Row();
rowset.getRows().add(row);
Field ccn = new Field();
row.getFields().add(ccn);
ccn.setSourceSize(80);
ccn.setSource("4571442212344321");
odpp.serviceProvider(serviceToken, serviceDefinition, rowset);
System.out.printf("Masked CCN is '%s'.\n", ccn.getSource());
odpp.terminateProvider(serviceToken);
odpp.terminateFramework();
```

Big Data

Example:

Sum up credit card transactions but mask the credit card numbers.

Map

```
String line = value.toString();
String[] elements = line.split(",");
String ccn = elements[1];
double amount =
Double.parseDouble(elements[2]);
field.setSource(ccn);
try {
    odpp.serviceProvider(serviceToken,
                serviceDefinition, rowset);
} catch (ODPPException e) {
    throw new IOException(e);
}
text.set((String) field.getSource());
output.collect(text, new
DoubleWritable(amount));
```

Reduce

```
double sum = 0.0;
while (values.hasNext()) {
    DoubleWritable value = values.next();
    sum += value.get();
}
output.collect(key, new DoubleWritable(sum));
```

Guardium Integration Points

- **User Defined Functions**
  - Guardium rules can be defined to rewrite SQL
  - Based on several criteria:
    - User, role, host name, IP address, ...

- from

```
SELECT FIRST_NAME, LAST_NAME, SSN
   FROM EMPLOYEES WHERE ...
```

- to

```
SELECT FIRST_NAME, LAST_NAME,
   OptimMask(SSN, "Provider=NID, method=mask")
   FROM EMPLOYEES WHERE ...
```

Q&A

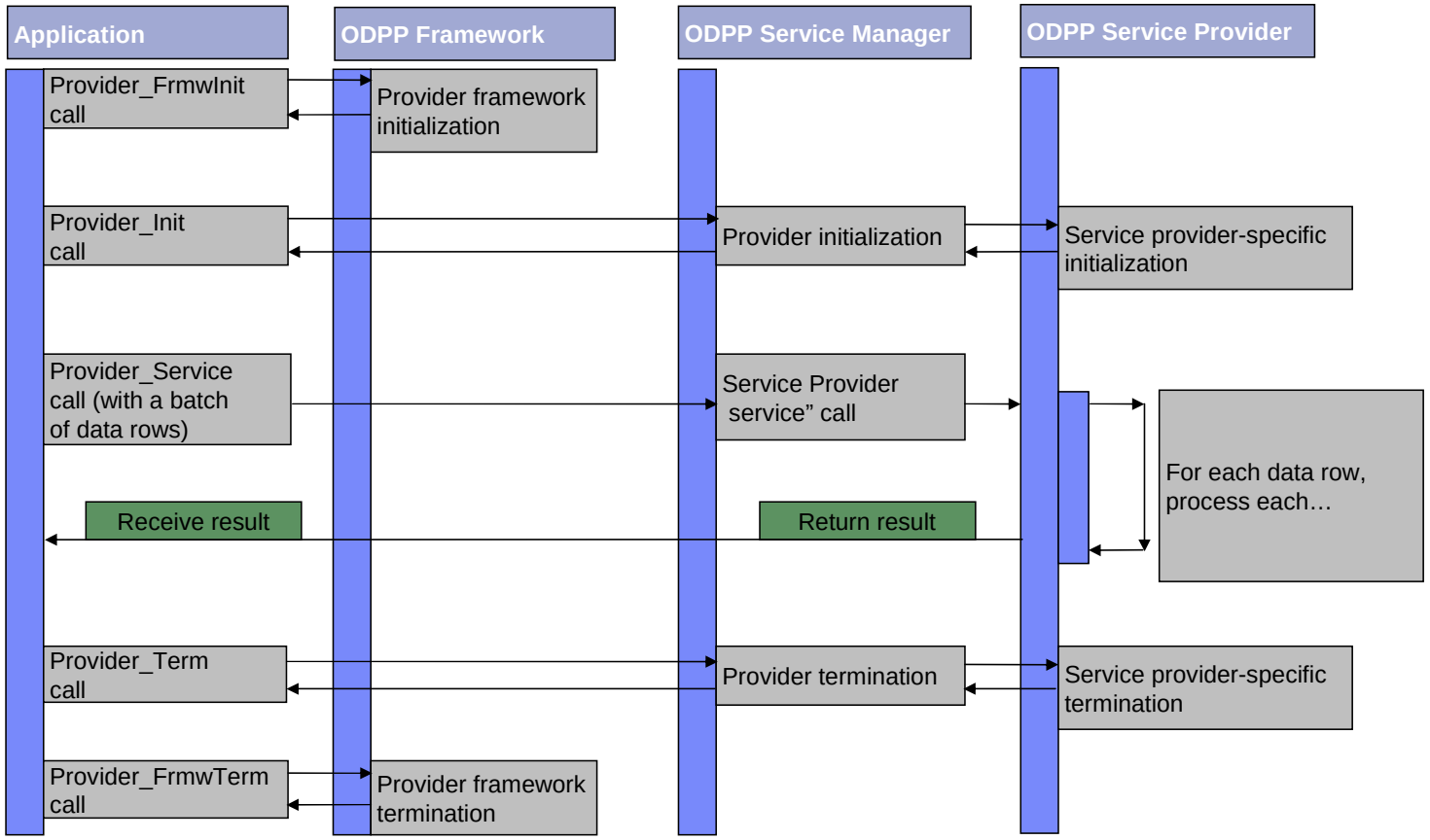
- **Questions?**

Reference Slides

ODPP APIs

- **Provider_FrmwInit**
  - This function initializes the ODPP common framework.
- **Provider_Init**
  - This function initializes an ODPP session and is required for each unique service type requested. An ODPP service token is returned after the session is established. This service token is then required on all subsequent calls for this session.
- **Provider_Service**
  - This function takes the set of data that needs to be masked and executes the Service Provider based upon the supplied service token.
- **Provider_Term**
  - This function closes the Service Provider and frees all the memory allocated during the Provider_Init. When the caller has finished with a Service, this is a required termination call.
- **Provider_GetError**
  - This function returns the oldest available Error Control Block (ECB) and associated tokens in the supplied area. The ECB contains all of the ODPP-related messages and errors. The caller can interpret the ECB or supply it to the Formatted Message Processor to retrieve a message in an available language.
- **Provider_GetFormattedErrorMsg**
  - This function returns the formatted message from the data in the supplied Error Control Block (ECB).
- **Provider_FrmwTerm**
  - This function frees/releases the ODPP framework and all of its components.

Application to/from ODPP Flow



ODPP Licensing

- **ODPP v2.1 is a licensed component**
  - Requires a license to function.
- **Supported ODPP licenses:**
  - Optim Distributed
  - Optim z/OS
  - Optim SOA/Optim Data Masking Solution
  - OEM licensing:
    - In a future ODPP v2.1 fix-pack.
- **ODPP license-type files:**
  - ODPPKEYF.OPT:
    - ODPP license key file
  - ODPPLICF.OPT:
    - ODPP license file (encrypted XML-type file).

ODPP Licensing

- **In Windows:**
  - 1. Check for the ODPPLL environment variable.
  - 2. Search the current process location.
  - Example:
    - Optim Distributed = RTWIN\BIN
- **In UNIX/Linux:**
  - There is no default location, the user must specify the ODPPLL environment variable
- **Via the ODPP provider-type framework initialization**
  - Using a pointer to a string-type path in the DP_FRMW_PARAMS_DEF structure

Character Set Support

- **Includes all data and control structures**
- **Base control structures utilize union-type sub-structures:**
  - Wide-character (Unicode) sub-structure
    - or –
  - Mixed-character (SBCS/MBCS) sub-structure
- **Based upon ICU 4.8**
- **Includes DBMS-specific ICU-type converters for all supported databases**

IBM z/OS Support

- **Single ODPP code base with some limited z/OS-dependent areas**
- **Runs as a set of USS-type libraries:**
  - Parallels other IBM type libraries (e.g. ICU, XML4C, etc.,)
  - Accessible from USS or native z/OS
- **Lookup will interface to DB2 z/OS**
- **Initial delivery via Optim Distributed v8.1 fix-pack**