Dynamic Data Masking
Implementation Best Practices
Abstract
This document defines the steps, resources, and estimated duration required for a successful DDM implementation project.

Supported Versions
- Dynamic Data Masking 9.6

Table of Contents
Abstract ............................................................................................................................................ 2
Supported Versions ......................................................................................................................... 2
Table of Contents ............................................................................................................................ 2
Overview .......................................................................................................................................... 3
Definition Phase ............................................................................................................................... 4
    Resources and Expected Duration .............................................................................................. 4
Discovery Phase .............................................................................................................................. 5
    Resources and Expected Duration .............................................................................................. 5
Application Mapping ......................................................................................................................... 5
    Resources and Expected Duration .............................................................................................. 5
Install and configure DDM ............................................................................................................... 6
    Resources and Expected Duration .............................................................................................. 6
Building DDM Security Rules .......................................................................................................... 6
Building Masking Rules .................................................................................................................. 7
    Building DDM Security Rules for Reporting Environments and Datawarehouses .................... 7
    Building DDM Security Rules for Business Applications ........................................................... 7
PeopleSoft Application Masking Example for Bank Account Number .......................................... 9
    The rule structure to support the PeopleSoft screen masking includes the following rules (top to bottom): .............................................................................................................................. 15
    Resources and Expected Duration ............................................................................................ 21
Capturing User Context Identification Information ......................................................................... 21
    Resources and Expected Duration ............................................................................................ 23
Production Deployment Best Practice ............................................................................................. 23
    Deployment Options: .................................................................................................................. 23
Central Hub Configuration Implications .......................................................................................... 24
Database Server Installation Implications ....................................................................................... 25
Hardware and Operating System Requirements for DDM Server ................................................. 25
    Initial DDM Server CPU Consumption Estimation ................................................................... 25
    Detailed CPU Consumption ...................................................................................................... 25
    Resources and Expected Duration ............................................................................................ 26
On-going Rule Management and Accountability ............................................................................. 26
Overview

In this paper, we address implementing DDM for security purposes only, without implementing performance or production control rules. It is also focused on securing business applications – which is the most challenging implementation. Other DDM use cases, such as securing DBA and development tools have only basic user interaction and thus will be achieved faster than the application use case covered in this document.

We estimate the effort to protect DBA and development tools at 50% of the effort required for similar PII masking scope within business applications that is presented in this document.

Note that an important success factor for DDM implementation is the availability as early as possible of team members that are familiar with the business application’s architecture, database configuration, user-authorization and authentication process.

DDM Implementation phases:

<table>
<thead>
<tr>
<th>Step</th>
<th>Phase</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Definition</td>
<td>Define a list of sensitive and personal information items (PII) that need to be protected to meet your security requirements.</td>
</tr>
<tr>
<td>2</td>
<td>Discovery</td>
<td>Identify the locations, environments, application names, application screens and reports, databases, tables and column names that contain PII. The result is a concise and prioritized list of items that can be organized according to their privacy and sensitivity level. The DDM implementation will follow this list of items.</td>
</tr>
<tr>
<td>3</td>
<td>Application mapping</td>
<td>Map all application screens that provide access to PII. This list will be used:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• To build DDM security rules,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• To verify that updates of masked PII to the databases is prevented</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• As a basis for the user acceptance testing</td>
</tr>
<tr>
<td>4</td>
<td>Install DDM and configure application database connectivity</td>
<td>Install DDM on an application testing image. Determine the best installation mode for each application in the implementation plan (installing DDM centrally or on each database server).</td>
</tr>
<tr>
<td>5</td>
<td>Build DDM rules to protect PII</td>
<td>Analyze application behavior, define security rules and ensure that PII is protected while verifying that screen drill-down and any other application navigation options are operating with masked data and ensuring that update and insert of data (to the database) does not include masked PII.</td>
</tr>
<tr>
<td>6</td>
<td>Identify and create matchers to mask unauthorized end-user access to PII</td>
<td>Identify how user authentication is done in the application and build DDM matchers accordingly.</td>
</tr>
<tr>
<td>7</td>
<td>Testing</td>
<td>Test the different application screens and reports ensuring that PII is secured. PII data reads must be masked while preventing masked data to be inserted or updated.</td>
</tr>
<tr>
<td>8</td>
<td>Go live</td>
<td>Install and configure DDM on production environments while importing predefined rule sets. Provide detailed auditing and summary reporting to key decision makers.</td>
</tr>
</tbody>
</table>
Definition Phase

The purpose of this phase is to create a definitive list of sensitive information items that need to be identified and protected.

To start the definition phase, a closed as possible list of PII items must be specified. Each item must include the appearance pattern, sensitivity and importance level for correctly evaluating the location and current risk level.

For example, a Social Security number is a nine-digit number in the format "nnn-nn-nnnn" and is a highly sensitive Personal Identifiable Information item.

For each PII item or group, specify the group of roles that should have access rights to the PII to perform their work (referred to as "white list") and/or a complementary list of roles that should access only masked values ("black list" end users that view masked data when accessing the sensitive information).

Examples of white or black list roles include call center employees, procurement team members, production support team, IT developers, and outsourced or offshore teams.

Resources and Expected Duration

- **Involved personnel**
  - Security team members: Provide PII characteristics and categorization; map user into appropriate roles and categorizing these roles into white and black lists.
  - Application and infrastructure team members: Provide an exhaustive list of locations, environments, applications and databases. Collect information for each environment on the ways to retain user authentication, roles, and responsibilities.

- **Estimated duration**
  - 1-4 weeks based on the number of environments and the number of sensitive information items in scope.

Here is a sample of the information summary collected upon completion of this phase:

<table>
<thead>
<tr>
<th>End user roles</th>
<th>Location (different locations might require different regulations)</th>
<th>Environment</th>
<th>Applications and tools</th>
<th>PII type and pattern</th>
<th>The level of security risk</th>
<th>DDM Security Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR employees</td>
<td>US</td>
<td>Datawarehouse</td>
<td>PeopleSoft production</td>
<td>SSN</td>
<td>Informative</td>
<td>Block</td>
</tr>
<tr>
<td>Clerks</td>
<td>EU (under the laws of the EU Data Protection directive)</td>
<td>PeopleSoft User Acceptance Testing (UAT)</td>
<td>Reporting tools – Business Objects</td>
<td>Client names</td>
<td>Low</td>
<td>Hide</td>
</tr>
<tr>
<td>Production support team members</td>
<td>Asia Pacific</td>
<td>Billing</td>
<td>Development tools (Toad, SQL developer)</td>
<td>Client addresses</td>
<td>Warn</td>
<td>Warn</td>
</tr>
<tr>
<td>Developers</td>
<td>Latin America</td>
<td>Billing clones</td>
<td>SAP Production</td>
<td>Employee names</td>
<td>Medium</td>
<td>Quarantine access</td>
</tr>
<tr>
<td>Application DBAs</td>
<td></td>
<td></td>
<td></td>
<td>Credit cards</td>
<td>High</td>
<td>Mask/scramble</td>
</tr>
<tr>
<td>QA teams</td>
<td></td>
<td></td>
<td></td>
<td>Health care information</td>
<td></td>
<td>Hide</td>
</tr>
<tr>
<td>Outsourced and offshore workforce</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Alert</td>
<td>Audit</td>
</tr>
</tbody>
</table>
**Discovery Phase**

The purpose of this phase is to create a list of the environments, applications, databases, tables, and columns that contain sensitive and personal information items.

Informatica Discovery was built to create a definitive list of environments, databases, tables, views and columns that contain PII.

It uses the PII patterns to initiate a discovery scan across your target environments, resulting with a list of identified items.

**Resources and Expected Duration**

- **Involved personnel**
  
  *Application and infrastructure team members*: Set-up and execute the Informatica Discovery solution on the exhaustive list of locations, environments and databases.

- **Estimated duration**
  
  An average of two days per application multiplied by the number of applications in the scope.

- **Phase result**
  
  A list of locations, environments, applications and databases with the corresponding information items found.

**Application Mapping**

The purpose of this phase is to provide a concise list of application screens that provide access to PII. The list will be used for creating the DDM rules as well as for acceptance testing.

Protecting business applications requires basic understanding of the data model as well application architecture and the authentication model. The first step begins by creating a list of the application screens that present PII. This list can be defined by the application users, application designers, developers, or even the application trainers.

Document the list of screens, including the application navigation path and sample data that can be used by the DDM administrators to define the security rules.

Here is a sample of the information summary collected upon completion of this phase:

<table>
<thead>
<tr>
<th>Application</th>
<th>Screen name</th>
<th>Navigation sequence</th>
<th>PII</th>
<th>Drill down screen options</th>
<th>Insert/ Update requirements and restrictions</th>
<th>Sample data</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeopleSoft</td>
<td>Customer inquiry</td>
<td>Login-&gt;O - &gt;C-&gt;I-&gt;N</td>
<td>Customer name, SSN, Customer address, Account number</td>
<td>1. Customer details</td>
<td>Cannot update customer records</td>
<td>Use Customer numbers 1000094 - 9999999</td>
</tr>
</tbody>
</table>

**Resources and Expected Duration**

- **Involved personnel**
  
  *Application team members*: Identify and build a thorough list of application screens.
• **Estimated duration**
  An average of three days per application (for about 10-20 sensitive PII items) multiplied by the number of applications in the scope.

• **Phase result**
  A list of application screens within business applications and databases with the corresponding information items found.

### Install and configure DDM

The purpose of this phase is to install and configure DDM on a development environment that will be used to define and test the security rules.

It is highly recommended that you install DDM on an application environment that will be exclusively designated for DDM implementation purposes and not be simultaneously used by other IT teams because creating DDM rules might involve performing “trial and error” tests several times and carefully investigating the application's detailed SQL logs. Working on a dedicated environment will ensure quick creation and testing of the rules.

DDM provides two main installation modes - installed on each of the database servers or on a dedicated server (referred to as “Central Hub Configuration”) that will act as a proxy for multiple environments/databases.

For rule creation purposes, it is better to install DDM on the database server as this configuration does not require you to change the application routing file (by installing on the database server you simply replace the database default listener port), and provides insight regarding all other tools and programs that extract PII information that have not been identified before.

### Resources and Expected Duration

- **Involved personnel**
  *Application and infrastructure team members:* Installing DDM.

- **Estimated duration**
  An average of one day per DDM installation and configuration.

- **Phase result**
  DDM Server is ready to proxy application SQL requests.

### Building DDM Security Rules

The purpose of this phase is to build DDM security rules that ensure PII is secured while allowing authorized application usage.

Building DDM security rules requires careful planning and continuous testing of the impact of these rules on the application.

The DDM rule set is built from a rule tree that is applied top-to-bottom on incoming SQL requests.

In general, typical security rule sets in DDM have the following sections:

a. **Auditing and alerting section:** Contains rules for auditing access to PII, regardless of the actions that will eventually be applied to the specific user request. Different types of alerts can be defined upon specific types of unauthorized PII requests.

b. **User/application authorization section:** Contains a definitive list of approved/unapproved combinations of users and programs/applications. Unauthorized users using certain applications or tools can be blocked or warned while auditing and alerting relevant users.
c. **Data Definition Language (DDL) control section**: Contains rules for controlling DDL commands (e.g., create and truncate commands) that can have users create objects that contain PII. Controlling who, when, and by what application new objects can be created and validating the new objects with a predefined authorization list and DDM rules maintains the ongoing high level of PII protection. The control on new object creation is done using DDM actions such as blocking, auditing, and validating against a list of approved maintenance tasks or alerting the DDM administrator. Advanced DDM actions can even include authorizing different DDL commands based on the content of an internal table or based on a web service call.

d. **Blocking section**: Contains rules that block users from accessing PII based on a combination of user groups, programs, PII information, and conditions (for example, blocking queries that try to extract PII based on a client name, address, or SSN). This folder will identify and block queries that use unauthorized conditions in incoming requests.

e. **Masking/scrambling/hiding PII Section**: The masking section contains rules with a list of column names and/or object names containing PII coupled with various mask/scramble/replacement functions. In addition, it can include additional rules for limiting the number or the content of specific PII rows retrieved or completely hiding PII results. In the following section we will provide the best practices required to define masking rules within business applications.

**Building Masking Rules**

In this section we provide detailed best practices on how to create masking rules, as they are the most complex to define. For information about defining other types or security and access control rules, review the DDM User Guide.

**Building DDM Security Rules for Reporting Environments and Datawarehouses**

Building security rules for reporting environments requires simply creating a masking rule with the PII table and column list with the appropriate masking function. The masking rule is much simpler then masking business applications because reporting tools do not provide drill-down or data update capabilities; implementation does not need to take into consideration reverse masking or preventing unintended masked-value updates.

**Note**: DDM has a set of instructions and rule examples for securing Business Objects, Cognos, Microstrategy and OBIEE.

**Building DDM Security Rules for Business Applications**

DDM provides a set of accelerators for packaged applications such as SAP, Siebel, PeopleSoft, and others. These accelerators include a set of predefined rules for securing certain modules and screens based on a list of responsibilities, roles, and user names.

Masking other packaged or home-grown applications requires careful rule setup to ensure:

1. **Valid masking functions**: A masking function creates a value that is acceptable to the application field validation. For example, if the customer number field accepts only numbers, masking the original customer number into ‘xxxxxxxx’ will cause an error message and the masking value will not be presented.

   **Ensuring valid masking functions**: Define a mask function that transforms numbers into numbers and characters into characters. Also verify that the limit of the data length is not exceeded. Try different masking techniques until the field is masked without any error. In the case where the
mask action is not masking the field as expected, use the logging of all “select” requests that the application submits to the database once the screen is populated to identify the correct table/columns that contain the relevant PII. The specific screen might be querying a view that was not identified in the mapping step and auditing the actual SQL request that the screen submits will enable you to identify it.

2. **Allow master-detail screen access – applicable only for primary-key based PII (e.g., customer number, bank account number)**

   Application screens allow drill-down capability to retrieve more details. For example, when accessing the general customer inquiry screen, the customer name, customer number, SSN, and customer address are masked appropriately, BUT - the general customer screen allows drilling down for accessing additional customer details such as customer orders and customer balance, which fail to retrieve relevant records as it uses the masked values.

   When the customer number is masked, the drill down operation uses the masked value and fails, causing the application to falsely reply that the customer has no details or orders.

   The reason for the application to error is that the application falsely thinks that the customer number equals to the masked number. Thus, when trying to retrieve additional information it uses the masked number instead of the real customer number. As the masked number is not a real customer, it has no real orders or details assigned to it – causing the application to return the error.

   **Allow master-detail screen access**

   When a primary PII is masked and drill-down returns no rows, use a reversible mask function. An example is “reverse transform” – replacing both the order of the digits as well as the values in a consistent and reversible form.

3. **Void masked data updates**

   As masked values are presented on the screen, saving or updating any data element in the screen might cause the update of the original PII value with the masked fictitious PII value. This is a common application behavior that occurs in SAP and PeopleSoft.

   DDM has been built to prevent unintended updates to masked PII values. When you create a masking rule in DDM for a certain table/column, by default any update request submitted to the masked table/column will be blocked while returning a custom message back to the user “Cannot update masked field “…”. See Ex 1

---

Ex 1

![Image of SQL command execution](image-url)
As in some applications, the block can prevent updates to legitimate data items; the default block must be overridden by adding an "update bypass rule" before the mask rule. The "update bypass rule" will take precedence to the masking rule, thus preventing the "update" request to reach the masking rule and getting blocked.

**How to build correct masking rules while ensuring that the masked data is presented, drill-down available, and that the masked values are not being updated?**

By creating a log-all rule and opening the appropriate application screens, you gain in-depth understanding of the appropriate rules that solve these issues. The log-all rule will populate the rule.log file with the detailed SQL requests that the application submits to the database, ensuring that the initial "select" request, the following drill down request "select...where customer_number = 'xxx'" and any update requests are identified, masked and updated correctly.

We have included a real life example of creating such a rule set using PeopleSoft. **Note**: it is important to get as a clean log as possible that is correlated with the application user screen actions. To achieve that, ensure that your application environment is not used by anyone except you during the rule building exercise.

**PeopleSoft Application Masking Example for Bank Account Number**

This is an example of a masking exercise for securing bank account numbers (a major account identifier) within PeopleSoft transaction screens. The example includes masking bank account numbers while ensuring that both drill-down and updates remain intact.

We have identified the relevant ‘search’ screen that must hide bank account information, and installed and defined a log-all rule in DDM.

Choosing the first bank ID presents masked bank accounts: Ex 2 & Ex 3
Ex 2

When drilling down, PeopleSoft correctly retrieves detailed records, while continuing to display masked bank account information.

Ex 3

When drilling down, PeopleSoft correctly retrieves detailed records, while continuing to display masked bank account information.
Ensuring insert and update transactions while keeping the bank account value masked in the various insert/update screens. We initiate a new transaction in the following screen, selecting the bank: Ex 5

Ensuring that the insert/update is correctly done with the reverse masked bank account value by adding the transaction reference ID and date (bank account number is already masked): Ex 6
Ex 6

Filling in transaction details, while ensuring that the bank account number is still masked: Ex 7

Ex 7

Adding drill down information for the new transaction and saving it: Ex 8
Retrieving the new inserted record presents it as it should be (while the bank account is still masked): Ex 9

Updating the reference field to “UPDATING VALUE” is saved correctly to the database with the correct reverse masked original bank account number: Ex 10
While searching it returns the expected values (with masked bank account number as expected): Ex 11
The rule structure to support the PeopleSoft screen masking includes the following rules (top to bottom):

1. **Bypass rule:**
   A rule at the top of the rule tree to remove frequently executed SQL requests from the logging and rule processing. In every application, the application server sends "keep alive" SQL requests frequently. Here is the regular expression pattern used to identify these SQL requests:
   ```regex
   .*\(SELECT Q._PRCSTYPE|UPDATE PS_SERVERACTVTY|FROM PS_SERVERACTVTY|FROM PSSERVERSTAT|UPDATE PSSERVERSTAT|UPDATE PSANALYTICREG\).*
   ```
   A bypass rule at the top of the rule set causes these requests to immediately be forwarded to the database without any further logging or rule processing.

2. **Update request reverse masking:**
   The update request is used to reverse mask any masked value that might appear in the UPDATE transaction. The UPDATE transaction that was recorded in PeopleSoft is:
   ```sql
   UPDATE PS_BNK_RCN_TRAN SET TRAN_REF_ID=@P1, DTTM_ENTERED=@P2 WHERE BNK_ID_NBR=@P3 AND BANK_ACCOUNT_NUM=@P4 AND TRAN_REF_ID=@P5 AND TRAN_DT=@P6
   ```
   As the value of the `@P4` is masked, reverse masking is applied by creating a rewrite rule:
   ```sql
   UPDATE PS_BNK_RCN_TRAN SET TRAN_REF_ID=@P1, DTTM_ENTERED=@P2 WHERE BNK_ID_NBR=@P3 AND BANK_ACCOUNT_NUM=dbo.DDM_MASK(@P4) AND TRAN_REF_ID=@P5 AND TRAN_DT=@P6
   ```
   And the actual rewrite rule used:
The insert command is also masked by rewriting the original insert into:

```
INSERT INTO
PS_BNK_RCN_TRAN(BNK_ID_NBR,BANK_ACCOUNT_NUM,TRAN_REF_ID,TRAN_DT,BUSINESS_UNIT,TRAN_AMT,CURRENCY_CD,TRAN_DESCR,RECON_TRANS_CODE,RECON_TYPE,RECON_STATUS,RECONCILE_OPRID,RECON_CYCLE_NBR,RECORD_SEQ_NUMBER,ACCTG_TMPL_ID,BUILD_ACCTG_STATUS,TRA_PROCESS_STATUS,PROCESS_INSTANCE,OPRID,DTTM_ENTERED,REC_ON_RUN_ID,BANK_ACCT_RVL_AMT,DOC_TYPE)
```

```
UPDATE PS_BNK_RCN_TRAN SET TRAN_REF_ID=@P1,DTTM_ENTERED=@P2 WHERE BNK_ID_NBR=@P3 AND BANK_ACCOUNT_NUM=@P4 AND TRAN_REF_ID=@P5 AND TRAN_DT=@P6
```

Identification method
- String
- Wildcard
- Regular Expression
- Case sensitive

Keep matcher result
Then try to match once every 3600 seconds per session

Action
- Action Type: Rewrite
- Alternate Statement
  UPDATE PS_BNK_RCN_TRAN SET TRAN_REF_ID=@P1,DTTM_ENTERED=@P2 WHERE BNK_ID_NBR=@P3 AND BANK_ACCOUNT_NUM=to.ddm_mask(@P4) AND TRAN_REF_ID=@P5 AND TRAN_DT=@P6

Whenever this rule is matched then
- Log when rule is applied

Stop if applied
VALUES(@P1, dbo.DDM_MASK(@P2), @P3, @P4, @P5, @P6, @P7, @P8, @P9, @P10, @P11, @P12, @P13, @P14, @P15, @P16, @P17, @P18, @P19, @P20, @P21, @P22, @P23).

Note that other insert and update requests that have not been identified will be blocked by the following Mask Action, until they are identified and added to the list of approved insert/update requests. It is also a good practice to add a blocking rule after the mask action, just in case an insert/update was not blocked by the masking rule and still needed to be blocked before updating the database with a masked Bank Account Number value.
3. Masking rule:
The masking rule uses the mask function on the Bank_Account_num field across all PeopleSoft tables:

![Edit Rule window](image)

4. Allowing drill-down functions with masked values:
The following rules have been created in order to allow drill-down while using the masked value as a condition for further information retrieval.
The reverse mask function is applied on the “where” clause condition in the following way (for conditions using literals):

```
BANK_ACCOUNT_NUM\s*=\$\"(w+)\"
```

Identification method: Regular Expression

Whenever this rule is matched then
- Log when rule is applied
And a second rule for fixing conditions that use bind parameters:

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverseMaskParam</td>
<td></td>
</tr>
</tbody>
</table>

**Matcher**

<table>
<thead>
<tr>
<th>Matching Method</th>
<th>Any</th>
</tr>
</thead>
</table>

**Action**

- **Action Type**: Search & Replace
- **Search text**: `BANK_ACCOUNT_NUM\s*=\$*(@\(\w+\))`
- **Identification method**: Regular Expression
- **Case sensitive**: False
- **Replacement string**: `BANK_ACCOUNT_NUM=dbo.DDM_Mask(@1)`

**Whenever this rule is matched then**

- **Log when rule is applied**

[OK]  [Cancel]
Resources and Expected Duration

- **Involved personnel**
  - *DDM Administrator*: Create SQL audit, analyze and define security rules.
  - *Application testers*: Work with the application screens to create and validate application SQL audit and masking.

- **Estimated duration**
  An average of five days per application (for about 10-20 sensitive PII items) multiplied by the number of applications in the scope.

- **Expected result**
  Validated DDM rule set that successfully audits user access and protects sensitive information items by masking or blocking them from unauthorized roles.

Capturing User Context Identification Information

The Dynamic Data Masking solution acts like an in-line database network protocol proxy between applications and databases, intercepting and applying security rules on incoming SQL requests before they reach the database.

Dynamic Data Masking rules identify user context (e.g., OS user, client host name), user LDAP/Active Directory grouping, and in predefined applications, the responsibilities and roles (e.g., SAP and PeopleSoft user roles).

The user context is used to selectively apply various actions, such as masking or scrambling sensitive information, hiding certain sensitive rows and columns (row level security), and blocking unauthorized requests while notifying the user or a security official and collecting a detailed audit trail. In development and DBA tools such as Toad, SQL Server Management Studio, PL/SQL
Developer, and SQL *Plus, the client is connecting directly to a database, so the client information is passed to the database and is easily captured by DDM as well. When the client is an application, connected using a single database user and a connection pool, the client information becomes irrelevant to the actual application end-user accessing the application through a web-browser (the user context captured by DDM in this case is the application server host name, IP address and, a single database login user name, e.g., username: apps).

In the following table we have collected various user context identification methods that have been used by DDM and options for capturing the user context based on the session_id of the SQL request.

<table>
<thead>
<tr>
<th>Source of client identification</th>
<th>DDM Retrieval Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client information is continuously updated within the database v$session views (in Oracle) or sysprocesses (for SQL Server). Applications that update the view include PeopleSoft, Siebel, and Oracle applications.</td>
<td>Using either a PL/SQL matcher or a Java matcher, DDM uses the session_id to fetch in real-time the user context information from the database view (v$session) or attaches a database function to the original SQL request, when the user context is found within sys_context, (Oracle VPD functions).</td>
</tr>
<tr>
<td>From an SQL remark that is inserted by the application into the original SQL request. The remark includes the user name and other descriptive information. This method is commonly used across reporting tools such as Business Objects, Cognos, Microstrategy and SAS.</td>
<td>Dynamic Data Masking parses the incoming SQL request and extracts the user name into a symbol. The symbol can be used in the Rule Engine for correlating with LDAP/ActiveDirectory.</td>
</tr>
<tr>
<td>From a program/application API such as a web service that retrieves the user context per database session_id.</td>
<td>Using the Java matcher or PL/SQL matcher, DDM executes the program based on the session_id to receive the user context. For example, in SAP, DDM gets the user context by executing a JCO program that returns the name/responsibility based on the session_id.</td>
</tr>
<tr>
<td>From applications that do not perform connection multiplexing and send the end-user name information as part of the initial user logon sequence.</td>
<td>Applications store the user/role/privileges on the database. In order for the application to get the user credentials, as part of the login sequence, the application submits an internal SQL request with the user login name. Using a database function, DDM can intercept the user name and if the user consistently uses the dedicated connection, DDM can correlate the connection with the user and apply rules accordingly.</td>
</tr>
</tbody>
</table>
Creating a dedicated application server instance to be used by the unauthorized users. DDM will mask ALL user requests generated from this specific application server connection pool – so specific user context identification is not needed. This instance will use a dedicated connection pool for these unauthorized users that will be routed through DDM. DDM will mask all users that use these connections. This method can be used to protect outsourced personnel. These users will use a different application server instance that will be routed to DDM. All other internal users will not be routed through DDM (bypassing the DDM proxy). Note: This requires an infrastructure change - creating a different application server instance and routing other users to it.

Changing application source-code to add the user context as a remark in the SQL request.

Changing application source-code to embed the user remark into the incoming SQL request.

In case you are not aware of the authentication mechanism of your home grown application, define a log_all rule and audit a single user’s logon sequence to the application. It is expected that you will trace SQL requests that provide the end-user name to the database authentication tables. This will allow you to identify and capture the end-user while applying the appropriate security rules.

**Resources and Expected Duration**

- **Involved personnel**
  - *DDM Administrator*: Create user-role identification rules.
  - *Application and infrastructure team members*: Provide user-role context identification information or application login SQL audit.

- **Estimated duration**
  An average of 1-2 days per application.

- **Expected result**
  Creating rules for capturing user authentication to assign the appropriate role set.

**Production Deployment Best Practice**

Install DDM on target production environments.

Identify batches and irrelevant transactions and define rules to have them bypass DDM to remove unrequired load from the DDM server.

Import predefined rules from the test DDM environment into production with a two phase process:

a. **Audit phase**: Import rules and disable all masking/blocking actions, transforming them into mere audit, ensuring their correctness. During the first couple of days, the audit will ensure that only relevant sensitive data item usage is audited and all unauthorized roles are identified correctly.

b. **Audit and masking phase – go live**: Change the audit action to include masking and blocking. Review the logs daily to ensure that masking and blocking only apply to unauthorized requests.

**Deployment Options:**

DDM can be deployed in two configuration models. Here are details which will allow you to determine which one is best suited for your application landscape:
Central Hub Configuration Implications

Installing DDM on central hubs (usually a cluster of servers) is best suited in the following cases:

- **Large number of small applications** either in production or non-production. Central installation will save the time and effort of local installations.

- **Protecting IT personnel, DBA and developer access using development and DBA tools such as Toad, SQL *Plus, SQL Management Studio, etc.** Installing DDM on a dedicated server hub enables you to increase the level of security by applying network segmentation. Network segmentation ensures that production databases are not accessible from the developer’s network segment and ONLY THROUGH DDM. This can be configured by defining a firewall rule that segregates the production database networks from the non-production ones.

- When a specific database server experiences frequent CPU outage, adding even the smallest DDM CPU footprint (less than 2%) is a problem. By installing DDM on the hardware-dedicated hub, no additional consumption is added to the already overloaded database server. Note that performance-improving rules can also be added to DDM, such as rewriting badly performing SQL requests, blocking reports with high costs during database peak times, automatically off-loading reports to a replication server, etc.

**Note:** After installing and configuring DDM, you need to change the application target database host and post number to point to the DDM hub. This involves changing the jdbc/odbc/tnsnames configuration file entry and restarting the application to restart the connection pool to be established through DDM.
**Database Server Installation Implications**

Installing DDM on the main database servers is best suited for securing a few large central applications or data warehouses with thousands of concurrent users.

The main benefit of this approach is that there is no need to make changes to the application target database host and port number to point to the DDM (as DDM will be configured to listen to the default database listener port, replacing the database listener to listen to a hidden port instead). Restarting the application is also needed in this configuration because we need to restart the connection pool to be established through DDM.

**Hardware and Operating System Requirements for DDM Server**

Dynamic Data Masking is a Java solution that can run on any Windows, Linux, and Unix system. It can be installed on a central hub (supporting hundreds of databases in a single installation) or on each database server.

The Dynamic Data Masking server consumes CPUs mostly for proxy work. The memory and disk requirements are minimal. The Dynamic Data Masking Server requires approximately 1 GB for memory and some disk space for audit purposes.

**Initial DDM Server CPU Consumption Estimation**

DDM consumes about 3% of the database CPU core count for rule processing.

For example, if you have 5 DB's, each with 24 CPU cores, the DDM server that will be needed to support all DB's will consume a 24x5x0.03 = 3.6 -> a full 4 core CPU. In this case, our recommendation would be to install DDM on a cluster of two Linux servers, each with a Quad core CPU and doing load balancing between the nodes, with failover for high availability (consuming 50% of each server's quad core CPUs).

**Detailed CPU Consumption**

Use the following calculation when required to provide a detailed CPU consumption for the Dynamic Data Masking Server use (for example, when designating a single DDM Server to support tens of different applications with varying loads).

DDM CPU consumption is linearly proportional to the SQL packet traffic that routes through it. Therefore, to calculate CPU consumption, you must determine the total packet traffic that a client and server send and receive per second (the round trip value). The total traffic includes all SQL traffic as well as remote database calls. So, if you use Dynamic Data Masking to intercept remote database calls, you need to include the round trip value for that traffic as well.

Equations to calculate the CPU usage:

**Variables**

- X1 = Database protocol round trip client/server traffic
- X2 = Remote database round trip traffic
- PR = Total round trip packets per second

**Equations**

\[ PR = (X1+X2)^2 \]

\[ CPU \ Usage = PR/10,000 \]
The final CPU usage value returns a Gigahertz (GHz) value.

**Note:** Applications and batches that use the direct routing action do not impact CPU consumption; therefore, you need to subtract their traffic from the final value.

**Resources and Expected Duration**

- **Involved personnel**
  *Application and infrastructure team members:* Installing DDM.

- **Estimated duration**
  An average of one day per DDM installation and configuration.

- **Phase result**
  DDM Server is ready to proxy application SQL requests.

**On-going Rule Management and Accountability**

Using easy-to-maintain rule sets that are built using readable folders makes the on-going maintenance of DDM a simple and straightforward task that can be done by a security administrator.

Rule maintenance updates are required for two main purposes:

1. **Additions to the mapped PII:** Following ever-changing privacy regulations, more objects are categorized as PII. This change requires adding new environments, application screens, tables and column names into the list of PII.
   Generally, rule changes are infrequent and usually occur on a monthly (or longer) basis and take a few days for the DDM administrator to identify (using the Informatica Discovery tool) and update.

2. **Creation and changes of application screens and objects (tables, snapshots, views) containing PII:** DDM has two complementary approaches on how to manage creation of new objects containing PII.
   The first approach uses rules identify, audit, and control the creation of these objects by the end-users in real-time. The security administrator can immediately be updated upon the creation of these new objects.
   The secondary approach (which is complementary to the first approach) is based on maintaining two object lists that are extracted at a predefined time interval (hourly, daily or weekly):
   a. The list of objects/columns that are defined in the DDM rules.
   b. The list of new or changed database objects (based on a data dictionary extract).
   An Informatica Discovery process can run on the results of the data dictionary changes to scan column names and content for PII based on patterns, resulting in a list that will be updated in the rules accordingly. These changes in production environments can require a couple of hours per week for the DDM administrator to complete.
   A report that presents the list of PII objects and the related DDM rules that reference these objects can be monitored and reported at any time interval to auditors.

**Author**

Alon Rosenthal