DELPHIX

WHITE PAPER

Data Masking with the DevOps Data Platform

Industry Background

Data masking has never been more relevant. With data breaches continuing to make headlines and the emergence of stringent data privacy regulations, it's imperative that businesses across all industries manage their data with greater caution and sensitivity. Not protecting personal, health, and sensitive information in compliance with data privacy regulations, such as GDPR, LGPD, and HIPAA, results in heavy fines and lasting reputational damage.

Exacerbating the challenge of protecting confidential data is the rapid increase in enterprise data volumes, particularly as data sprawls across environments used for development, testing, analytics, and other "non-production" use cases. Recent research estimates that for every copy of production data, businesses typically create over ten copies that multiply their overall surface area of risk. Security-minded organizations are adopting data masking as a solution for protecting these copies. In fact, masking technology is fast becoming a part of the reference architecture for organizations seeking a holistic approach for managing and securing data across the entire enterprise.

Solution Overview

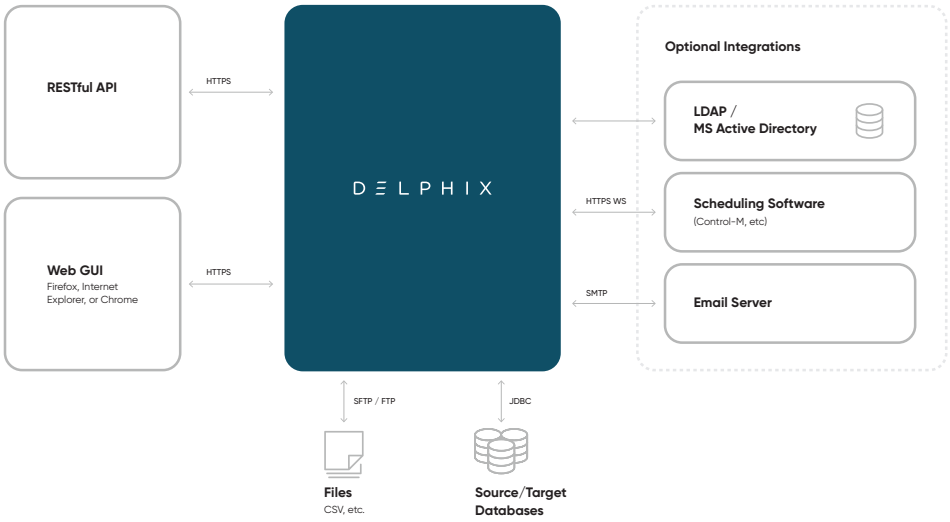
The DevOps Data Platform provides a comprehensive approach to data masking that meets enterprise-class performance, scalability, and security requirements. Delphix enables businesses to successfully protect sensitive data through these key steps:

- **Profiling Sensitive Data:** Identify sensitive information such as names, email addresses, and payment information to provide an enterprise-wide view of risk and to pinpoint targets for masking.
- **Securing Sensitive Data:** Apply masking to transform sensitive data values into fictitious yet realistic equivalents, while still preserving the business value and referential integrity of the data for use cases such as development and testing. Unlike approaches that leverage encryption, masking not only ensures that transformed data is still usable in non-production environments, but also entails an irreversible process that prevents original data from being restored through decryption keys or other means.
- **Scaling and Integration:** Extend the solution to meet enterprise security requirements and integrate into critical workflows (e.g. for SDLC use cases or compliance processes).

Taken together, these capabilities allow businesses to define, manage, and apply security policies from a single point of control across large, complex data estates. Delphix can enable global operations with support for international addresses and character sets. Moreover, Delphix masking is quickly configured and deployed via GUI-driven workflows without requiring any specialized programming expertise or lengthy services engagements.

How Delphix Masking Works

An instance of the Delphix DevOps Data Platform—a Delphix "engine"—is a self-contained operating environment and application that is provided as a virtual appliance certified to run on a variety of platforms including VMware, AWS, and Microsoft Azure. Delphix's graphical interface can be accessed from web browsers including Internet Explorer, Firefox, or Chrome. It has a robust role-based controls system enabling organizations to apply fine-grained permissions over what users have access to and what tasks they can and cannot perform.



Profiling Sensitive Data

After connecting to a supported data source, Delphix identifies what data should be secured. Sensitive data discovery is performed using two different methods, column level profiling and data level profiling.

Column Level Profiling

Column level profiling uses regular expressions (regex) to scan the metadata (column names) of the selected data sources. There are several dozen pre-configured profile expressions designed to identify common sensitive data types (Social Security numbers, names, addresses, etc). Users also have the ability to write their own profile regular expressions.

Example: First Name Expression <(?>(fi?rst)_?(na?me?)|f_?name)(?!\w*ID)>

Data Level Profiling

Data level profiling also uses regex, but to scan the actual data instead of the metadata. Similar to column level profiling, there are several dozen pre-configured expressions and users can add their own.

US Phone No. Expression < ((\(?\b[0-9]{3}\)?[-.]?[0-9]{3}[-.]?[0-9]{4}\b)(?<![0-9]{6}[.][0-9]{4}))>

Delphix comes prepackaged with over 50 profile expressions developed after validation with dozens of F500 customers to help businesses discover over 25 types (account numbers, addresses, etc.) of sensitive data using both column and data level profiling.

Examples of Data Discoverable with Pre-Built Profiler Expressions

Account Numbers	Driver License Number	School Name
Physical Addresses	Email	Security Code
Beneficiary ID	First Name	Serial Number
Biometrics	IP Address	Signature
Certificate ID	Last Name	Social Security Number
City	Location	Tax ID
Country	Plate Number	Telephone Number
Credit Card	PO Box Numbers	VIN Number
Customer Number	Precinct	Web Address
Date of Birth	Record Number	Zip Code

Profiling jobs can be executed across multiple sources to provide businesses with an enterprise-wide view of sensitive data risk. When a data item is identified as sensitive, Delphix recommends specific masking algorithms to be used for securing the data.

Application and Regulation-Specific Profiling Templates

Delphix also offers profiling templates to identify data for specific application types (e.g. SAP, Oracle EBS, Salesforce) or data relevant to specific privacy regulations (e.g. GDPR, HIPAA). Profiling templates encompass sets of regular expressions for finding data commonly associated with apps/regulations, or a pre-built inventory of fields that Delphix needs to mask. By adding additional intelligence to the profiling process, businesses can eliminate manual discovery and validation, allowing them to quickly and accurately mask the right fields with the correct algorithms.

Securing Sensitive Data

Delphix's primary method for securing data is masking. Masking algorithms create a structurally similar but fictitious version of data that can be used for purposes such as application development and testing. Masking protects the actual sensitive information while generating a functional substitute for occasions when the real data is not required.

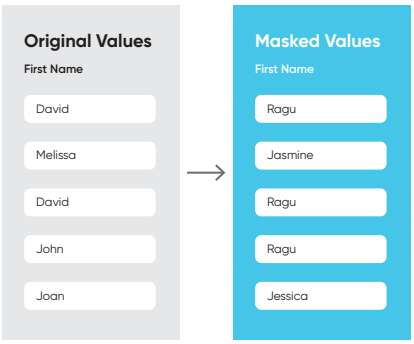
- **Delphix Masking – Is Irreversible –** Masked data cannot be "reverse engineered" and restored to its original unmasked state.
- **Creates Results Representative of the Source Data:** The output of Delphix masking resembles production data for non-production purposes. This could include geographic distributions, credit card distributions (e.g. leaving the first 4 numbers unchanged, but scrambling the rest), or maintaining human readability of (fake) names and addresses.
- **Preserves Referential Integrity:** Delphix has the ability to mask data consistently to maintain referential integrity. If an account number is a primary key and scrambled as part of masking, then all instances of that account number linked through key pairs will be masked identically. Additionally, the Delphix platform scales horizontally so that masking algorithms will preserve referential integrity across multiple, heterogeneous data sources (see Scaling and Integration).

Once sensitive data fields have been identified, Delphix automatically recommends an out-of-the-box algorithm for securing the data. These algorithms fall into one of the following frameworks:

Mapping Algorithm Framework

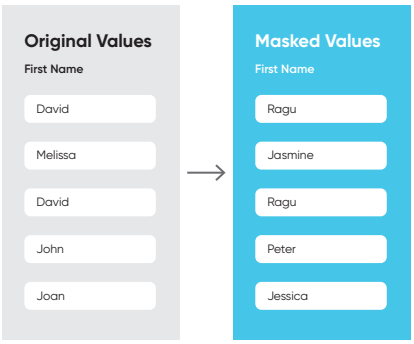
A mapping algorithm allows users to state what values will replace the original data. It sequentially maps original data values to masked values that are pre-populated to a lookup table through the Delphix user interface. To satisfy any uniqueness requirements, the algorithm maps data in a 1:1 fashion. Mapping produces no collisions in the masked data and the algorithm always matches the same input to the same output. For example “David” will always become “Ragu” with no other names masking to “Ragu.” The algorithm checks whether an input has already been mapped; if so, the algorithm changes the data to its designated output. Mapping algorithms handles arbitrary string data and preserves referential integrity.

Secure Lookup Algorithm



Secure Lookup preserves referential integrity: “David” is consistently masked to” Ragu”; It also creates collisions: both “David” and “John” mask to “Ragu.”

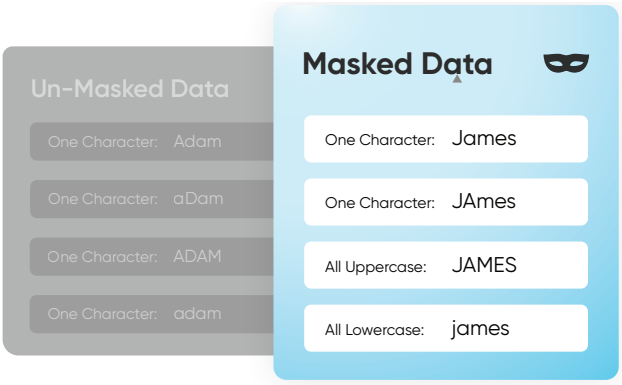
Mapping Algorithm



Mapping preserves referential integrity: “David” is consistently masked to “Ragu”; It satisfies uniqueness constraints: no other value will be masked to “Ragu” except “David.”

Secure Lookup

This is the most commonly used type of algorithm. It is easy to generate and works with different languages. When this algorithm replaces real, sensitive data with fictional data, it is possible that it will create repeating data patterns, known as “collisions.” For example, the names “Tom” and “Peter” could both be masked as “Matt”. Because names and addresses naturally recur in real data, this mimics an actual data set. Secure lookup handles arbitrary string data and preserves referential integrity. However, if you want the Masking Engine to mask all data into unique outputs, you should use Character Mapping.



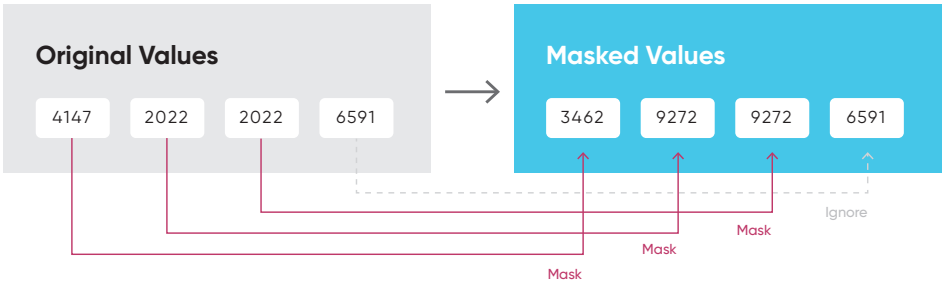
Character Mapping Framework

This algorithm maps text values, defined by a set of character groups, to other text values generated from the same character groups. Mappings are calculated algorithmically, so it is not necessary to provide the set of mapping values. The algorithm preserves any characters not assigned to a group. Any characters from the first Unicode plane can be mapped - this covers most characters used in modern languages. Other (supplementary) characters can only be preserved.

Description: Ability to mask non-ASCII characters

Segment Mapping Algorithm Framework

Segment mapping algorithms produce no overlaps or repetitions in the masked data. They let users create unique masked values by dividing a target value into a maximum of 36 segments and masking each segment individually. Businesses might use this method for information involving unique values, such as Social Security numbers, primary key columns, or foreign key columns. Segment mapping handles strings of a known format and preserves referential integrity.



Example Masking a credit card number in segments, preserving last 4 digits

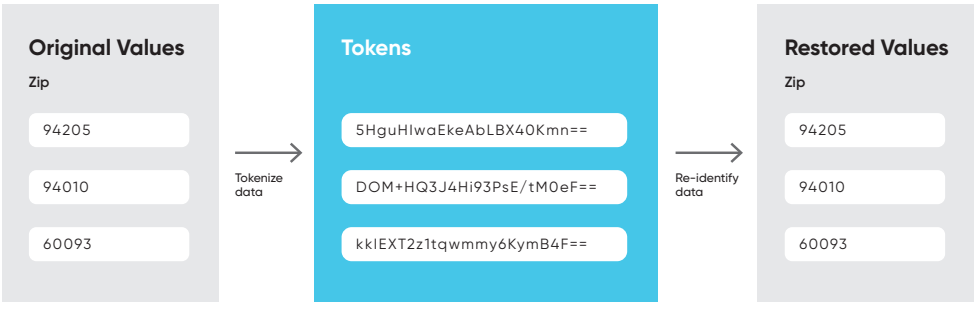
Binary Lookup Algorithm Framework

A binary lookup algorithm is similar to the secure lookup algorithm but is used when entire files are stored in a specific column. For example, if a bank has an object column that stores images of checks, Delphix can use a binary lookup algorithm to mask those images. Delphix cannot change data within images themselves such as the names on X-rays or drivers licenses. However, the algorithm can replace all such images with a new, fictional image. This algorithm masks binary columns with blob, varbinary, or image data and preserves referential integrity.

Tokenization Algorithm Framework

A tokenization algorithm is the only type of algorithm that allows for the reversal of a masked value back to its original value. For example, Delphix can a tokenize data before it is sent to an external vendor for analysis allowing the vendor to identify accounts that need attention without having any access to the original, sensitive data. Once the vendor provides feedback, tokenized values can be reversed, enabling the data owner to take action on the appropriate accounts.

Like mapping, a tokenization algorithm creates a unique token for each input such as “David” or “Melissa.” The actual data values are converted into tokens that no longer convey any meaning. Tokenization handles arbitrary string data and preserves referential integrity.



Partner processes data with original values obscured

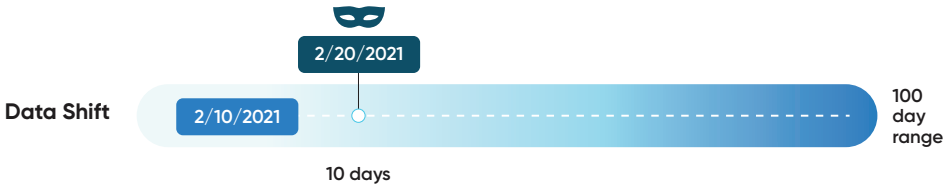
Date Replacement Framework

Delphix enables masking a date value based on specified beginning and end dates. Masked output values are calculated algorithmically using the algorithm's key, so rekeying the algorithm will cause a different output value to be generated for each input. It is also possible for an input to be masked to itself.

Description: The algorithm takes into account that three consecutive months are what is important to preserve. This could also be a specified number of hours, seconds, days, weeks, or years.

Date Shift Framework

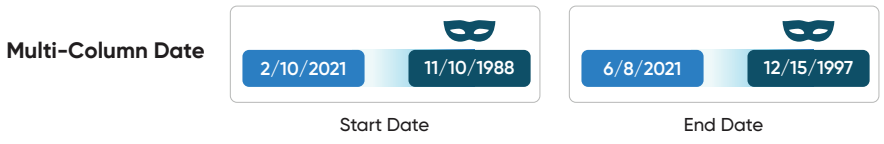
This algorithm masks date values to different dates based on a specified range around the input value. Masked values are calculated algorithmically using the algorithm's key, so rekeying the algorithm will cause different outputs to be generated for each input. All valid input values will be masked to a new value, and the new value will never match the input.



Description: In this example the birth year range needed to stay in 1950-1959.

Dependent Date Shift Algorithm

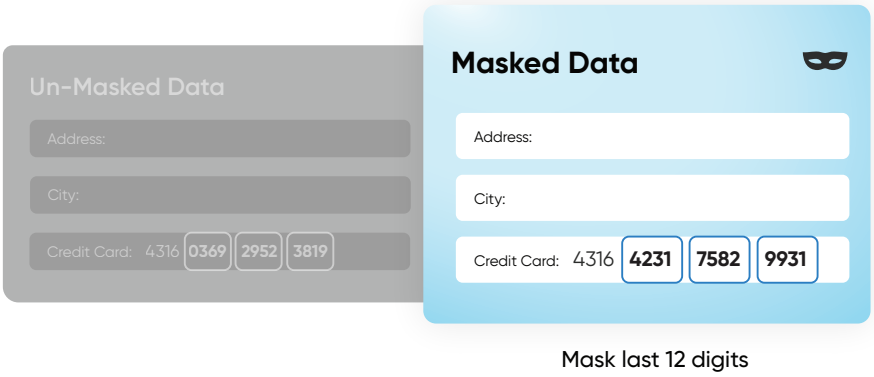
When a dependency exists between the two dates that must be maintained, this algorithm provides a method to manipulate dates together. Examples of this include date of admission and date of discharge or date of birth and date of death. If we were to attempt to mask these dates independently, we may end up with a situation where a later date such as date of discharge, was masked to be earlier than date of admission. If we were dealing with date of birth and date of death we may end up masking the values in a way that turns an 80 year old into a 5 month old.



Description: In this example, the rule was to always have the year of discharge four years after the year of admission.

Payment Card Framework

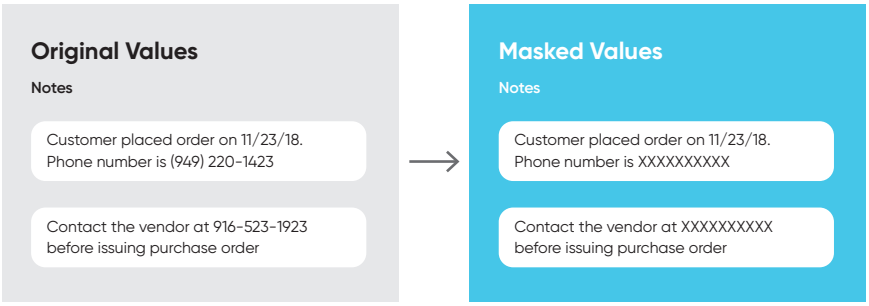
The payment card algorithm masks payment card numbers based on the starting digits to be preserved and the minimum number of positions to be masked. This framework is built on top of the Character Mapping Algorithm Framework with a character set of [0-9]. All characters outside of this character group remain unmasked. Masked values are calculated algorithmically using the algorithm's key, so rekeying the algorithm will cause different outputs to be generated for each input. The last digit may remain the same if the calculated check digit is equivalent to the last digit of the input. Any inputs with more than one digit will never mask to the original value.



Description: Masking credit card numbers can keep the starting digits preserved or mask the entire number.

Free Text Algorithm Framework

This algorithm removes sensitive data that appears in free-text columns such as "Notes."
For example, the algorithm can look for predefined strings such as "St," "Cir," "Blvd," and other words that suggest an address. It can also use pattern matching to identify potentially sensitive information. Once sensitive information within the text has been identified, the algorithm can hide or show information by displaying either a "black list" (designated values will be removed/redacted) or a "white list" (only designated material will be visible and other material will be removed/redacted). This algorithm handles arbitrary string data and does not preserve referential integrity.

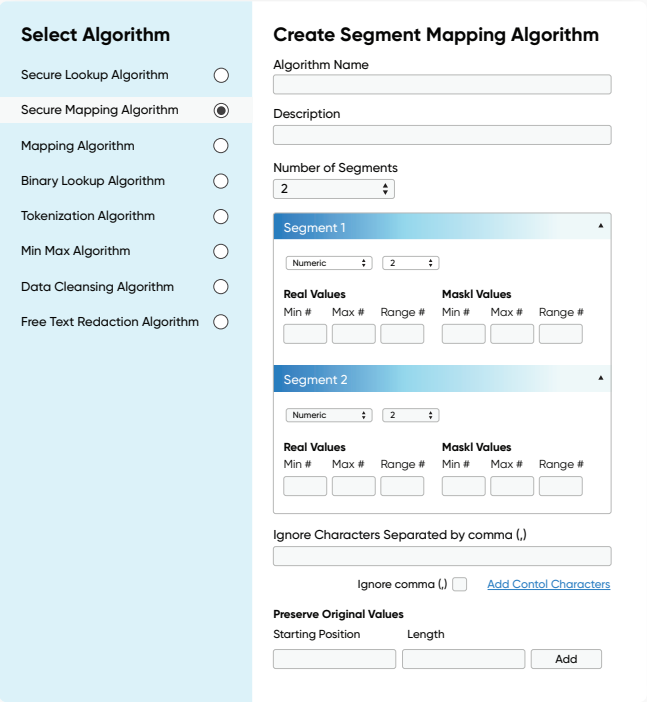


Example Redacting phone numbers in a 'Notes' column

Creating New and Custom Algorithms

Delphix gives businesses the ability to easily define their own masking routines if none of the default algorithms meet their requirements. Users first select an algorithm framework as a basis for a new custom algorithm, then define and modify algorithm properties through a GUI-driven process. For example, users can specify new lookup tables for the secure lookup algorithm, customize segments for a segment mapping algorithm, and determine value ranges for the min max algorithm framework.

Creating a Custom Segment Mapping Algorithm



If business requirements demand that data be transformed in a way that is not supported by the baseline product, a custom algorithm known as a mapplet can be created by Delphix Professional Services and imported into the Delphix platform. Mapplets are realized as JavaScript functions conforming to a specific format and are expected to compute a masked value given the inputted original value.

Masking SDK

The Masking SDK enables customers and partners to develop algorithm extensions using industry-standard tools, without the detailed Delphix masking product internals knowledge currently required to write custom algorithms. This provides customers and partners a direct means to build new masking algorithms in cases where standard Delphix algorithms cannot be easily adapted to meet customer requirements.

Executing Masking Jobs

Masking jobs are created via a GUI-driven workflow in which the user selects a target database, algorithms to use based on profiling results, resources allocated to the job, and, optionally, SQL statements to be run before or after execution of the job. Delphix can process and output masked data values in two different ways:

In-Place Masking: An instance of Delphix will read data from a source, secure the data within the engine and then update the data source with the secure data. In-place masking only transforms the columns flagged as containing sensitive information, leaving the other columns alone. Since this method potentially requires copying production data into a non-production zone while the masking takes place, sensitive data might exist in the non-production zone until the masking is complete.

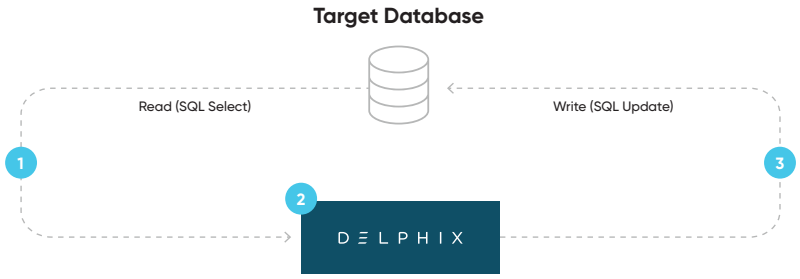


Figure 2. In-place masking

1. Read original unmasked data from target
2. Transform original values in memory
3. Update sensitive data values with mask values on target

On-the-Fly Masking: Delphix reads data from the data source, secures the data in the engine and then places the secure data in a target source (different from the location of the original data source) in an Extract Transform Load (ETL) process. Delphix extracts the data from a source environment, such as a production copy, gold copy, or disaster recovery copy (only reading from a database not an archived file). It masks the data in the memory of the application server on which it resides and then loads the masked data to the target environment.

Delphix does not modify the original source data; only the target data changes.

Key variables that influence masking performance are the number of tables to be masked, rows per table, columns per table, masking algorithm per column, data type, avg size per column, as well as indexes, constraints, and triggers on the masked table.

Figure 3. On-the-fly masking

1. Read original unmasked data
2. Transform data values in memory
3. Write masked data to target

Scaling and Integration

Delphix allows for multi-engine deployments that enable businesses to consistently mask data at scale across multiple, heterogeneous sources. In these scenarios, Delphix facilitates the synchronization of information defining masking jobs across multiple masking engines. This information can include the algorithms to be executed, data source connectors, metadata inventories, and the set of tables or files that an engine will run profiling, masking, or tokenization against. Engine synchronization provides a flexible way to move these masking "objects"—the algorithms and related information associated with a masking job—necessary to run an identical job on another engine.

There are two specific scenarios in which organizations benefit from orchestration between multiple masking engines. In the first, a multi-engine implementation addresses the problem of horizontal scale— achieving consistent masking across a large set of data sources by deploying multiple masking engines. The second architecture addresses the desire to author algorithms on one engine, to test and certify them on another, and finally to deploy them to a production engine.

Distributed Execution

For many organizations, the size of the profiling and masking workloads requires more than one production masking engine. These masking engines can be identical in configuration or be partially equivalent depending on the organization's needs. Syncable objects are authored on one engine, labeled "Control Masking Engine" in the diagram below. Those objects are then distributed to "Compute Masking Engines" using engine synchronization APIs. The synchronized algorithms and masking jobs will produce the same masked output on all of the engines, thus enabling large data estates to be masked consistently.

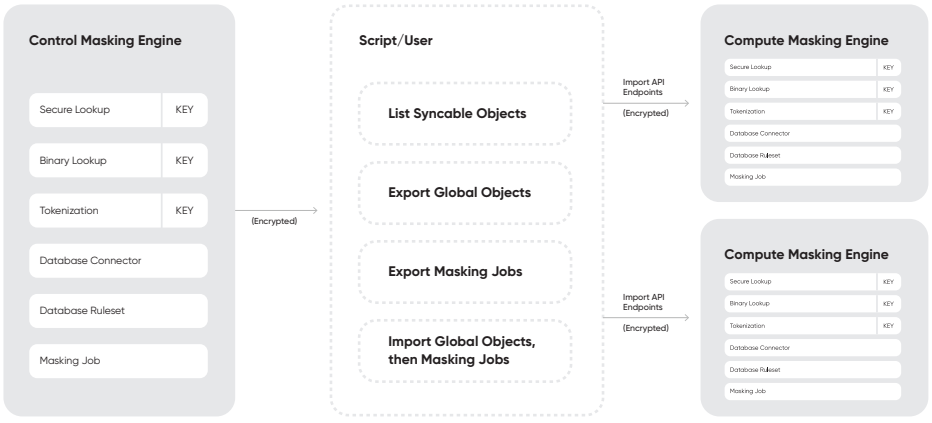


Figure 4. Synchronizing masking jobs across multiple engines

Software Development Life Cycle (SDLC)

Using an SDLC process often requires setting up multiple masking engines, each for a different part of the cycle (Development, QA, Production). Here, algorithms are authored on the first engine, labeled “Dev Engine” in the diagram below. When the developer is satisfied, the algorithms are exported from the Dev Engine and imported to the QA Engine where they can be tested and certified. Finally, they are exported from the QA engine and imported to the Production Engine.

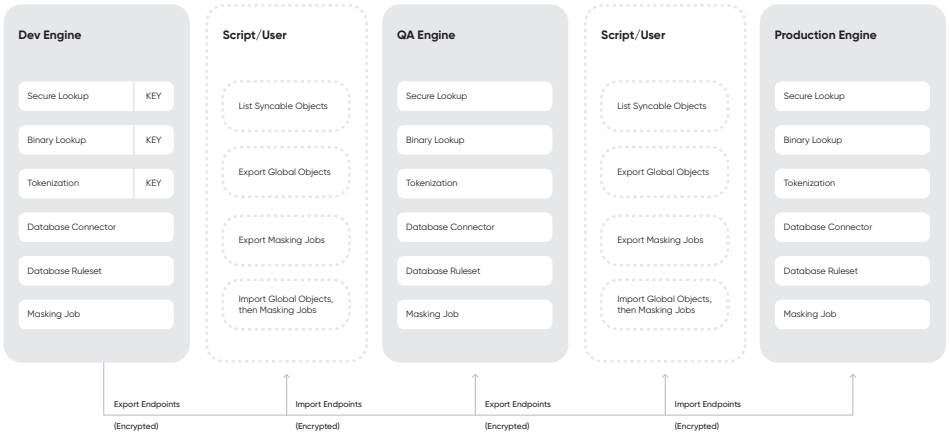


Figure 5. Coordinating engines across SDLC stages

API Integration

Delphix includes a robust, RESTful API set that allows teams to access and manipulate a programmatic representation of masking objects and resources using a predefined set of operations. The APIs use JSON (JavaScript Object Notation) to ingest and return representations of the various objects used throughout various operations. JSON is a standard format and, as such, has many tools available to help with creating and parsing request and response payloads, respectively. Businesses can leverage the APIs to programmatically control Delphix profiling and masking capabilities, integrating them into workflows such as:

- Triggering profiling or masking jobs as data changes, or as new data enters into data repositories
- Creating customized regulatory compliance reports that log masking activities
- Connecting to enterprise monitoring systems such as Splunk
- Synchronizing masking jobs across multiple Delphix engines

Data Source Support

Delphix masking supports profiling, masking, and tokenizing a variety of different data sources including distributed databases, mainframe, PaaS databases, and files. Delphix breaks up support for data sources into two categories:

Delphix Connectors: These apply to data sources that a masking engine can connect to directly using built-in connectors that have been optimized to perform masking, profiling and tokenization. Delphix has dedicated masking connectors for the following data sources:

- **Distributed Database:** DB2 LUW, Oracle, MS SQL, MySQL, SAP ASE (Sybase), PostgreSQL, MariaDB
- **Mainframe/Midrange:** DB2 Z/OS, DB2 iSeries, VSAM
- **PaaS Database:** AWS RDS Oracle
- **Files:** Excel, Fixed Width, Delimited, XML

Other data sources can be masked by uploading JDBC drivers to Delphix engines.

File Extract Mask and Load Sources (FEML): This method is used to mask and tokenize data sources that do not have dedicated Delphix connectors. FEML uses existing APIs from data sources to extract the data to a file, mask the file, and then use APIs to load the masked file back into the database. Additional data sources such as Informix, Azure SQL, Hadoop, Teradata, and many others can be masked using FEML.

Towards Secure Data Delivery

Collectively, Delphix masking capabilities allow businesses to define, maintain, and deploy a set of security policies from a single point of management. Delphix codifies what type of information is deemed sensitive based on different standards or regulations, determines exactly how that information is secured, and records security actions by logging all masking jobs.

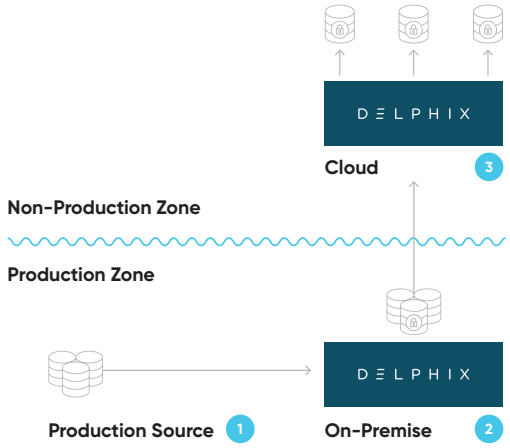


Figure 6. Deliver masked, virtual data copies to a non-production zone

1. Sync with source prod zone
2. Mask sensitive data
3. Deliver masked data to non-prod zone and provision copies

The Delphix DevOps Data Platform also combines data masking with the ability to virtualize data. Delphix can provision virtual copies of production data, apply masking within the confines of the production zone, then automatically deliver multiple masked copies to downstream environments for development, testing, analytics, or other non-production use cases. This supports efforts to accelerate key business processes while also increasing data governance: Teams can easily control who has access to what data, when, where, and for how long.

Data Masking with the DevOps Data Platform

DELPHIX

Delphix is the industry leading data company for DevOps.

Data is critical for testing application releases, modernization, cloud adoption, and AI/ML programs. We provide an automated DevOps data platform for all enterprise applications. Delphix masks data for privacy compliance, secures data from ransomware, and delivers efficient, virtualized data for CI/CD.

Our platform includes essential DevOps APIs for data provisioning, refresh, rewind, integration, and version control. Leading companies, including UKG, Choice Hotels, J.B. Hunt, and Fannie Mae, use Delphix to accelerate digital transformation. For more information, visit www.delphix.com or follow us on LinkedIn, Twitter, and Facebook.

000233