

Прототип решения для реализации статического маскирования данных средствами IBM Information Server

| | | |
|-------|--|----|
| 1 | Общее описание прототипа решения..... | 3 |
| 2 | Характеристика компонентов прототипа решения | 4 |
| 2.1 | Схема прототипа решения по маскированию данных..... | 4 |
| 2.2 | IBM Information Governance Catalog | 5 |
| 2.3 | IBM Information Analyzer | 5 |
| 2.4 | IBM DataStage | 6 |
| 2.5 | Конфигурационная программа «DsMask» | 6 |
| 2.6 | Оператор маскирования «DsMask» | 7 |
| 2.7 | Примеры дополнительных классов конфиденциальных данных..... | 7 |
| 3 | Встроенные алгоритмы маскирования..... | 8 |
| 3.1 | Хеширование без сохранения формата | 8 |
| 3.2 | Хеширование с сохранением формата | 8 |
| 3.3 | Алгоритмы подстановки и перемешивание исходных значений | 9 |
| 3.4 | Вызов алгоритмов Optim Data Privacy Providers Library | 10 |
| 3.5 | Вспомогательные алгоритмы..... | 10 |
| 3.6 | Вызовы пользовательских скриптов..... | 10 |
| 3.7 | Контроль уникальности подстановок..... | 11 |
| 4 | Разработка и настройка заданий маскирования | 11 |
| 4.1 | Организация процесса маскирования..... | 11 |
| 4.2 | Запуск конфигурационной программы «DsMask» | 12 |
| 4.3 | Необходимые настройки проекта IBM DataStage..... | 14 |
| 4.4 | Режимы маскирования и структура заданий..... | 16 |
| 4.4.1 | Задания для потокового режима маскирования | 17 |
| 4.4.2 | Задания для маскирования «на месте» | 19 |
| 4.4.3 | Настройки оператора маскирования «DsMask»..... | 23 |
| 4.4.4 | Настройки подключений к источникам и получателям данных..... | 25 |
| 4.5 | Ручной запуск заданий маскирования | 27 |
| 4.6 | Пакетное маскирование набора таблиц базы данных | 28 |
| 5 | Разработка и настройка заданий генерации справочников | 30 |
| 5.1 | Справочник подстановки для перемешивания исходных значений | 30 |
| 5.2 | Справочник подстановки по ключу | 33 |
| 5.3 | Генерация справочника подстановок для маскирования ФИО | 35 |
| 6 | Общий синтаксис правил маскирования..... | 36 |
| 6.1 | Ключи инициализации..... | 37 |
| 6.2 | Функции для обращения к встроенным алгоритмам | 37 |
| 6.3 | Функции-скрипты Groovy..... | 38 |
| 6.4 | Синтаксис правил маскирования | 38 |
| 6.5 | Контроль уникальности подстановок..... | 40 |
| 7 | Спецификация встроенных алгоритмов | 41 |
| 7.1 | Числовой хеш-код | 41 |
| 7.2 | Двоичный хеш-код..... | 42 |

| | |
|--|----|
| 7.3 Маскирование даты | 43 |
| 7.4 Хеширование с сохранением формата | 43 |
| 7.5 Подстановка по хеш-коду | 45 |
| 7.6 Подстановка по значениям атрибутов | 47 |
| 7.7 Вызов алгоритмов ODPP | 48 |
| 7.8 Строковые преобразования | 48 |
| 7.9 Конкатенация строк | 49 |
| 7.10 Разделение строк | 50 |
| 7.11 Подстановка символов | 50 |
| 7.12 Проекция..... | 52 |
| 8 Первоначальная установка прототипа решения | 52 |
| 8.1 Установка серверных компонентов IBM Information Server | 52 |
| 8.2 Установка Windows-клиента IBM Information Server | 53 |
| 8.3 Корректировка настроек операционной системы | 53 |
| 8.4 Корректировка настроек серверных компонентов IBM Information Server | 54 |
| 8.4.1 Настройки IBM Db2 | 54 |
| 8.4.2 Настройки IBM DataStage | 54 |
| 8.4.3 Включение операционного мониторинга IBM DataStage..... | 55 |
| 8.5 Донастройка IBM Information Analyzer | 55 |
| 8.6 Настройка подавления предупреждений Information Analyzer (опционально) | 56 |
| 8.7 Установка необходимых драйверов СУБД..... | 57 |
| 8.8 Настройка проектов IBM DataStage | 58 |
| 8.9 Установка дополнительных компонентов решения | 60 |
| 8.9.1 Распаковка архивных файлов | 60 |
| 8.9.2 Регистрация примеров определения классов данных | 61 |
| 8.9.3 Регистрация оператора маскирования | 62 |
| 8.9.4 Настройка сервиса контроля уникальности | 62 |
| 8.9.5 Проверка работоспособности..... | 63 |
| 8.10 Запуск и остановка компонентов IBM Information Server | 64 |
| 8.10.1 Общие сведения о порядке запуска и остановки компонентов | 64 |
| 8.10.2 Отключение автоматического запуска компонентов (опционально) | 64 |

1 Общее описание прототипа решения

Прототип решения по маскированию данных на платформе IBM Information Server представляет собой пример частичной реализации задачи высокопроизводительного статического маскирования данных на основе правил маскирования, заданных в отношении видов конфиденциальной информации.

Прототип решения исключает необходимость в настройке конкретных операций маскирования для отдельных таблиц и полей, вместо этого осуществляя автоматический подбор правил маскирования исходя из присвоенных полям классов данных.

Организация процесса маскирования включает в себя согласованные действия нескольких видов пользователей системы:

1. Аналитики осуществляют профилирование данных (определение видов хранимой конфиденциальной информации) вручную либо с использованием автоматизированного инструмента профилирования, и публикуют результаты профилирования в единый репозиторий метаданных.
2. Инженеры по маскированию настраивают и отлаживают правила маскирования в соответствии с действующими требованиями, указывая классы маскируемых данных, алгоритмы маскирования и параметры алгоритмов.
3. Специалисты по безопасности контролируют результаты и процедуры маскирования, в том числе проводя приёмку правил маскирования (проверяя отсутствие конфиденциальной информации в замаскированных копиях данных).
4. Потребители замаскированных копий данных получают к ним доступ в результате регулярных или инициируемых по запросу запусков процесса маскирования.

Прототип решения по маскированию данных включает в себя следующие компоненты:

- Information Governance Catalog (IGC) – составная часть IBM Information Server, обеспечивающая ведение описаний классов данных (применительно к видам конфиденциальной информации), связанных бизнес-терминов и категорий (при необходимости), а также описаний баз данных, схем, таблиц и полей.
- Information Analyzer (IA) – составная часть IBM Information Server, обеспечивающая автоматизированный поиск конфиденциальной информации в массивах данных (в первую очередь в реляционных базах данных), с поддержкой сопоставления колонок с классами данных конфиденциальной информации.
- DataStage (DS) – составная часть IBM Information Server, реализующая платформу масштабируемой массивно-параллельной обработки данных и включающая в себя коннекторы для различных типов баз данных.
- Примеры дополнительных классов данных, расширяющие поставляемый с Information Server набор стандартных классов данных типовыми видами конфиденциальной информации для Российской Федерации.
- Конфигурационная программа «DsMask» – программное средство, формирующее профили маскирования для исходных таблиц с указанием конкретных операций над полями на основе настроенных правил маскирования и присвоенных классов данных полей.

- Оператор маскирования «DsMask» – программное расширение IBM DataStage, обеспечивающее непосредственную замену исходных значений колонок таблиц на замаскированные значения в рамках заданий IBM DataStage.
- Примеры правил маскирования, демонстрирующие использование алгоритмов маскирования для основных видов конфиденциальной информации.
- Примеры заданий DataStage для маскирования данных и для формирования справочников.

Маскирование может осуществляться в одном из двух режимов: потоковом режиме и режиме маскирования на месте.

При маскировании в потоковом режиме чтение исходной информации осуществляется из одной копии маскируемой БД, а результаты маскирования записываются в другую копию.

При маскировании на месте используется одна БД, при этом исходные данные читаются из этой БД, результаты маскирования временно сохраняются в сжатых файлах на сервере маскирования, а затем производится замена исходных данных на замаскированные путём очистки маскируемой таблицы и загрузки в неё замаскированных данных.

Правила маскирования могут применяться как к отдельным полям (например, номер банковской карты), так и к группам полей (например, ФИО в виде трёх отдельных полей), и могут использовать в своей логике информацию из не-маскируемых (не-конфиденциальных) полей (например, использование признака пола при маскировании ФИО).

Правила маскирования строятся на основе вызовов алгоритмов маскирования и устанавливают:

- виды конфиденциальной информации (классы данных), маскируемые с использованием конкретного правила;
- последовательность вызова алгоритмов маскирования в виде шагов соответствующего правила;
- параметры вызова алгоритмов маскирования (для настраиваемых алгоритмов);
- порядок передачи значений между шагами правила маскирования.

Такая организация правил маскирования позволяет осуществлять гибкую настройку логики правил без необходимости выполнять большой объём программирования.

2 Характеристика компонентов прототипа решения

2.1 Схема прототипа решения по маскированию данных

Общая схема прототипа решения по маскированию приведена ниже на рисунке 1 ниже.

Синим цветом показаны элементы платформы IBM Information Server, жёлтым – дополнительные компоненты в составе прототипа решения, зелёным – информационные объекты, дорабатываемые или разрабатываемые при внедрении решения по маскированию, оранжевым – источники и получатели данных.



Рисунок 1. Общая схема прототипа решения по маскированию данных

2.2 IBM Information Governance Catalog

IBM Information Governance Catalog – это инструмент для углублённой работы с метаданными (бизнес, техническими и операционными): ведения бизнес-гlossария предприятия, описания всех информационных активов компании, управления взаимосвязями между различными типами метаданными, визуализации полных путей обработки данных от источника вплоть до полей отчётов и др.

В рамках решения по маскированию данных IGC обеспечивает ведение перечня видов конфиденциальной информации в виде специальных информационных объектов – классов данных. К каждому классу данных приписаны правила поиска (обнаружения) соответствующей этому классу информации, которые используются при профилировании данных в инструменте IBM Information Analyzer.

IGC также обеспечивает ведение описаний баз данных, таблиц и полей, и может применяться как единый репозиторий соответствующих метаданных при работе команд внедрения и сопровождения решения по маскированию.

К каждому полю может быть приписан класс данных, который характеризует хранимую в этом поле информацию. Решение по маскированию использует хранимые в IGC сведения о приписанных полям таблиц классах конфиденциальной информации для применения к этим полям соответствующих правил маскирования.

2.3 IBM Information Analyzer

Инструмент IBM Information Analyzer предоставляет возможности для профилирования данных, уменьшающие затраты и риски проекта посредством оперативного выявления проблем с качеством данных и мониторинга изменений в структуре данных и в контенте.

Методы профилирования данных концентрируются на контенте данных: они исследуют значения в каждом столбце, извлекают дополнительную информацию из этих значений, а затем используют эту информацию для оценки структурной целостности источников данных и отношений между ними.

В рамках решения по маскированию данных IA обеспечивает поиск скрытой конфиденциальной информации при отсутствии точной априорной информации о её размещении. Результаты работы IA предоставляются пользователю-аналитику, который может проконтролировать правильность выбора классов данных для колонок таблиц, а затем опубликовать уточнённые результаты анализа в централизованном репозитории метаданных, поддерживаемом инструментом IBM Information Governance Catalog.

2.4 IBM DataStage

IBM DataStage представляет собой гибкую, функциональную и масштабируемую платформу интеграции данных. Обеспечивается как вертикальное (в рамках одного сервера), так и горизонтальное (в рамках кластера серверов) масштабирование, с обеспечением практически линейного роста производительности при увеличении доступных вычислительных ресурсов.

В составе IBM DataStage предусмотрены коннекторы к практически всем видам источников данных, причём для большинства промышленных СУБД предусмотрены оптимизированные коннекторы, реализующие высокоскоростную загрузку и выгрузку данных.

2.5 Конфигурационная программа «DsMask»

Конфигурационная программа «DsMask» принимает на вход комплект правил маскирования, а также сведения о соответствии классов конфиденциальной информации колонкам таблиц, зафиксированные по результатам профилирования данных в инструменте IA и опубликованные в репозитории метаданных IGC.

Результатом работы конфигурационной программы являются профили маскирования таблиц, в каждом из которых указан перечень действий (фактически, специальная мини-программа) по маскированию колонок соответствующей таблицы.

Профили маскирования сохраняются в конфигурационную базу данных формата H2 (<http://h2database.com/>), которая в дальнейшем используется оператором маскирования «DsMask». Идентификация профилей маскирования осуществляется исходя из зафиксированного в репозитории метаданных полного логического имени таблицы, включая имя базы данных, имя схемы и имя таблицы.

Конфигурационная программа осуществляет базовые проверки корректности правил маскирования, включая проверку синтаксиса скриптов на языке Groovy и проверку наличия необходимых атрибутов в элементах правил маскирования. Выводимый по итогам работы конфигурационной программы текстовый отчёт включает в себя информацию о составе сформированных профилей маскирования и о выполняемых в рамках каждого из этих профилей операциях маскирования.

2.6 Оператор маскирования «DsMask»

Оператор маскирования «DsMask» представляет собой реализацию логики по маскированию табличных (структурированных) данных в виде узла обработки «Java Integration Stage» для среды выполнения IBM DataStage.

При настройке оператора маскирования должен быть указан профиль маскирования (логическое имя таблицы, включая имя базы данных, схемы и собственно таблицы), который и определяет состав обрабатываемых колонок таблицы и применяемые алгоритмы маскирования. Реальные (физические) имена входных и выходных таблиц на работу оператора маскирования никак не влияют.

Профиль маскирования загружается из конфигурационной базы данных в формате H2, подготовленной с помощью конфигурационной программы «DsMask». Конфигурационная база данных хранит произвольное количество профилей маскирования, при этом оператор маскирования загружает для своей работы один (указанный) профиль.

При работе оператора маскирования осуществляется замена значений колонок, входящих в профиль маскирования, замаскированными значениями, вычисленными на основе установленных профилем маскирования алгоритмов маскирования. Значения колонок обрабатываемой таблицы, не входящие в профиль маскирования, остаются неизменными.

Оператор маскирования взаимодействует со вспомогательным сервисом контроля уникальности, который обеспечивает обнаружение коллизий при маскировании на основе алгоритмов с расчётом хеш-кодов. Сервис контроля уникальности ведёт таблицу соответствия замаскированных и исходных значений, что и позволяет обнаружить ситуации получения идентичных результатов маскирования для разных исходных значений.

2.7 Примеры дополнительных классов конфиденциальных данных

В составе прототипа решения поставляется набор примеров определений классов конфиденциальных данных в следующем составе:

1. ОГРН для юридических лиц и индивидуальных предпринимателей;
2. номер паспорта гражданина Российской Федерации;
3. номер заграничного паспорта гражданина Российской Федерации;
4. наименование организации, выдавшей документ, удостоверяющий личность гражданина Российской Федерации;
5. СНИЛС;
6. ИНН для физического или юридического лица;
7. фамилия, имя и отчество на русском языке (целиком либо отдельными полями);
8. почтовый адрес;
9. название профессии.

Для идентификации таких видов конфиденциальных данных, как ФИО, почтовый адрес и название профессии, используются справочники, поставляемые в виде CSV-файлов в кодировке UTF-8. При необходимости содержимое используемых справочников может быть дополнено или скорректировано в соответствии с реальным наполнением соответствующих полей источников данных.

3 Встроенные алгоритмы маскирования

Прототип решения включает в себя встроенные алгоритмы маскирования следующих видов:

1. Хеширование без сохранения формата исходного значения.
2. Хеширование с сохранением формата исходного значения.
3. Подстановка значений на основе хеш-кода или исходного значения.
4. Вспомогательные алгоритмы подготовки данных.
5. Вызов алгоритмов Optim Data Privacy Providers Library (ODPP).
6. Вызовы пользовательских скриптов.
7. Контроль уникальности подстановок.

3.1 Хеширование без сохранения формата

Алгоритмы хеширования без сохранения формата используются для согласованной необратимой замены исходных значений и включают:

- расчёт числового хеш-кода;
- расчёт двоичного хеш-кода.

При настройке алгоритмов может опционально указываться ключ инициализации, замена значения которого позволяет изменить результаты работы алгоритмов.

Исходные данные для расчёта хеш-кодов интерпретируются как строки и преобразуются в двоичные последовательности в кодировке UTF-8. Если исходные данные представляют собой двоичную последовательность (массив октетов), то она используется для расчёта хеш-кода как есть, и предварительное преобразование в строку не производится.

Числовой хеш-код вычисляется как контрольная сумма CRC-32. Результат расчёта числового хеш-кода – 64-битное знаковое целое число в диапазоне от 0 до $2^{32}-1$.

Двоичный хеш-код вычисляется в соответствии с одним из алгоритмов Message Digest, поддерживаемых IBM JDK (см. перечень в документе: <https://www.ibm.com/docs/en/sdk-java-technology/8?topic=jcaasr-appendix-standard-names>).

Результат расчёта двоичного хеш-кода возвращается как в виде строки в шестнадцатеричной системе, так и в виде двоичной последовательности (массива октетов).

3.2 Хеширование с сохранением формата

Алгоритмы хеширования с сохранением формата используются для согласованной необратимой замены исходных значений без изменения формата данных, и включают:

- общий алгоритм хеширования с сохранением формата (Format Preserving Encryption, FPE);
- специализированный алгоритм для маскирования дат.

При настройке алгоритмов может опционально указываться ключ инициализации, замена значения которого позволяет изменить результаты работы алгоритмов.

Алгоритм FPE вычисляет код аутентификации сообщения (HMAC-512) над исходным значением, преобразованным в строковый формат в кодировке UTF-8. Двоичное значение кода аутентификации используется для вычисления позиции подстановочного символа каждого из символов исходной строки, при этом состав подстановочных символов зависит от класса

символов (например, строчные и прописные буквы, числа и др.). Состав обрабатываемых классов символов является настраиваемым и задаётся специальным файлом формата XML. Символы, не входящие в один из обрабатываемых классов, остаются неизменными. Поддерживается работа с произвольными символами Unicode.

Специализированный алгоритм для маскирования дат вычисляет числовой хеш-код над значением даты, и использует его для вычисления результата маскирования. Стандартный вариант алгоритма использует полученный хеш-код для вычисления номера дня в году, в результате номер года остаётся неизменным, а сама дата – заменяется на другую в рамках этого года.

3.3 Алгоритмы подстановки и перемешивание исходных значений

Алгоритмы подстановки значений используются для согласованной замены текстовых данных, включая имена, названия и адресную информацию, и включают:

- подстановку по хеш-коду;
- подстановку по значениям атрибутов.

Алгоритмы подстановки значений используют таблицы подстановок, сохранённые в формате баз данных H2. Структура таблиц подстановок зависит от требуемой логики маскирования. Генерация таблиц подстановки осуществляется с использованием заданий DataStage или специальной утилитой, реализующей формирование таблицы подстановки ФИО на основе справочника типичных имён и фамилий.

Подстановка по хеш-коду осуществляет поиск в таблице подстановок, в которой должно присутствовать числовое поле позиции подстановки (в диапазоне значений от 0 до N-1, где N – размер таблицы), а также одно или несколько полей, используемых для замещения исходных значений. Над полем позиции подстановки должен присутствовать индекс (например, это поле может быть объявлено первичным ключом таблицы).

Значение позиции подстановки рассчитывается алгоритмом как числовой хеш-код CRC-32 над одним или несколькими исходными значениями, с последующим вычислением остатка от деления полученного значения хеш-кода на размер таблицы подстановок.

Для обеспечения возможности изменения результатов маскирования предусмотрен опциональный параметр ключа инициализации.

Подстановка по значениям атрибутов осуществляет поиск в таблице подстановок, указанной в настройках алгоритма, записей, для которых значения «ключевых» полей совпадают со значениями атрибутов исходных (маскируемых) данных. Над сочетанием полей таблицы подстановок, используемых при поиске, должен присутствовать индекс. Результатом работы алгоритма являются значения полей найденной подстановочной записи, которые могут быть использованы для замены исходных значений.

Операция перемешивания исходных значений может быть реализована на основе алгоритма подстановки по значениям атрибутов. Для этого предварительно формируется таблица подстановок, содержащая для каждого исходного значения его случайно выбранное

подстановочное значение из набора значений, используемых в системе-источнике. Такая реализация перемешивания позволяет получить согласованные результаты маскирования для множества полей, таблиц и баз данных. Пример задания DataStage для подготовки таблицы подстановок, реализующей перемешивание числовых идентификаторов, включён в состав материалов решения.

3.4 Вызов алгоритмов Optim Data Privacy Providers Library

В составе решения предусмотрена возможность встраивания в правила маскирования алгоритмов, входящих в состав библиотеки IBM InfoSphere Optim Data Privacy Providers (ODPP). Использование этих алгоритмов требует наличия установленных библиотек ODPP, которые должны быть соответствующим образом лицензированы (не являются частью платформы IBM Information Server).

Состав алгоритмов ODPP, описание их конфигурационных параметров и примеры вызова приведены в документации на ODPP: <https://www.ibm.com/docs/en/iotdm/11.3?topic=reference-optim-data-privacy-provider-library>

3.5 Вспомогательные алгоритмы

Вспомогательные алгоритмы предназначены для нормализации маскируемых значений и компенсации различий в представлении данных между разными системами, что необходимо для получения согласованных результатов маскирования.

Вспомогательные алгоритмы включают:

- конкатенацию строк с использованием заданного разделителя;
- разделение строк на части с использованием регулярного выражения;
- подстановку символов на основе настраиваемой таблицы подстановок;
- проекцию атрибутов, т.е. выбор нужных полей из предыдущих шагов как конечный результат правила маскирования;
- элементарные строковые преобразования, включая перевод в верхний, нижний и смешанный регистр, удаление конечных и/или начальных пробелов;
- замену частей строки на основе заданных регулярных выражений.

Специальным случаем вспомогательных алгоритмов является поддержка выражений-предикатов, позволяющих пропустить некоторые шаги правила маскирования на основе заданных пользователем условий. Пропуск «лишних» в некоторых ситуациях шагов правила позволяет увеличить производительность маскирования за счёт исключения выполнения ненужных операций.

3.6 Вызовы пользовательских скриптов

Пользовательские скрипты на языке Groovy позволяют реализовать в составе правила маскирования логику, выходящую за рамки стандартных алгоритмов. В то же время при подготовке пользовательских скриптов и использующих их правил маскирования можно опираться на возможности стандартных алгоритмов.

Например, пользовательские скрипты могут использоваться для расчёта контрольных разрядов при маскировании значений ИНН в формате, применяемом в Российской Федерации. При этом основным алгоритмом маскирования ИНН может быть алгоритм хеширования с сохранением формата, дополненный пользовательской логикой по расчёту контрольных разрядов.

3.7 Контроль уникальности подстановок

При всём удобстве и эффективности алгоритмов маскирования, основанных на расчёте хеш-кодов, результат маскирования с использованием этих алгоритмов не даёт гарантий от возникновения коллизий, т.е. ситуаций наличия идентичных замаскированных значений для нескольких разных исходных значений. Такие коллизии возникают на практике при работе с выборками данных большого объёма, от нескольких миллионов значений. В тех ситуациях, когда коллизии при маскировании недопустимы, можно воспользоваться встроенным в решение алгоритмом контроля уникальности подстановок.

Контроль уникальности подстановок поддерживается для всех алгоритмов, основанных на расчёте хеш-кода, и обеспечивается за счёт ведения внутренней таблицы соответствия исходных и замаскированных значений на стороне сервера маскирования. При обнаружении коллизии выполняется повторный расчёт замаскированного значения модифицированной версией алгоритма, которая добавляет к данным для расчёта хеш-кода номер итерации. Повторные расчёты с увеличивающимся номером итерации выполняются, пока не будет получено уникальное значение, либо до достижения предельного количества итераций (в этом случае сигнализируется ошибка маскирования).

Применение контроля уникальности подстановок эффективно в случаях, когда оценка количества возможных допустимых замаскированных значений как минимум в 10 раз превосходит количество различных маскируемых исходных значений. Если количество допустимых замаскированных значений недостаточно велико, то увеличивается количество коллизий и возникают лишние итерации расчёта. В результате возможны ошибки при выполнении маскирования из-за невозможности сформировать уникальное значение за заданное предельное количество итераций. В таких ситуациях следует использовать другие методы маскирования, например, перемешивание исходных значений.

4 Разработка и настройка заданий маскирования

4.1 Организация процесса маскирования

Перед выполнением процесса маскирования должны быть выполнены следующие подготовительные работы:

1. Сформирован набор классов конфиденциальной информации в виде информационных объектов типа «Класс данных» в инструменте Information Governance Catalog.
2. Выполнен импорт структуры таблиц из баз данных систем-источников (или их копий) средствами инструмента Metadata Asset Manager.
3. Выполнена разметка конфиденциальных полей таблиц систем-источников соответствующими классами данных. Разметка может быть выполнена вручную, прямым

редактированием описаний в Information Governance Catalog, либо автоматизировано, с использованием средств анализа данных инструмента Information Analyzer.

4. Разработан комплект правил маскирования, определяющего операции по преобразованию значений конфиденциальных классов с использованием доступных алгоритмов маскирования.
5. Подготовлены необходимые справочники подстановок, используемые при маскировании имён, наименований и/или адресной информации (см. раздел 5 *Разработка и настройка заданий генерации справочников*).

В процессе сопровождения системы маскирования данных перечисленные выше виды подготовительных работ могут неоднократно повторяться для внесения изменений в состав маскируемых данных и логику маскирования.

Процесс маскирования включает в себя следующие этапы:

1. Запуск конфигурационной программы «DsMask» для сопоставления операций маскирования с конкретными таблицами и полями, и формирования профилей маскирования (мини-программ для маскирования данных по каждой таблице).
2. Для каждой таблицы – запуск задания DataStage, осуществляющего чтение данных из исходной базы данных, маскирование данных и запись результатов маскирования в целевую базу данных. Выполнение заданий маскирования осуществляется в параллельном режиме с использованием доступных вычислительных мощностей.
3. Контроль результатов маскирования, путём проверки статусов выполнения заданий маскирования на отсутствие ошибок и предупреждений при их выполнении.

Порядок запуска конфигурационной программы «DsMask», структура используемых заданий IBM DataStage, порядок запуска заданий и контроля результатов их выполнения описаны в последующих подразделах.

4.2 Запуск конфигурационной программы «DsMask»

Конфигурационная программа «DsMask» осуществляет формирование профилей маскирования на основе классов данных, присвоенных полям таблиц систем-источников, а также подготовленного набора правил маскирования. Сформированные профили маскирования сохраняются в конфигурационную базу данных формата H2 и представляют собой перечень операций маскирования, которые должны быть выполнены для каждой таблицы.

Настройка конфигурационной программы «DsMask» осуществляется путём создания конфигурационного файла в формате XML. Пример конфигурационного файла входит в состав конфигурационной программы «DsMask» и поставляется в виде файла «jconf-job-default.xml».

Описание параметров конфигурационной программы «DsMask» приведено в таблице ниже.

| Параметр | Описание |
|-------------|---|
| in.tab.type | Используемые источники данных о структуре таблиц, перечисление значений через запятые: <ul style="list-style-type: none">• igc – IBM Information Governance Catalog;• file – файл в формате XML с описанием структуры таблиц |

| Параметр | Описание |
|---------------------|--|
| in.tab.file.name | Имя файла в формате XML с описанием структуры таблиц. Описание формата файла приведено ниже |
| in.tab.igc.url | JDBC URL для подключения к базе данных XMETA в составе IBM Information Governance Catalog, типовое значение: jdbc:db2://localhost:50000/xmeta |
| in.tab.igc.username | Логин пользователя для подключения к базе данных XMETA, типовое значение: xmeta |
| in.tab.igc.password | Пароль пользователя для подключения к базе данных XMETA |
| in.dc.rules | Путь к файлу со списком конфиденциальных классов данных |
| in.masking.rules | Путь к каталогу с файлами правил маскирования в формате XML |
| in.masking.context | Имя контекста маскирования, обычно используется пустое значение |
| out.dir | Путь к каталогу для хранения конфигурационных баз данных |
| out.config | Имя создаваемой конфигурационной базы данных без указания пути |
| out.dump | Базовый путь к файлам диагностических дампов. При отсутствии параметра диагностические дампы не создаются |

Пример файла настроек конфигурационной программы «DsMask» приведён ниже.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<comment>Sample job file for dsmask-jconf program</comment>
<entry key="in.tab.type">igc,file</entry>
<entry key="in.tab.file.name">extra-tables.xml</entry>
<entry key="in.tab.igc.url">jdbc:db2://localhost:50000/xmeta</entry>
<entry key="in.tab.igc.username">db2inst1</entry>
<entry key="in.tab.igc.password">P@$$w0rd</entry>
<entry key="in.dc.rules">confidential-dataclass.xml</entry>
<entry key="in.masking.rules">masking-rules</entry>
<entry key="in.masking.context"></entry>
<entry key="out.dir">config</entry>
<entry key="out.config">default</entry>
<entry key="out.dump">dump-default</entry>
</properties>
```

Запуск конфигурационной программы «DsMask» осуществляется следующими командами:

```
cd КаталогУстановкиDsMask/dsmask-jconf
./BuildConf.sh ФайлНастроек.xml
```

Запуск конфигурационной программы «DsMask» для обновления профиля маскирования может быть включён в состав подготовительных операций по пакетному маскированию определённой базы данных.

Пример протокола работы конфигурационной программы «DsMask»:

```
2020-10-15 17:44:25 INFO BuildConf:83 - DsMask configurator started.
2020-10-15 17:44:26 INFO BuildConf:86 - Table information loaded, total 7 tables.
2020-10-15 17:44:26 INFO BuildConf:89 - Data classes loaded, total 14 classes.
2020-10-15 17:44:27 INFO BuildConf:92 - Masking rules loaded, total 17 rules, 36 functions, 1 keys.
2020-10-15 17:44:30 INFO ScriptChecker:73 - Tested 21 script functions, with total 0 failures.
2020-10-15 17:44:30 INFO BuildConf:103 - Masking context: [].
2020-10-15 17:44:30 INFO BuildConf:112 - Configuration database opened.
2020-10-15 17:44:30 INFO BuildConf:115 - Keys saved.
2020-10-15 17:44:30 INFO BuildConf:122 - Table db1.s1.t1 processed, total 1 masking operations
2020-10-15 17:44:30 INFO BuildConf:187 -         Operation: rule vu-single
2020-10-15 17:44:30 INFO BuildConf:189 -         INPUT    id
2020-10-15 17:44:30 INFO BuildConf:192 -         OUTPUT   id
2020-10-15 17:44:30 INFO BuildConf:122 - Table db1.s1.t2 processed, total 2 masking operations
2020-10-15 17:44:30 INFO BuildConf:187 -         Operation: rule name-full-initials
2020-10-15 17:44:30 INFO BuildConf:189 -         INPUT    fullname
2020-10-15 17:44:30 INFO BuildConf:189 -         INPUT    shortname
2020-10-15 17:44:30 INFO BuildConf:192 -         OUTPUT   shortname
2020-10-15 17:44:30 INFO BuildConf:187 -         Operation: rule name-full
2020-10-15 17:44:30 INFO BuildConf:189 -         INPUT    fullname
2020-10-15 17:44:30 INFO BuildConf:192 -         OUTPUT   fullname
2020-10-15 17:44:30 INFO BuildConf:130 - Table db1.s1.t3 skipped - no matching rules
2020-10-15 17:44:30 INFO BuildConf:135 - DsMask configurator complete, database closed.
2020-10-15 17:44:30 INFO BuildConf:204 - Table structure dump has been written to dump.tables.xml
2020-10-15 17:44:30 INFO BuildConf:214 - Confidential data classes have been written to dump.dcs.txt
```

При обнаружении синтаксических ошибок в правилах маскирования и включённых в них скриптах на языке Groovy конфигурационная программа «DsMask» выводит в протокол диагностическую информацию о выявленных ошибках. Для формирования профилей маскирования необходимо исправить ошибки и повторно запустить конфигурационную программу.

4.3 Необходимые настройки проекта IBM DataStage

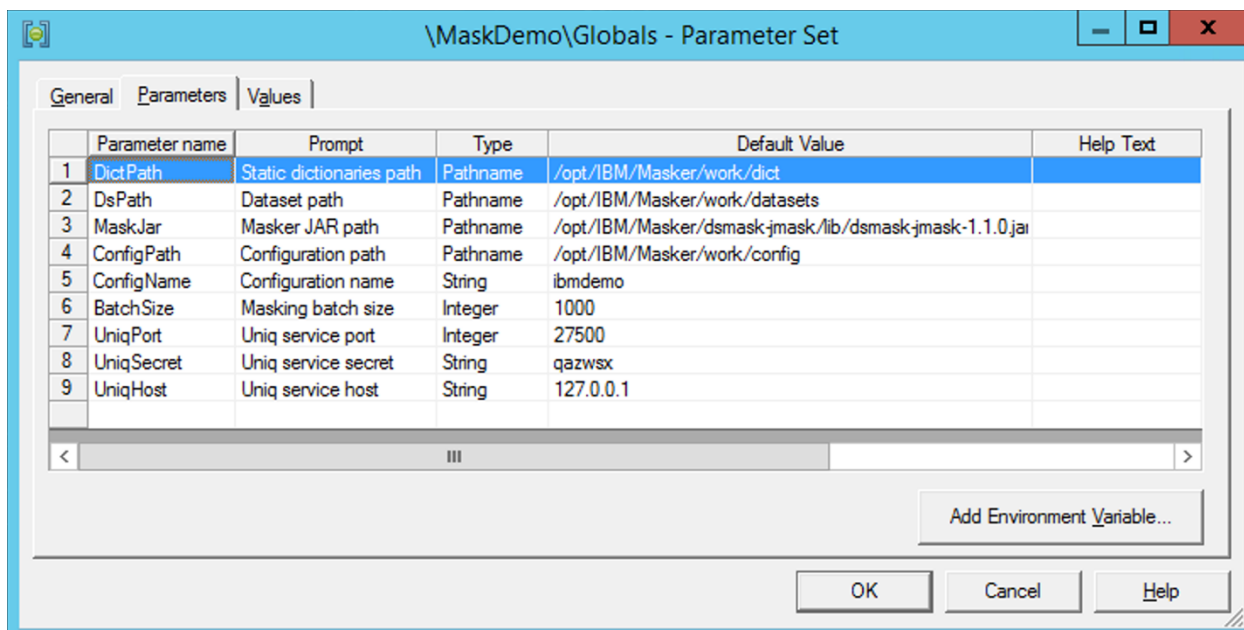
Перед подготовкой заданий маскирования необходимо выполнить ряд настроек используемого проекта IBM DataStage, которые описаны в разделе «8.8 Настройка проектов IBM DataStage».

С целью упрощения управления заданиями маскирования рекомендуется использование поддерживаемого DataStage механизма группировки параметров в объекты ParameterSet.

Типовая конфигурация для маскирования данных включает в себя, как минимум, три основных объекта ParameterSet:

- глобальные настройки («Globals»): обычно существует в единственном экземпляре и задаёт основные константы для инсталляции решения по маскированию;
- подключения к источникам («DbParams»): по одному набору параметров на каждую базу данных, являющуюся источником;
- подключения к получателям («DbOutParams»): по одному набору параметров на каждую базу данных, являющуюся получателем.

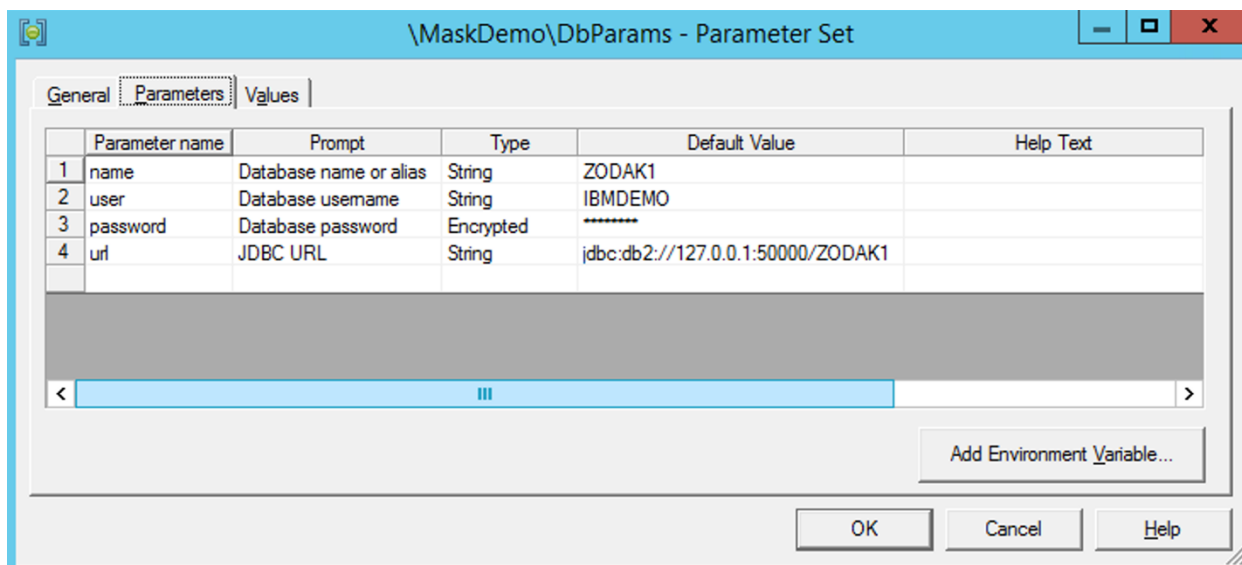
Типичная структура глобальных настроек, реализованных в виде набора параметров «Globals», показана на рисунке ниже.



Описание параметров в наборе «Globals» приведено в таблице ниже.

| Параметр | Пример значения | Описание |
|------------|---|---|
| DsPath | /opt/IBM/Masker/work/datasets | Путь к каталогу для сохранения наборов данных DataStage |
| DictPath | /opt/IBM/Masker/work/dict | Путь к каталогу со справочниками |
| ConfigPath | /opt/IBM/Masker/work/config | Путь к каталогу с базами данных конфигураций маскирования |
| ConfigName | default | Логическое имя используемой конфигурационной базы данных маскирования |
| MaskJar | /opt/IBM/Masker/dsmask-jmask/lib/dsmask-jmask.jar | Путь к библиотеке реализации оператора маскирования «DsMask» |
| BatchSize | 1000 | Количество строк, маскируемых за одну пакетную операцию |
| UniqHost | localhost | Сетевое имя либо сетевой адрес сервера, на котором функционирует сервис контроля уникальности |
| UniqPort | 27500 | Номер сетевого порта сервиса контроля уникальности |
| UniqSecret | ***** | Код доступа к сервису контроля уникальности |

Типичная структура набора параметров «DbParams» показана на рисунке ниже.



Описание параметров в наборе «DbParams» приведено в таблице ниже.

| Параметр | Пример значения | Описание |
|----------|--|---|
| name | dbname | Имя базы данных, используемое при обращении (для Oracle – имя из tnsnames.ora, для Db2 – локальный псевдоним, и т.п.) |
| user | maskuser | Логин для подключения к БД |
| password | ***** | Пароль для подключения к БД |
| url | jdbc:sqlserver://192.168.25.101:1433; databaseName=dbname | JDBC URL для подключения к БД |

Типичная структура набора параметров «DbOutParams» соответствует структуре набора параметров «DbParams», поэтому её описание здесь не дублируется.

4.4 Режимы маскирования и структура заданий

Маскирование может осуществляться в одном из двух режимов, и выбор режима маскирования определяет используемый дизайн заданий:

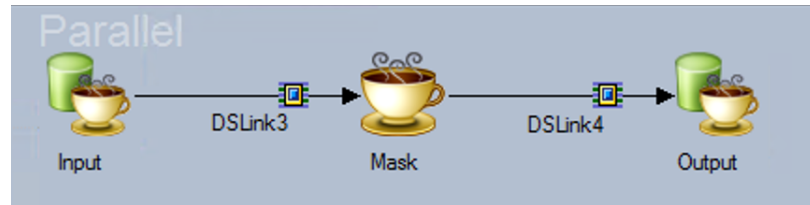
- маскирование в потоковом режиме – записи читаются из исходной таблицы в одной копии базы данных и записываются (заменяют содержимое) такой же таблицы в другой копии базы данных;
- маскирование «на месте» – записи читаются из исходной таблицы, маскируются и записываются в промежуточный набор данных на стороне сервера DataStage (набор данных обычно сжимается). Затем содержимое исходной таблицы заменяется на замаскированные данные.

В обоих режимах используются универсальные задания маскирования, не зависящие от структуры маскируемых таблиц. Обработка произвольных структур таблиц обеспечивается применением механизма Runtime Column Propagation, который должен быть включён на связях между этапами заданий.

4.4.1 Задания для потокового режима маскирования

При реализации потокового режима маскирования достаточно создать по одному заданию на каждый тип коннекторов с СУБД-источниками.

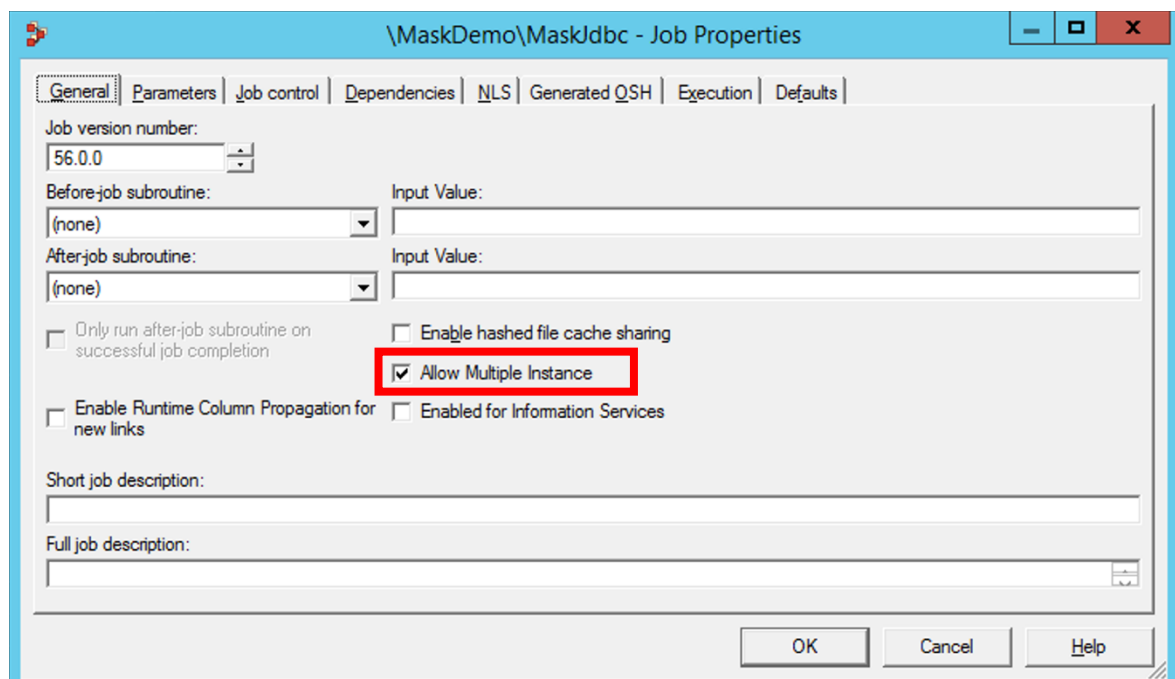
Пример дизайна задания маскирования для потокового режима приведён на рисунке ниже.



Задание состоит из трёх шагов:

1. Чтение исходных данных из системы-источника. Используется коннектор для СУБД источника, либо обобщённый JDBC или ODBC коннектор.
2. Маскирование данных. Используется Java Integration Stage, в котором настроен вызов оператора маскирования «DsMask».
3. Запись замаскированных данных в систему-получатель. Используемый коннектор обычно совпадает с шагом 1, параметры соединения соответствуют БД получателя.

Настройки задания должны разрешать параллельное выполнение нескольких его копий, как показано на рисунке ниже.



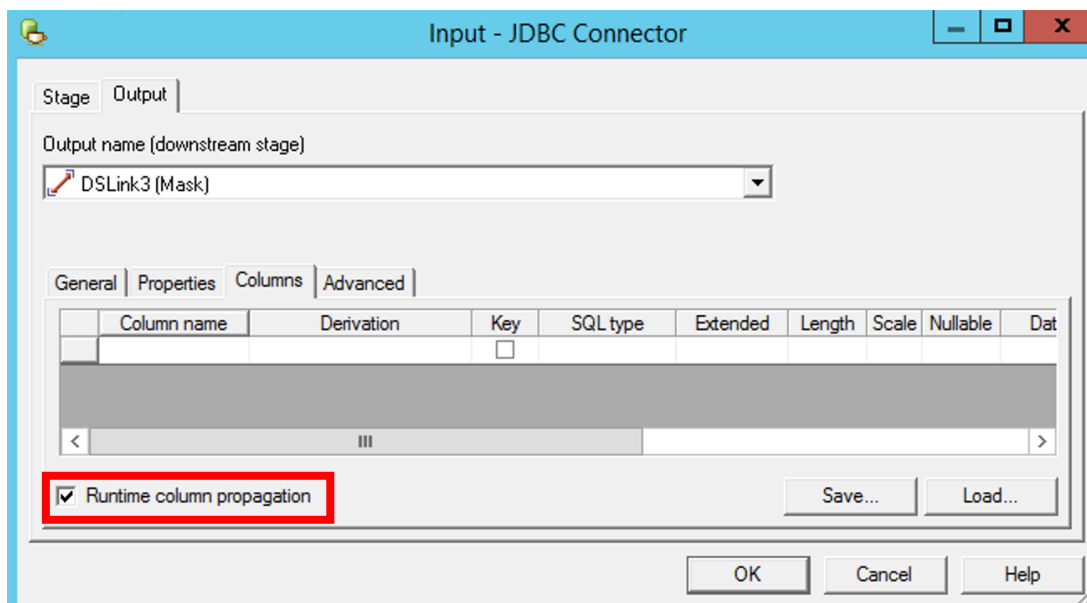
Типовой состав параметров задания маскирования в потоковом режиме показан на рисунке ниже.

| | Parameter name | Prompt | Type | Default Value | Help Text |
|---|-------------------|----------------------|---------------|----------------------|--------------------------------------|
| 1 | DbOutParams | DbOutParams paramet | Parameter Set | (As pre-defined) | Database connection parameters |
| 2 | \$APT_CONFIG_FILE | Configuration file | Pathname | /opt/IBM/Information | The Parallel job configuration file. |
| 3 | Globals | Globals parameters | Parameter Set | (As pre-defined) | |
| 4 | DbParams | DbParams parameter | Parameter Set | (As pre-defined) | Database connection parameters |
| 5 | InputTable | Input table name | String | [repl].[TAB1] | |
| 6 | MaskingProfile | Masking profile name | String | bazuka.repl.tab1 | |
| 7 | OutputTable | Output table name | String | [repl].[TAB1_COPY] | |
| 8 | BatchId | Job batch identifier | String | - | |

Примечания:

1. В задание передаются группы параметров Globals, DbParams и DbOutParams, описанные ранее.
2. Параметры InputTable и OutputTable задают имена входной и выходной таблиц.
3. Параметр MaskingProfile – логическое имя таблицы, определяющее состав выполняемых операций маскирования.
4. Переменная \$APT_CONFIG_FILE устанавливает имя файла настроек параллельного выполнения.
5. Параметр BatchId используется для группировки выполняемых заданий в пакетные операции маскирования баз данных (используется в дальнейшем для контроля успешности пакетной операции в целом).

Для коннекторов оператора чтения данных и оператора маскирования должен быть установлен режим «Runtime Column Propagation», как показано на рисунке ниже:



Информация о настройках коннекторов для источников и получателей данных, а также о настройках оператора маскирования приведена в последующих подразделах.

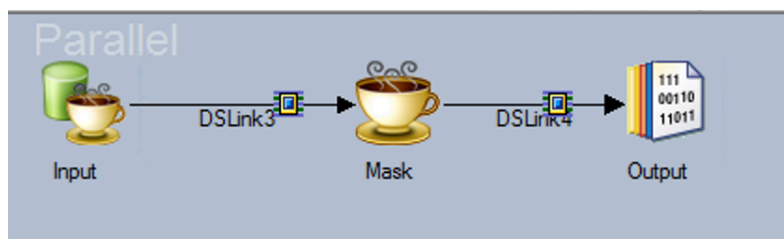
4.4.2 Задания для маскирования «на месте»

При реализации маскирования «на месте» обычно требуется по три задания на каждый тип коннекторов с СУБД-источниками:

- задание маскирования, выполняющее чтение записей из источника, маскирование и запись в локальный набор данных;
- задание загрузки, выполняющее замену содержимого таблицы источника на информацию, ранее сохранённую при маскировании в локальный набор данных;
- управляющее задание - последовательность операций вызова задания маскирования и затем задания загрузки как единой операции.

4.4.2.1 Дизайн задания маскирования «на месте»

Пример дизайна задания маскирования для варианта маскирования «на месте» приведён на рисунке ниже.



Задание состоит из трёх шагов:

1. Чтение исходных данных из системы-источника. Используется коннектор для СУБД источника, либо обобщённый JDBC или ODBC коннектор.
2. Маскирование данных. Используется Java Integration Stage, в котором настроен вызов оператора маскирования «DsMask».
3. Запись замаскированных данных в локальный набор данных.

Настройки задания должны разрешать параллельное выполнение нескольких его копий.

Типовой состав параметров задания маскирования в режиме маскирования на месте показан на рисунке ниже.

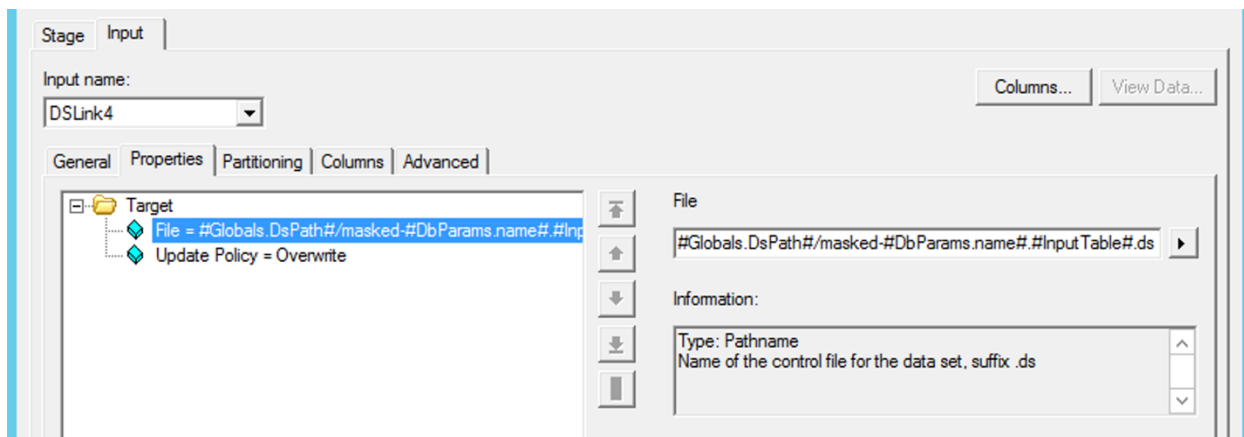
| General Parameters Job control Dependencies NLS Generated QSH Execution Defaults | | | | | |
|--|-------------------|----------------------|---------------|----------------------|--------------------------------------|
| | Parameter name | Prompt | Type | Default Value | Help Text |
| 1 | \$APT_CONFIG_FILE | Configuration file | Pathname | /opt/IBM/Information | The Parallel job configuration file. |
| 2 | Globals | Globals parameters | Parameter Set | (As pre-defined) | |
| 3 | DbParams | DbParams parameters | Parameter Set | (As pre-defined) | Database connection parameters |
| 4 | InputTable | Input table name | String | [repl].[TAB1] | |
| 5 | MaskingProfile | Masking profile name | String | bazuka.repl.tab1 | |
| 6 | BatchId | Job batch identifier | String | - | |

Примечания:

1. Передаваемые параметры аналогичны параметрам для задания потокового маскирования, за исключением отсутствия параметров, определяющих подключение к выходной базе данных и имя выходной таблицы.

Для выходных коннекторов оператора чтения данных и оператора маскирования должен быть установлен режим «Runtime Column Propagation», как показано на примере задания «потокowego» маскирования.

Пример настроек оператора записи в локальный набор данных показан на рисунке ниже.



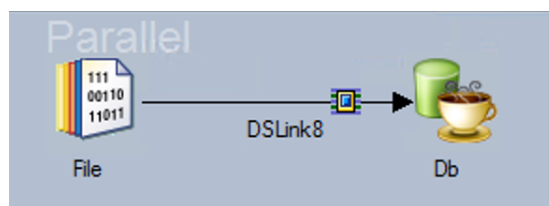
Примечания:

1. Включён режим перезаписи существующего набора данных (Update Policy = Overwrite).
2. Имя файла для размещения набора данных обычно формируется в соответствии со следующим шаблоном:
«#Globals.DsPath#/masked-#DbParams.name#. #InputTable#.ds», где:
 - Globals.DsPath – путь к каталогу размещения наборов данных из набора настроек Globals;
 - DbParams.name – логическое имя исходной базы данных;
 - InputTable – имя исходной таблицы (обычно вместе с указанием схемы).

Информация о настройках коннекторов для источников данных, а также о настройках оператора маскирования приведена в последующих подразделах.

4.4.2.2 Дизайн задания загрузки данных при маскировании на месте

Пример дизайна задания загрузки данных приведён на рисунке ниже.



Задание состоит из двух шагов:

1. Чтение замаскированных записей из локального набора данных, сформированного заданием маскирования.
2. Запись замаскированных записей в систему-получатель. Используемый коннектор обычно совпадает с коннектором шага 1 задания маскирования, параметры соединения соответствуют БД получателя.

Настройки задания должны разрешать параллельное выполнение нескольких его копий.

Типовой состав параметров задания загрузки данных показан на рисунке ниже.

| General Parameters Job control Dependencies NLS Generated QSH Execution Defaults | | | | | |
|--|-------------------|----------------------|---------------|----------------------|--------------------------------------|
| | Parameter name | Prompt | Type | Default Value | Help Text |
| 1 | DbOutParams | DbOutParams param | Parameter Set | (As pre-defined) | Database connection parameters |
| 2 | \$APT_CONFIG_FILE | Configuration file | Pathname | /opt/IBM/Information | The Parallel job configuration file. |
| 3 | Globals | Globals parameters | Parameter Set | (As pre-defined) | |
| 4 | DbParams | DbParams parameters | Parameter Set | (As pre-defined) | Database connection parameters |
| 5 | InputTable | Input table name | String | MASKDEMO.TAB1 | |
| 6 | OutputTable | Output table name | String | MASKDEMO.TAB2 | |
| 7 | BatchId | Job batch identifier | String | - | |

Примечания:

1. Передаваемые параметры аналогичны параметрам для задания потокового маскирования.
2. Параметры InputTable и DbParams используются для формирования имени файла, из которого читаются замаскированные записи. Подключение к БД-источнику и чтение из неё информации этим заданием не производится.

Для выходного коннектора оператора чтения данных должен быть установлен режим «Runtime Column Propagation», как показано на примере задания «потокового» маскирования.

Пример настроек оператора чтения из локального набора данных показан на рисунке ниже.

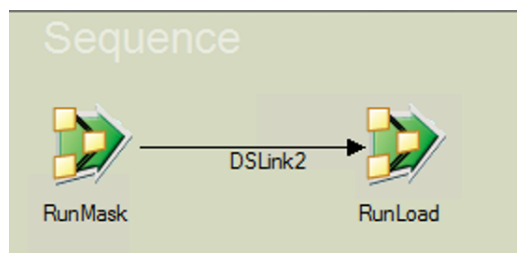
The screenshot shows the 'Stage' configuration window for an 'Output' operator. The 'Output name' is set to 'DSLink8'. The 'General' tab is selected, displaying a tree view of the configuration. Under 'Source', the 'File' property is configured with the template: `#Globals.DsPath#/masked-#DbParams.name#.#InputTable#.ds`. The 'Options' section shows 'Missing Columns Mode = Ignore'. On the right, the 'File' property is expanded, showing its 'Type' as 'Pathname' and its description as 'Name of the control file for the data set, suffix .ds'.

Примечания:

1. Имя исходного читаемого файла с замаскированными данными формируется по тому же шаблону, что и в задании маскирования (см. выше в этом разделе).

4.4.2.3 Дизайн управляющего задания для маскирования на месте

Пример дизайна управляющего задания-последовательности для реализации маскирования на месте приведён на рисунке ниже.



Задание состоит из двух последовательно выполняемых шагов:

1. Запуск задания маскирования в варианте «на месте»
2. Запуск задания загрузки замаскированных данных

Типовой состав параметров управляющего задания приведён на рисунке ниже.

| General Parameters Job control Dependencies NLS | | | | | |
|---|-------------------|----------------------|---------------|----------------------|--------------------------------------|
| | Parameter name | Prompt | Type | Default Value | Help Text |
| 1 | \$APT_CONFIG_FILE | Configuration file | Pathname | /datum/sw/Informatic | The Parallel job configuration file. |
| 2 | DbOutParams | DbOutParams paramet | Parameter Set | (As pre-defined) | Database connection parameters |
| 3 | DbParams | DbParams parameters | Parameter Set | (As pre-defined) | Database connection parameters |
| 4 | Globals | Globals parameters | Parameter Set | (As pre-defined) | |
| 5 | InputTable | InputTable | String | DEMO1.INPUT | |
| 6 | OutputTable | OutputTable | String | DEMO1.INPUT | |
| 7 | MaskingProfile | MaskingProfile | String | demo.demo1.input | |
| 8 | SafeInputTable | SafeInputTable | String | DEMO1-INPUT | |
| 9 | BatchId | Job batch identifier | String | - | |

Примечания:

1. Передаваемые параметры аналогичны параметрам для задания потокового маскирования.
2. Дополнительный параметр SafeInputTable используется для формирования имени экземпляра запуска задания. От параметра InputTable он отличается запретом на использование некоторых специальных символов, в том числе точки («.»).

На рисунке ниже показаны настройке шага запуска задания маскирования.

RunMask - Job Activity

General Job Triggers

Job name: LocalJdbc

Invocation Id: #DbParams.name#-#SafeInputTable#

Execution action: Reset if required, then run

☐ Do not checkpoint run.

Parameters

| Name | Value Expression |
|-------------------|-------------------|
| \$APT_CONFIG_FILE | \$APT_CONFIG_FILE |
| Globals | Globals |
| DbParams | DbParams |
| InputTable | InputTable |
| MaskingProfile | MaskingProfile |
| BatchId | BatchId |

Insert Parameter

Clear

Clear All

Set to Default

All to Default

Type: Pathname
Prompt: Configuration file

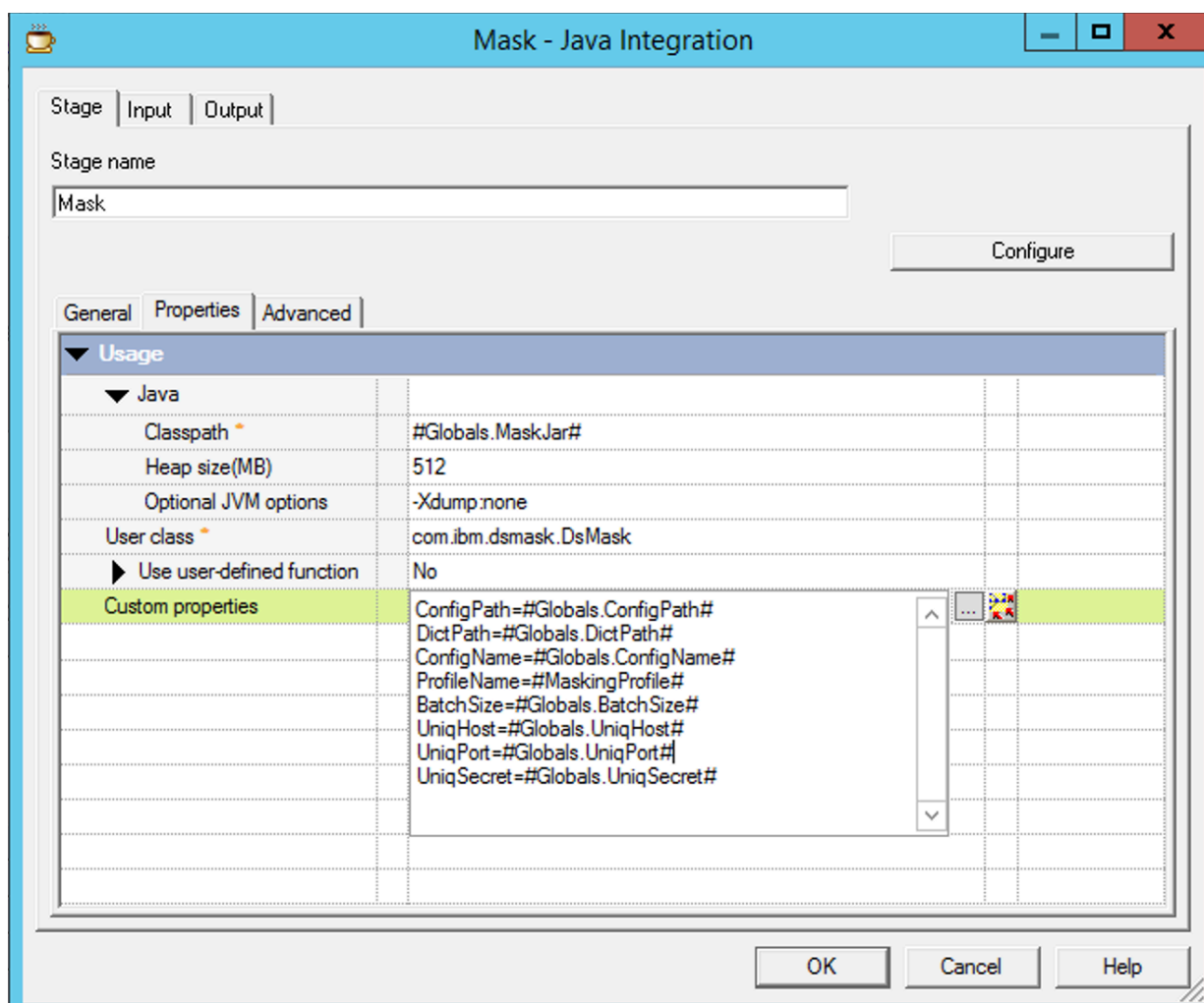
Примечания:

1. Имя запуска формируется на основе имени БД-источника и имени маскируемой таблицы, что облегчает диагностику при возникновении ошибок.
2. При необходимости перед запуском задания выполняется сброс его состояния (требуется, если предыдущий запуск завершился ошибкой).
3. Параметры запускаемого задания заполняются значениями параметров запускающего задания.

Настройки шага запуска задания загрузки данных аналогичны настройкам шага запуска задания маскирования, показанным выше.

4.4.3 Настройки оператора маскирования «DsMask»

Стандартные настройки оператора маскирования «DsMask» показаны на рисунке ниже.



Примечания:

1. Путь к классу оператора маскирования задаётся параметром MaskJar из набора параметров Globals. Имя класса реализации оператора маскирования – «com.ibm.dsmask.DsMask».
2. Увеличенный размер кучи (в примере на рисунке – 512 Мбайт) требуется для обработки сложных правил маскирования и при большом размере обрабатываемого пакета записей (задаётся параметром BatchSize).
3. Рекомендуется отключить генерацию диагностических дампов при ошибках (опция «-Xdump:none») во избежание возникновения переполнений файловой системы сервера DataStage.
4. Значения основных параметров оператора маскирования заданы через значения параметров группы Globals, что позволяет единообразно корректировать настройки операторов маскирования во всех используемых заданиях.

Описание параметров оператора маскирования приведено в таблице ниже.

| Параметр | Пример значения | Описание |
|------------|----------------------|---|
| DictPath | #Globals.DictPath# | Путь к каталогу со справочниками |
| ConfigPath | #Globals.ConfigPath# | Путь к каталогу с базами данных конфигураций маскирования |
| ConfigName | #Globals.ConfigName# | Логическое имя используемой конфигурационной базы данных маскирования |

| Параметр | Пример значения | Описание |
|-------------|----------------------|---|
| ProfileName | #MaskingProfile# | Имя профиля маскирования – логическое имя маскируемой таблицы в формате «БД.СХЕМА.ТАБЛИЦА» |
| BatchSize | #Globals.BatchSize# | Количество строк, маскируемых за одну пакетную операцию |
| UniqHost | #Globals.UniqHost# | Сетевое имя либо сетевой адрес сервера, на котором функционирует сервис контроля уникальности |
| UniqPort | #Globals.UniqPort# | Номер сетевого порта сервиса контроля уникальности |
| UniqSecret | #Globals.UniqSecret# | Код доступа к сервису контроля уникальности |

4.4.4 Настройки подключений к источникам и получателям данных

Подключения к источникам и получателям данных рекомендуется осуществлять с помощью специализированного коннектора, реализующего оптимизации для соответствующего вида СУБД. При недоступности специализированного коннектора, либо при работе с относительно небольшими объёмами данных можно использовать универсальный коннектор JDBC.

Для работы коннектора JDBC на сервере DataStage необходимо разместить соответствующий набор драйверов JDBC, которые требуется зарегистрировать в файле isjdbc.config. Подробная информация по настройке драйверов JDBC представлена в документации на DataStage.

Пример настройки параметров для подключения к источнику через JDBC-коннектор приведён на рисунке ниже.

| Connection | | Test Load Save |
|---|---------------------|----------------|
| URL * | #DbParams.url# | |
| User name | #DbParams.user# | |
| Password | #DbParams.password# | |
| Attributes | | |
| Usage | | View Data |
| Generate SQL at run time | Yes | |
| Table name * | #InputTable# | |
| Enable quoted identifiers | No | |
| SQL | | |
| ▶ Select statement * | | |
| Enable partitioned reads | No | |
| Transaction | | |
| Record count | 2000 | |
| Isolation level | Default | |
| Auto-commit mode | Disable | |
| Mark end of wave | No | |
| ▶ Run begin and end SQL statements | No | |
| Session | | |
| Fetch size | 0 | |
| Report schema mismatch | No | |
| Default length for columns | 200 | |
| Default length for long columns | 20000 | |
| Fail on truncation | Yes | |
| Generate all columns as Unicode | No | |
| ▶ Character set for non-Unicode columns | Default | |
| Keep conductor connection alive | Yes | |

| | |
|---------------------------------------|-------------|
| ▶ Run before and after SQL statements | No |
| ▼ Java settings | |
| Heap size | 512 |
| JVM options | -Xdump:none |
| ▶ Limit number of returned rows | No |

Рисунок. Параметры JDBC-подключения к БД-источнику.

Примечания:

1. Непосредственно свойства для установления соединения с БД настраиваются через набор параметров «DbParams», описанный в предыдущих разделах.
2. Текст запроса для выборки данных генерируется автоматически, при этом имя исходной таблицы определяется через параметр задания «InputTable».
3. При использовании JDBC-коннектора часто требуется увеличенный лимит на использование оперативной памяти (в примере выше установлено значение 512 Мбайт). Также для всех Java-компонентов целесообразно отключить создание диагностических дампов при ошибках (опция «-Xdump:none»).

Пример настройки параметров для подключения к получателю через JDBC-коннектор приведён на рисунке ниже.

| ▼ Connection Test Load Save | |
|--|------------------------|
| URL * | #DbOutParams.url# |
| User name | #DbOutParams.user# |
| Password | #DbOutParams.password# |
| Attributes | |
| ▼ Usage View Data | |
| Write mode | Insert |
| Generate SQL at run time | Yes |
| Table name * | #Output Table# |
| ▼ Table action * | Truncate |
| ▶ Generate create table statement at run time | Yes |
| ▶ Generate drop table statement at run time | Yes |
| ▼ Generate truncate table statement at run time | Yes |
| Stop the job when truncate table statement fails | No |
| Truncate table statement * | |
| Perform table action first | Yes |
| Enable quoted identifiers | No |
| ▼ SQL | |
| ▶ Insert statement * | |
| ▶ Update statement * | |
| ▶ Delete statement * | |
| ▶ Custom statements | |
| ▼ Transaction | |
| Record count | 2000 |
| Isolation level | Default |
| Auto-commit mode | Disable |
| ▶ Run begin and end SQL statements | No |

| | | | |
|---|-------------|--|--|
| ▼ Session | | | |
| Batch size | 2000 | | |
| Drop unmatched fields | No | | |
| Report schema mismatch | No | | |
| Default length for columns | 200 | | |
| Default length for long columns | 20000 | | |
| Fail on truncation | Yes | | |
| ▶ Character set for non-Unicode columns | Default | | |
| Keep conductor connection alive | Yes | | |
| ▶ Run before and after SQL statements | No | | |
| ▼ Java settings | | | |
| Heap size | 512 | | |
| JVM options | -Xdump:none | | |

Рисунок. Параметры JDBC-подключения к БД-получателю.

Примечания:

1. Непосредственно свойства для установления соединения с БД устанавливаются через набор параметров «DbOutParams», описанный ранее.
2. Текст запроса для выборки данных генерируется автоматически, при этом имя исходной таблицы определяется через параметр задания «OutputTable».
3. Таблица предварительно очищается через операцию TRUNCATE (синтаксис зависит от типа СУБД), текст оператора генерируется на основе имени таблицы, заданной в параметре «OutputTable».
4. Для виртуальной машины Java установлен увеличенный лимит памяти (512 Мбайт) и отключена генерация диагностических дампов («-Xdump:none»).

4.5 Ручной запуск заданий маскирования

Ручной запуск заданий маскирования обычно применяется при комплексной отладке заданий маскирования, правил маскирования и настроек подключения к базам данных, являющихся источниками исходной и получателями замаскированной информации.

Ручной запуск заданий маскирования может осуществляться одним из следующих способов:

- из приложения DataStage Designer Client;
- из Web-интерфейса DataStage Flow Designer;
- из командной строки с использованием команды dsjob.

Во всех случаях необходимо указать значения параметров задания, дождаться его завершения и проконтролировать отсутствие ошибок.

При использовании приложения DataStage Designer Client запуск задания, мониторинг хода его выполнения и просмотр протоколов работы может осуществляться в среде этого приложения.

При запуске задания из Web-интерфейса DataStage Flow Designer общий результат выполнения (успех/ошибка) доступен непосредственно в этом приложении. Для просмотра протокола задания необходимо перейти в Web-интерфейс Operations Console, либо воспользоваться командой dsjob.

При запуске задания с помощью команды dsjob мониторинг хода выполнения задания, а также просмотр протокола задания может осуществляться дополнительными вызовами команды dsjob, а также средствами Web-интерфейса Operations Console.

Более подробная информация о работе с заданиями DataStage приведена в документации на продукт IBM Information Server. В частности, опции запуска команды dsjob описаны в следующем разделе документации: <https://www.ibm.com/docs/en/iis/11.7?topic=interface-commands-controlling-infosphere-datastage-jobs>

4.6 Пакетное маскирование набора таблиц базы данных

В составе решения по маскированию предусмотрен пример скрипта MaskBatcher для пакетного запуска набора заданий по маскированию выбранной базы данных, который может служить основой для обеспечения автоматического выполнения процесса маскирования в соответствии с требованиями и спецификой систем Заказчика. Пример разработан на комбинации языков Groovy и Bourne Shell для максимального облегчения доработки и адаптации его логики.

Скрипт составляет список таблиц, содержащих конфиденциальные поля, путём выполнения запроса над базой данных Information Governance Catalog (так называемой базы данных ХМЕТА). Определение конфиденциальных полей осуществляется на основе присвоенных этим полям классов данных. Конфиденциальными считаются классы данных, связанные со специальным бизнес-термином «Конфиденциально».

Для каждой таблицы, требующей маскирования, выполняется запуск задания маскирования с помощью утилиты dsjob. Объединение заданий маскирования в согласованную совокупность операций маскирования для последующего мониторинга и контроля результатов осуществляется путём передачи в каждое запускаемое задание специального параметра – уникального идентификатора группы заданий, – на основе которого затем можно получить полный состав выполняемых заданий маскирования и их результаты.

Перед запуском заданий маскирования для определённой базы данных скрипт MaskBatcher проверяет наличие ранее запущенных заданий маскирования для той же самой базы данных, и отменяет запуск при их обнаружении.

Настройки скрипта MaskBatcher разделены на две части, хранящиеся в разных конфигурационных файлах:

- глобальные настройки, определяющие реквизиты подключения к базе данных ХМЕТА и детали запуска заданий DataStage;
- локальные настройки, определяющие логические имена входной и выходной баз данных.

Глобальные настройки читаются из файла «MaskBatcher-config.xml». Локальные настройки читаются из файла, имя которого указывается в параметре командной строки, либо (при отсутствии параметра) из файла «MaskBatcher-jobfile.xml».

Пример команды запуска пакетного маскирования:

```
./MaskBatcher.sh MaskBatcher-jobfile.xml
```

Пример файла локальных настроек:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<comment>Single job settings for the MaskBatcher program</comment>
<entry key="igc.dbname">AIS1</entry>
<entry key="input.conn">AIS_SRC</entry>
```

```
<entry key="output.conn">AIS_DST</entry>
</properties>
```

Пример файла глобальных настроек:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<comment>Configuration settings for the MaskBatcher program</comment>
<entry key="igc.url">jdbc:db2://localhost:50000/xmeta</entry>
<entry key="igc.username">xmeta</entry>
<entry key="igc.password">passw0rd</entry>
<entry key="job.runner">ssh dsadm@localhost
/opt/IBM/Masker/batcher/RunJob.sh</entry>
<entry key="job.lister">ssh dsadm@localhost
/opt/IBM/Masker/batcher/ListJobs.sh</entry>
</properties>
```

Описание настроек скрипта MaskBatcher приведено в таблице ниже.

| Настройка | Описание |
|--------------|---|
| igc.dbname | Логическое имя маскируемой базы данных в рамках IGC |
| input.conn | Имя значения в наборе параметров «DbParams» для подключения к БД-источнику |
| output.conn | Имя значения в наборе параметров «DbOutParams» для подключения к БД-получателю |
| igc.url | JDBC URL для подключения к БД ХМЕТА |
| igc.username | Логин пользователя для подключения к БД ХМЕТА |
| igc.password | Пароль пользователя для подключения к БД ХМЕТА |
| job.runner | Команда для запуска задания. Пример скрипта для запуска заданий подготовлен в виде файла RunJob.sh. Для подключения к серверу DataStage с необходимыми правами обычно используется запуск скрипта через SSH. |
| job.lister | Команда для получения списка выполняемых заданий. Пример скрипта для получения списка заданий подготовлен в виде файла ListJobs.sh. Для подключения к серверу DataStage с необходимыми правами обычно используется запуск скрипта через SSH. |

Статусы выполнения и журналы работы отдельных заданий маскирования можно просмотреть через Web-интерфейс инструмента Operations Console из состава IBM Information Server.

Для более удобного просмотра запусков пакетных заданий рекомендовано подключить любой инструмент формирования отчётности к базе данных операционных метаданных (часто совмещается с ХМЕТА). Примеры запросов для выборки маскируемых баз данных, выполненных пакетных операций маскирования и сводки по маскируемым таблицам включены в состав решения по маскированию.

Примечание. В составе платформы IBM Information Server для формирования отчётов поставляется инструмент IBM Cognos Analytics. Возможно использование любых других инструментов построения отчётности, поддерживающих работу с источниками данных через JDBC-драйвер.

5 Разработка и настройка заданий генерации справочников

5.1 Справочник подстановки для перемешивания исходных значений

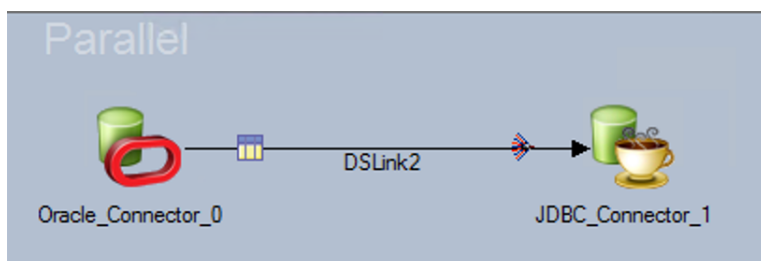
Для реализации маскирования перемешиванием исходных значений необходимо сформировать справочник подстановок, а затем использовать справочник подстановок в правиле маскирования, вызвав алгоритм подстановки значения по справочнику.

Примечание. Метод маскирования перемешиванием не может обеспечить стабильный результат маскирования при изменении состава исходных (перемешиваемых) значений. Например, при расширении исходного набора значений перемешивание необходимо производить заново, и оно даст другой результат маскирования для всех или почти всех маскируемых значений.

Справочник подстановок формируется с помощью задания DataStage из двух шагов:

1. Выборка таблицы подстановок запросом из базы данных системы-источника либо специальной служебной базы данных.
2. Сохранение записей таблицы подстановок в таблицу файловой базы данных H2.

Пример дизайна задания для формирования справочника подстановок показан на рисунке ниже.



Примечания:

1. На шаге 1 используется коннектор к БД-источнику, либо обобщённый JDBC-коннектор. Используется специальный SQL-запрос для формирования таблицы подстановок (см. пример ниже).
2. На шаге 2 используется JDBC-коннектор к базе данных H2.

Типичные настройки коннектора к источнику данных, используемые в задании генерации справочника подстановок, приведены на рисунке ниже. Используется заданный пользователем оператор выборки данных (SELECT).

| ▼ Connection | | Test | Load | Save |
|---------------------------------|-----------------------------|-----------|------|------|
| Server | #MaskSrc.DB_NAME# | | | |
| User name * | #MaskSrc.DB_USER# | | | |
| Password * | #MaskSrc.DB_PASS# | | | |
| Use external authentication | No | | | |
| Oracle client version | 12c | | | |
| ▼ Usage | | View Data | | |
| Read mode | Select | | | |
| Generate SQL at runtime | No | | | |
| ▶ Table name * | | | | |
| Enable quoted identifiers | No | | | |
| ▼ SQL | | | | |
| Where clause | | | | |
| Other clause | | | | |
| ▼ Select statement * | (SELECT isn AS isn_1, is... | | | |
| Read select statement from file | No | | | |
| ▶ PL/SQL block * | | | | |
| ▶ Enable partitioned reads | No | | | |
| ▼ Transaction | | | | |
| Isolation level | Read committed | | | |
| Record count | 2000 | | | |
| Mark end of wave | No | | | |
| ▼ Session | | | | |
| Array size | 2000 | | | |
| Prefetch row count | 1 | | | |
| Prefetch buffer size | 0 | | | |
| ▶ Enable LOB references | No | | | |

Пример настройки JDBC URL для подключения к БД H2 (настройка «Connection / URL» в свойствах JDBC-коннектора):

```
jdbc:h2:#MaskGlobals.DICT_PATH#/demo-subst;  
AUTO_SERVER=TRUE;FILE_LOCK=SOCKET;COMPRESS=TRUE
```

- глобальная настройка MaskGlobals.DICT_PATH указывает на путь к каталогу для хранения справочников;
- значение «demo-subst» - логическое имя базы данных, на основе которого формируются имена файлов H2;
- опции AUTO_SERVER=TRUE и FILE_LOCK=SOCKET необходимы для работы с базами данных H2 в среде DataStage;
- опция COMPRESS=TRUE включает встроенный алгоритм сжатия базы данных H2 для сокращения объема хранения на диске.

Типичные настройки JDBC-коннектора к файловой базе данных H2, в которую записывается формируемая таблица подстановок, показаны на рисунке ниже.

| ▼ Connection | | Test | Load | Save |
|---|--------------------|-----------|------|------|
| URL * | jdbc:h2:MaskGlo... | | | |
| User name | | | | |
| Password | | | | |
| Attributes | | | | |
| ▼ Usage | | View Data | | |
| Write mode | Insert | | | |
| Generate SQL at run time | Yes | | | |
| Table name * | ISN_SUBST | | | |
| ▼ Table action * | Replace | | | |
| ▼ Generate create table statement at run time | Yes | | | |
| Stop the job when create table statement fails | Yes | | | |
| Create table statement * | | | | |
| ▼ Generate drop table statement at run time | Yes | | | |
| Stop the job when drop table statement fails | No | | | |
| Drop table statement * | | | | |
| ▶ Generate truncate table statement at run time | Yes | | | |
| Perform table action first | Yes | | | |
| Enable quoted identifiers | No | | | |
| ▼ SQL | | | | |
| ▶ Insert statement * | | | | |
| ▶ Update statement * | | | | |
| ▶ Delete statement * | | | | |
| ▶ Custom statements | | | | |
| ▼ Transaction | | | | |
| Record count | 20000 | | | |
| Isolation level | Default | | | |
| Auto-commit mode | Disable | | | |
| ▶ Run begin and end SQL statements | No | | | |
| ▼ Session | | | | |
| Batch size | 20000 | | | |
| Drop unmatched fields | No | | | |
| Report schema mismatch | No | | | |
| Default length for columns | 200 | | | |
| Default length for long columns | 20000 | | | |
| Fail on truncation | Yes | | | |
| ▶ Character set for non-Unicode columns | Default | | | |
| Keep conductor connection alive | Yes | | | |
| ▶ Run before and after SQL statements | No | | | |
| ▼ Java settings | | | | |
| Heap size | 512 | | | |
| JVM options | -Xdump:none | | | |

Примечания:

1. В настройках указано имя выходной таблицы (ISN_SUBST), перед записью в таблицу настроено действие по её пересозданию (Replace).
2. Для оптимизации записи в JDBC-коннекторе используется увеличенный размер транзакции и пакета операций вставки (в обоих случаях используется значение 20000).
3. Дополнительно установлен увеличенный размер разрешённой памяти для виртуальной машины Java (512 Мбайт) и отключена генерация диагностических дампов при ошибках (параметр «-Xdump:none»).

Пример запроса для формирования таблицы подстановок для БД Oracle Database:

```
(SELECT v AS v_1, v AS v_2 FROM myschema.mytable WHERE v<100)
  UNION ALL
(SELECT tab1.v_1, tab2.v_2
 FROM
  (SELECT ROWNUM AS pos_1, v AS v_1 FROM myschema.mytable WHERE v>=100) tab1,
  (SELECT ROWNUM AS pos_2, v AS v_2 FROM
    (SELECT v FROM myschema.mytable WHERE v>=100
     ORDER BY dbms_random.value()) x) tab2
 WHERE tab1.pos_1=tab2.pos_2)
```

В этом запросе формируется таблица подстановок для поля «V» таблицы «MYTABLE» в схеме «MYSCHEMA». Значения менее 100 остаются неизменными (замещаются на идентичные), так как являются служебными. Значения от 100 и выше в этом примере заменяются на результат случайного перемешивания.

Приведённый выше запрос возвращает разные результаты при повторных запусках над одними и теми же данными. Возможны альтернативные формы запроса, использующие сортировку не по псевдослучайному значению, как в примере выше, а по значению хеш-функции над исходным значением (с добавлением дополнительного значения в виде «соли»). Такой вариант запроса возвращает одни и те же значения при сохранении содержимого исходной таблицы. См. пример альтернативной формы ниже:

```
(SELECT v AS v_1, v AS v_2 FROM myschema.mytable WHERE v<100)
  UNION ALL
(SELECT tab1.v_1, tab2.v_2
 FROM
  (SELECT ROWNUM AS pos_1, v AS v_1 FROM
    (SELECT v FROM myschema.mytable WHERE v>=100
     ORDER BY v) y) tab1,
  (SELECT ROWNUM AS pos_2, v AS v_2 FROM
    (SELECT v FROM myschema.mytable WHERE v>=100
     ORDER BY STANDARD_HASH(TO_CHAR(v) || 'P@$$w0rd')) x) tab2
 WHERE tab1.pos_1=tab2.pos_2)
```

Обратите внимание на необходимость сортировки записей в подзапросе «tab1» для получения предсказуемых и воспроизводимых результатов.

5.2 Справочник подстановки по ключу

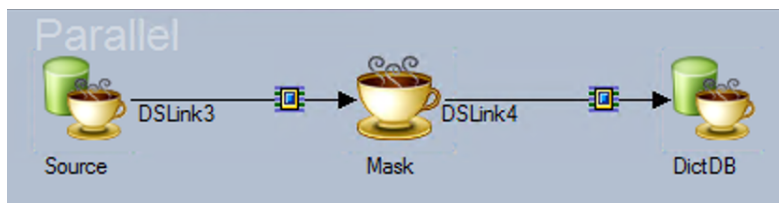
Данный вид справочников используется, к примеру, для маскирования неполных значений согласованно с полными значениями (например, «Иванов И.И.» vs «Иванов Иван Иванович»). Требуется наличие в маскируемых таблицах уникального идентификатора маскируемой сущности (например, идентификатор клиентской записи).

Маскирование осуществляется правилом, включающим в себя уникальный идентификатор сущности (для которого необходимо завести специальный класс данных), полное значение и варианты неполных значений (для которых потребуются отдельные конфиденциальные классы данных).

Справочник подстановок формируется с помощью задания DataStage из трёх шагов:

1. Выборка исходных данных для маскирования запросом к базе данных системы-источника либо специальной служебной базе данных.
2. Маскирование для заполнения справочника.
3. Сохранение записей таблицы подстановок в таблицу файловой базы данных H2.

Пример дизайна задания для формирования справочника подстановок показан на рисунке ниже.



Примечания:

1. На шаге 1 используется коннектор к БД-источнику, либо обобщённый коннектор типа JDBC или ODBC. Используется SQL-запрос для получения уникального идентификатора сущности и полного варианта маскируемого значения.
2. На шаге 2 используется оператор маскирования «DsMask». Обычно при этом применяется альтернативная база данных конфигурации маскирования, в которую помещается специальный технический профиль маскирования для таблицы-справочника (см. описание ниже). Сокращённые замаскированные варианты значений при этом формируются на основе полного замаскированного значения.
3. На шаге 3 используется JDBC-коннектор к базе данных H2 для создания новой либо открытия существующей базы данных со справочниками, и сохранения в неё новой справочной таблицы.

Настройки коннектора для исходной БД и коннектора для целевой БД H2 аналогичны (с точностью до текста используемого SQL-запроса, имени целевой БД и имени целевой таблицы подстановок) приведённым в разделе 5.1 *Справочник подстановки для перемешивания исходных значений*.

Формирование технических профилей маскирования для фиктивных таблиц осуществляется с помощью конфигурационной программы DsMask путём указания структуры фиктивных таблиц, а также присвоенных колонкам таблиц классов данных в виде XML-документа специального формата.

Имя файла с XML-документом указывается в параметре «in.tab.file.name» конфигурационной программы «DsMask», при этом параметр «in.tab.type» должен включать в себя значение «file» (см. раздел 4.2 *Запуск конфигурационной программы «DsMask»*).

Пример структуры XML-документа для описания структуры фиктивных таблиц:

```
<?xml version="1.0" encoding="UTF-8"?>
<dsmask-table-info>
  <table db="sys-dict" name="demo1.phys_cust">
    <field name="customer_id" dcs="" />
    <field name="sex" dcs="Sex" />
    <field name="full_name" dcs="LastFirstMiddleNameRus" />
  </table>
</dsmask-table-info>
```

```

    <field name="full_name_lat" dcs="LastFirstMiddleNameLat" />
    <field name="first_name" dcs="FirstNameRus" />
    <field name="middle_name" dcs="MiddleNameRus" />
    <field name="last_name" dcs="LastNameRus" />
    <field name="dob" dcs="DOB" />
  </table>
</dsmask-table-info>

```

В приведённом выше примере описана фиктивная таблица «demo1.phys_cust», в которой поле «customer_id» не маскируется (так как ему не приписаны классы конфиденциальной информации). Для остальных полей в атрибуте «dcs» указаны классы конфиденциальной информации, в соответствии с которыми должно быть выполнено маскирование.

При маскировании фиктивной таблицы указывается имя профиля маскирования, соответствующее указанному в XML-документе при запуске конфигурационной программы. В результате формируется таблица подстановок, позволяющая сопоставить значения ключевых полей (например, поле «customer_id» в примере выше) с замаскированными значениями конфиденциальных полей.

5.3 Генерация справочника подстановок для маскирования ФИО

Генерация справочника подстановок для маскирования ФИО реализована специальным модулем в составе конфигурационной программы «DsMask».

Настройка генерации справочника подстановок ФИО осуществляются путём создания конфигурационного файла в формате XML. Пример конфигурационного файла входит в состав конфигурационной программы «DsMask» и поставляется в виде файла «jconf-dict-names.xml».

Описание параметров генерации справочника подстановок ФИО приведено в таблице ниже.

| Параметр | Описание |
|-------------|---|
| NamesMale | Имя текстового файла с мужскими именами и отчествами в мужской и женской форме |
| NamesFemale | Имя текстового файла с женскими именами |
| FamMale | Имя текстового файла с мужскими фамилиями |
| FamFemale | Имя текстового файла с женскими фамилиями |
| SaltKey | Ключ для упорядочивания комбинаций, меняется при необходимости изменения содержания генерируемого справочника |
| DbName | Имя выходной БД в формате H2 |
| TotalNames | Количество генерируемых ФИО, поровну мужских и женских |

Пример файла настроек генерации справочника подстановок ФИО приведён ниже.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<comment>Sample job file for dictionary builder</comment>
<entry key="NamesMale">./names-data/names_m.txt</entry>

```

```

<entry key="NamesFemale">./names-data/names_f.txt</entry>
<entry key="FamMale">./names-data/fam_m.txt</entry>
<entry key="FamFemale">./names-data/fam_f.txt</entry>
<entry key="SaltKey">qazwsx</entry>
<entry key="DbName">./dict/dict-names</entry>
<entry key="TotalNames">10000000</entry>
</properties>

```

Запуск модуля генерации справочника подстановок ФИО осуществляется следующими командами:

```

cd КаталогУстановкиDsMask/dsmask-jconf
./BuildDict.sh ФайлНастроек.xml

```

Перечень стандартных таблиц подстановок ФИО, формируемых модулем генерации типовых справочников в составе конфигурационной программы «DsMask», приведен ниже.

| Имя таблицы | Описание |
|-----------------|--|
| DICT_FIO | Справочник значений ФИО в мужском и женском форматах |
| DICT_FIO_MALE | Справочник значений ФИО в мужском формате |
| DICT_FIO_FEMALE | Справочник значений ФИО в женском формате |

Структура стандартных таблиц подстановки ФИО приведена ниже.

| Имя поля | Тип данных | Назначение |
|----------|--------------|--|
| ID | INTEGER | Номер записи, от 0 до N-1, где N – размер справочника |
| NFULL | VARCHAR(150) | Полное ФИО в формате «Фамилия Имя Отчество» |
| NFIRST | VARCHAR(40) | Имя |
| NMIDDLE | VARCHAR(40) | Отчество |
| NLAST | VARCHAR(40) | Фамилия |
| SEX | CHAR(1) | Признак пола, значение «F» либо «M». Поле присутствует только в таблице DICT_FIO (в остальных таблицах оно не имеет смысла). |

6 Общий синтаксис правил маскирования

Правила маскирования описываются в формате XML и сохраняются в файлы с произвольными именами и расширением «.xml». Набор правил маскирования помещается (в виде набора файлов) в определённый каталог и его подкаталоги.

Путь к каталогу правил маскирования указывается в параметрах конфигурационной программы «DsMask».

Правила маскирования включают в себя элементы следующих видов:

- ключ – текстовое значение, используемое для инициализации алгоритма маскирования. Изменение значения ключа меняет результаты маскирования;
- функция – обращение к алгоритму маскирования с указанием вида алгоритма и его параметров;
- правило – связывает набор входных и выходных классов конфиденциальной информации и устанавливает последовательность операций маскирования в виде вызовов определённых функций.

6.1 Ключи инициализации

Пример синтаксиса для задания ключа:

```
<key name="KEY-NAME" value="KEY-VALUE"/>
```

Здесь KEY-NAME – имя ключа, которое будет использоваться при настройке функций для ссылки на ключ, а KEY-VALUE – строковое значение ключа.

6.2 Функции для обращения к встроенным алгоритмам

Примеры синтаксиса функций для обращения к встроенным алгоритмам:

```
<function name="vu-concat" type="Concat"/>
```

```
<function name="vu-fpe" type="FPE">
  <![CDATA[
    KEY default
    CLASS russian-simple
    SKIP-BEFORE 4
  ]]>
</function>
```

В примере выше определены две функции с именами «vu-concat» и «vu-fpe», которые используют алгоритмы конкатенации строк («Concat») и хеширования с сохранением формата (FPE), соответственно. Для функции «vu-fpe» заданы параметры, а именно имя ключа («default»), справочник видов символов («russian-simple») и количество не изменяемых символов в начале строки (4).

Дополнительный пример функции для маскирования дат:

```
<function name="dob-simple" type="DateOp">
  <![CDATA[
    KEY default
  ]]>
</function>
```

В примере выше определена функция «dob-simple», осуществляющая вызов встроенного алгоритма для маскирования дат «DateOp». В алгоритм передаётся в качестве параметра имя ключа инициализации «default».

6.3 Функции-скрипты Groovy

Для реализации пользовательской логики обработки данных предусмотрена возможность создания функций в виде скриптов на языке программирования Groovy.

Описание функции состоит из двух частей: тело функции и тело скрипта для генерации примера аргументов вызова функции. Пример аргументов вызова используется для автоматической проверки работоспособности функции в процессе работы конфигурационной программы «DsMask».

Пример простой функции на языке Groovy:

```
<function name="name-get-1" type="GroovyScript">

    <text><![CDATA[
return 'M'.equalsIgnoreCase(input[0]) ? input[1] : input[2]
    ]]></text>

    <text-input><![CDATA[
return ['M', 'Пагосян Гагик Спартакович', 'Фанова Анна Ивановна'] as Object[]
    ]]></text-input>

</function>
```

Пример чуть более сложной функции на языке Groovy:

```
<function name="snils-normalize" type="GroovyScript">
    <text><![CDATA[
// Exclude all non-numeric characters from the input
Object invoke(Object input) {
    if (input==null)
        return null;
    String str = input.toString().trim();
    str = str.replaceAll("[^\d]", "");
    return str;
}
    ]]></text>

    <text-input><![CDATA[
return "112-233-445 95";
    ]]></text-input>

</function>
```

6.4 Синтаксис правил маскирования

Правила маскирования связывают входные и выходные классы данных с последовательностью шагов маскирования, каждый из которых является вызовом функции маскирования.

Пример синтаксиса для определения простого правила маскирования:

```
<rule name="DOB">
  <input dc="DOB"/>
  <output dc="DOB"/>
  <step name="mask" function="dob-simple">
    <ref name="$" pos="1"/>
  </step>
</rule>
```

В примере выше определено простое правило для маскирования даты рождения, названное «DOB». Это правило маскирует поля, с которыми связан класс данных «DOB» (дата рождения, Date Of Birth), путём вызова функции «dob-simple» над исходным значением даты рождения.

Аргументы вызова функций маскирования определяются в виде последовательности тегов «ref», в каждом из которых указывается атрибут «name» (имя одного из предыдущих шагов, либо значение «\$» для ссылки на вход правила маскирования) и атрибут «pos» (позиция в векторе выходных значений, начиная с 1) для определения значения аргумента.

Пример более сложного правила маскирования:

```
<rule name="name-full-sex-hash">
  <input dc="GEN"/>
  <input dc="LastFirstMiddleNameRus"/>
  <output dc="LastFirstMiddleNameRus"/>
  <step name="norm" function="name-norm">
    <ref pos="2"/>
  </step>
  <step name="hash-male" function="name-male-hash">
    <predicate> <![CDATA[
return 'M'.equalsIgnoreCase(input[1])
]]>
    </predicate>
    <ref name="norm" pos="1"/>
    <ref name="$" pos="1"/>
  </step>
  <step name="hash-female" function="name-female-hash">
    <predicate> <![CDATA[
return 'F'.equalsIgnoreCase(input[1])
]]>
    </predicate>
    <ref name="norm" pos="1"/>
    <ref name="$" pos="1"/>
  </step>
  <step name="project" function="name-get-1">
    <ref name="$" pos="1"/>
    <ref name="hash-male" pos="1"/>
    <ref name="hash-female" pos="1"/>
  </step>
</rule>
```

Данное правило принимает на вход информацию о признаке пола («GEN» от Gender) и полное ФИО, и возвращает результат маскирования ФИО с учётом признака пола. Процесс маскирования включает в себя:

1. вызовы операций нормализации ФИО по регистру символов и пробелам (функция «name-norm»),
2. обращение к одной из двух таблиц подстановок через вызовы функций «name-male-hash» и «name-female-hash»,
3. формирование итогового результата подстановки вызовом функции «name-get-1».

Лишние обращения к таблицам подстановок блокируются с использованием механизма предикатов (тег «predicate» в соответствующих шагах алгоритма).

6.5 Контроль уникальности подстановок

Настройка контроля уникальности подстановок выполняется на уровне одного из шагов правила маскирования. Используемая в этом шаге функция маскирования должна поддерживать итеративное выполнение (применимо к основным алгоритмам хеширования и подстановок на основе хеш-кода).

Пример правила маскирования, использующего функцию контроля уникальности подстановок:

```
<rule name="card-pan">
  <input dc="CardPan"/>
  <output dc="CardPan"/>
  <step name="norm" function="card-normalize">
    <ref name="$" pos="1"/>
  </step>
  <step name="mask" function="card-fpe">
    <ref name="norm" pos="1"/>
    <uniq-check provider="card-pan">
      <uniq-input pos="1"/>
      <uniq-output pos="1"/>
    </uniq-check>
  </step>
  <step name="luhn" function="card-luhn">
    <ref name="mask" pos="1"/>
  </step>
</rule>
```

В примере выше показано правило для маскирования номера банковской карты с использованием алгоритма FPE. Данный алгоритм может формировать коллизии, т.е. одинаковые замаскированные значения для разных исходных значений, что обычно проявляется при очень большом количестве исходных маскируемых значений (от нескольких миллионов).

Для исключения коллизий в настройки правила маскирования было добавлено правило контроля уникальности в виде тега «uniq-check», в котором указывается:

- логическое имя провайдера уникальных значений («card-pan»), в рамках которого сервис контроля уникальности будет контролировать и корректировать наличие коллизий;
- состав входных (исходных, незамаскированных) и выходных (замаскированных) атрибутов для контроля, в виде последовательности тегов «uniq-input» и «uniq-output».

Входные и выходные атрибуты для контроля уникальности указываются как позиции (начиная с 1) во входных аргументах и в результате работы функции маскирования.

Теги «uniq-input» и «uniq-output» могут отсутствовать, в этом случае контроль уникальности ведётся на основе полного набора входных параметров и результатов выполнения функции маскирования.

При обнаружении сервисом контроля уникальности коллизии оператор маскирования повторяет вызов функции маскирования, увеличив передаваемый номер итерации. Логика используемого алгоритма маскирования использует номер итерации при формировании замаскированного значения, в результате будет сформировано другое значение, которое также будет проверено на уникальность.

Если достаточно большое количество итераций (по умолчанию – 50000) не позволило сформировать уникальное значение, для соответствующей маскируемой строки сигнализируется ошибка маскирования. В этом случае рекомендуется либо отказаться от контроля уникальности, либо скорректировать параметры алгоритма маскирования таким образом, чтобы снизить вероятность коллизий.

В качестве оптимизации сервис контроля уникальности фиксирует номер итерации, на которой для заданного исходного значения была преодолена возникшая коллизия. Эта оптимизация позволяет при повторном маскировании выполнять не более двух итераций для каждой из выявленных коллизий (одна – для обнаружения факта коллизии и получения нужного номера итерации, вторая – для получения итогового замаскированного значения).

7 Спецификация встроенных алгоритмов

7.1 Числовой хеш-код

Идентификатор алгоритма: NumberHash

Поддержка итераций: да

Параметры алгоритма приведены в таблице ниже.

| Параметр | Описание |
|----------|---|
| KEY | Имя ключа инициализации. Опциональный, по умолчанию отсутствует |
| FROM | Левая граница диапазона значений, целое 32-битное число. Опциональный, по умолчанию равен 0. |
| TO | Правая граница диапазона значений, целое 32-битное число. Опциональный, по умолчанию равен 10. |

Входные значения используются для расчёта контрольной суммы CRC-32.

Массивы октетов обрабатываются без дополнительных преобразований, текстовые данные интерпретируются как массивы октетов в кодировке UTF-8. Прочие типы данных преобразуются к текстовому формату и далее обрабатываются аналогично текстовым данным. При передаче нескольких полей они разделяются при обработке октетом со значением 1.

Если в параметрах задан ключ, его значение обрабатывается алгоритмом расчёта контрольной суммы после обработки всех входных значений.

Результатом работы алгоритма является целое число в указанном диапазоне, полученное на основе расчёта контрольной суммы CRC32 над входными данными (начальная граница диапазона плюс остаток от деления значения хеш-кода на величину диапазона).

Пример настройки функции маскирования с использованием этого алгоритма:

```
<function name="test-number-hash" type="NumberHash">
  <![CDATA[
    KEY default
    FROM 0
    TO 10000
  ]]>
</function>
```

7.2 Двоичный хеш-код

Идентификатор алгоритма: DigestHash

Поддержка итераций: да

Параметры алгоритма приведены в таблице ниже.

| Параметр | Описание |
|----------|--|
| KEY | Имя ключа инициализации. Опциональный, по умолчанию отсутствует |
| TYPE | Идентификатор алгоритма Message Digest, см. перечень в документе: https://www.ibm.com/docs/en/sdk-java-technology/8?topic=icaasr-appendix-standard-names |

Входные значения используются для расчёта хеш-кода в соответствии с алгоритмом, установленным в параметре TYPE. При отсутствии установленного параметра TYPE используется алгоритм SHA-512.

Массивы октетов обрабатываются без дополнительных преобразований, текстовые данные интерпретируются как массивы октетов в кодировке UTF-8. Прочие типы данных преобразуются к текстовому формату и далее обрабатываются аналогично текстовым данным. При передаче нескольких полей они разделяются при обработке октетом со значением 1.

Если в параметрах задан ключ, его значение обрабатывается алгоритмом хеширования после обработки всех входных значений.

Результатом работы алгоритма является хеш-код над входными данными, возвращаемый в двух форматах: в шестнадцатеричного представления (первый элемент) и в формате массива октетов (второй элемент).

Пример настройки функции маскирования с использованием этого алгоритма:

```
<function name="test-digest-hash" type="DigestHash">
  <![CDATA[
    KEY default
    TYPE SHA-1
  ]]>
```

</function>

7.3 Маскирование даты

Идентификатор алгоритма: DateOp

Поддержка итераций: нет (отсутствует, при необходимости может быть добавлена)

Параметры алгоритма приведены в таблице ниже.

| Параметр | Описание |
|----------|--|
| KEY | Имя ключа инициализации. Обязательный параметр |
| FORMAT | Формат даты или даты/времени, используемый при разборе строкового представления входных значений. Опциональный, по умолчанию устанавливается в значение «yyyy-mm-dd» |

Алгоритм поддерживает маскирование данных в формате даты или даты-времени, включая обработку значений, представленных как строки в заданном формате. Применение алгоритма наиболее целесообразно для маскирования даты рождения как персональных данных.

Маскируемое значение переводится в строковый формат даты (время, при наличии, отбрасывается), после чего для полученной строки, дополненной ключом инициализации, вычисляется хеш-код CRC-32.

Результирующее значение даты и метки времени формируется с сохранением номера года, при этом остаток от деления вычисленного значения хеш-кода на число 365 интерпретируется как номер дня в году.

Выходное значение имеет тот же тип данных, что и входное (дата, метка времени или форматированная строка с датой и, опционально, меткой времени).

Формат даты или времени при работе со строковыми данными задаётся как шаблон, совместимый с классом `java.text.SimpleDateFormat` (см. описание в документации: <https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html>).

Пример настройки функции маскирования с использованием этого алгоритма:

```
<function name="test-dateop" type="DateOp">
  <![CDATA[
    KEY default
  ]]>
</function>
```

7.4 Хеширование с сохранением формата

Идентификатор алгоритма: FPE

Поддержка итераций: да

Параметры алгоритма приведены в таблице ниже.

| Параметр | Описание |
|-------------|---|
| KEY | Имя ключа инициализации. Опциональный, по умолчанию отсутствует |
| CLASS | Идентификатор набора классов символов (см. описание ниже). Обязательный параметр |
| SKIP-BEFORE | Количество пропускаемых символов в начале входной строки, по умолчанию 0 |
| SKIP-AFTER | Количество пропускаемых символов в конце входной строки, по умолчанию 0 |

Алгоритм применяется только к единичным значениям, интерпретируемым как строки. Если при настройке правила маскирования в алгоритм подано сразу несколько значений, каждое из этих значений будет обработано независимо от других, формируя отдельное выходное значение.

При отсутствии установленного ключа инициализации (параметра KEY) используется значение ключа по умолчанию, в качестве которого используется недокументированное фиксированное значение. Рекомендовано всегда указывать ключ инициализации.

Алгоритм рассматривает входную строку как последовательность символов в кодировке Unicode, в том числе поддерживая составные символы. Результатом работы алгоритма является модифицированная последовательность символов в кодировке Unicode той же длины, что и входное значение, при этом количество байт в кодировке UTF-8 может измениться.

В процессе работы алгоритма осуществляется замена символов исходной строки на другие символы, принадлежащие тому же самому классу символов. Используемый алгоритмом набор классов символов определяется через параметр CLASS. Символы входной строки, не принадлежащие ни одному из классов символов, алгоритм оставляет неизменными.

На первом этапе работы алгоритма символы в количестве SKIP-BEFORE в начале входной строки и SKIP-AFTER в конце входной строки копируются в выходную строку без изменений. Таким образом, если длина строки меньше либо равна сумме значений параметров SKIP-BEFORE и SKIP-AFTER, возвращается неизменённая строка.

На втором этапе работы алгоритма входная строка интерпретируется (целиком) как массив октетов в кодировке UTF-8, а затем используется для вычисления кода проверки подлинности сообщения (HMAC) в соответствии с алгоритмом HMAC-SHA512.

На третьем этапе работы алгоритма осуществляется последовательная обработка каждого символа входной строки, за исключением обработанных на первом этапе. Для каждого символа определяется его класс, после чего вычисленное на втором этапе значение кода HMAC используется (при необходимости циклически) для формирования индекса подстановочного символа из набора символов того же самого класса. Допускается совпадение исходного и подставляемого вместо него символов.

Описания наборов классов символов загружаются из каталога, заданного параметром «DictPath» оператора маскирования (см. раздел 4.4.3 *Настройки оператора маскирования «DsMask»*). Каждый набор классов символов настраивается в виде файла в формате XML, имя которого

формируется по следующему шаблону: «`css-ИмяНабора.xml`», где `ИмяНабора` – идентификатор набора классов символов, задаваемый параметром `CLASS` алгоритма.

Пример настройки класса символов для маскирования русских и латинских букв, а также десятичных цифр:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<char-class-set name="russian-simple">
  <char-class name="numbers">
    <char-range begin-char="0" end-char="9"/>
  </char-class>
  <char-class name="small-latin">
    <char-range begin-char="a" end-char="z"/>
  </char-class>
  <char-class name="large-latin">
    <char-range begin-char="A" end-char="Z"/>
  </char-class>
  <char-class name="small-cyrillic">
    <char-range begin-char="a" end-char="я"/>
    <char-range begin-char="ё" end-char="ё"/>
  </char-class>
  <char-class name="large-cyrillic">
    <char-range begin-char="А" end-char="Я"/>
    <char-range begin-char="Ё" end-char="Ё"/>
  </char-class>
</char-class-set>
```

В примере выше диапазоны символов задаются путём непосредственного указания начальных и конечных символов в их текстовом представлении с использованием атрибутов `begin-char` и `end-char`. Вместо этого допускается задание символов в виде шестнадцатеричного представления в кодировке Unicode, тогда используются атрибуты `begin-hex` и `end-hex`.

Пример настройки функции маскирования с использованием этого алгоритма:

```
<function name="test-fpe" type="FPE">
  <![CDATA[
    KEY default
    CLASS russian-simple
  ]]>
</function>
```

7.5 Подстановка по хеш-коду

Идентификатор алгоритма: `HashLookup`

Поддержка итераций: да

Параметры алгоритма приведены в таблице ниже.

| Параметр | Описание |
|----------|--|
| KEY | Имя ключа инициализации. Обязательный параметр |

| Параметр | Описание |
|----------|--|
| DB | Имя базы данных, содержащей таблицу подстановок. Обязательный параметр |
| TABLE | Имя таблицы подстановок. Обязательный параметр |
| ID | Имя поля таблицы подстановок, содержащего индекс подстановки (уникальные числовые значения в диапазоне от 0 до N-1, где N – количество строк таблицы подстановок). Обязательный параметр |
| OUT | Имена полей таблицы подстановок, значения которых будут результатом вызова алгоритма. Обязательный параметр |
| INDEXES | Список позиций в векторе входных значений, которые должны быть использованы для расчёта хеш-кода. Нумерация позиций начинается с 1 (единицы). Опциональный параметр, при его отсутствии хеш-код вычисляется по всем входным параметрам |
| USER | Имя пользователя-владельца базы данных. Опциональный параметр, по умолчанию имя пользователя не устанавливается |

Алгоритм вычисляет числовой хеш-код над входными значениями (с учётом ограничений, заданных параметром INDEXES) аналогично алгоритму расчёта числового хеш-кода. Полученный хеш-код используется для обращения к таблице подстановок в базе данных формата H2, в результате чего из записи с соответствующим номером извлекаются значения выходных полей. При отсутствии записи с нужным номером алгоритм сигнализирует ошибку маскирования, которая означает, что таблица подстановок была сформирована некорректно.

Примечание. Структура и состав стандартных таблиц подстановки ФИО, формируемых модулем генерации типовых справочников в составе конфигурационной программы «DsMask», приведена в разделе 5.3 *Генерация справочника подстановок для маскирования ФИО*.

Пример настройки функции маскирования с использованием этого алгоритма:

```
<function name="name-hash" type="HashLookup">
  <![CDATA[
    DB dict-names
    TABLE dict_fio
    ID id
    OUT nfull nlast nfirst nmiddle
    KEY default
    INDEXES 1
  ]]>
</function>
```

7.6 Подстановка по значениям атрибутов

Идентификатор алгоритма: KeyLookup

Поддержка итераций: нет (не требуется)

Параметры алгоритма приведены в таблице ниже.

| Параметр | Описание |
|----------|---|
| DB | Имя базы данных, содержащей таблицу подстановок. Обязательный параметр |
| TABLE | Имя таблицы подстановок. Обязательный параметр |
| KEYS | Имена полей таблицы подстановок, используемых как ключ для поиска. Обязательный параметр |
| OUT | Имена полей таблицы подстановок, значения которых будут результатом вызова алгоритма. Обязательный параметр |
| INDEXES | Список позиций в векторе входных значений, которые должны быть использованы для сопоставления с ключевыми полями при поиске. Нумерация позиций начинается с 1 (единицы). порядок указания индексов должен соответствовать порядку указания полей в параметре KEYS. Опциональный параметр, при его отсутствии ключевые поля, установленные в KEYS, сопоставляются со входными значениями |
| USER | Имя пользователя-владельца базы данных. Опциональный параметр, по умолчанию имя пользователя не устанавливается |

Алгоритм принимает на вход одно или несколько значений, которые используются как параметры для поиска записи в таблице подстановок, сопоставляемые с ключевыми полями этой таблицы.

Результат поиска – одно или несколько полей найденной записи.

При отсутствии подстановочной записи возвращаются пустые (NULL) значения.

Пример настройки функции маскирования с использованием этого алгоритма:

```
<function name="cid-mask" type="KeyLookup">
  <![CDATA[
    DB cidrepl
    TABLE cidrepl
    KEYS cid_src
    OUT cid_dst
    KEY default
    INDEXES 1
  ]]>
</function>
```

7.7 Вызов алгоритмов ODPP

Идентификатор алгоритма: ODPP

Поддержка итераций: нет (не может быть реализована)

Алгоритм осуществляет вызов провайдера маскирования через интерфейс Optim Data Privacy Providers (ODPP). Состав доступных операций маскирования и их параметры описаны в документации IBM InfoSphere Optim:

<https://www.ibm.com/docs/en/iotdm/11.3?topic=reference-optim-data-privacy-provider-library>

Алгоритм применяется только к единичным значениям, интерпретируемым как строки. Если при настройке правила маскирования в алгоритм подано сразу несколько значений, каждое из этих значений будет обработано независимо от других, формируя отдельное выходное значение.

Для установки значения ключа инициализации в параметрах ODPP-провайдера можно указать имя ключа в фигурных скобках, как показано в примерах ниже. При инициализации провайдера имя ключа будет заменено на фактическое значение, сопоставленное с ним.

Примеры функций маскирования с использованием алгоритмов ODPP:

```
<!-- маскирование ИНН с сохранением первых 4 разрядов -->
<function name="odpp-inn" type="ODPP">
<![CDATA[
    PRO=AFF,MTD=REP,ALGO=FPE,COPY=(1,4),LANG="ru",KEY="{default}"
]]>
</function>
```

```
<!-- маскирование СНИЛС -->
<function name="odpp-snils" type="ODPP">
<![CDATA[
    PRO=AFF,MTD=REP,ALGO=FPE,LANG="ru",KEY="{default}"
]]>
</function>
```

```
<!-- маскирование email -->
<function name="odpp-email" type="ODPP">
<![CDATA[
    PRO=EML,MTD=HASH,HASHDOM=REG,ALGO=SHA256
]]>
</function>
```

7.8 Строковые преобразования

Идентификатор алгоритма: StringOp

Поддержка итераций: нет (не требуется)

Алгоритм применяет заданные строковые операции ко входному значению, интерпретируемому как строка. Если в алгоритм передано несколько строк, каждая из них будет обработана независимо от других.

Каждая операция задаётся в настройках алгоритма как отдельная строка, содержащая идентификатор операции (обязательно) и аргументы операции (в зависимости от вида операции), отделённые друг от друга пробелами. Операции выполняются в том порядке, в котором они указаны в настройках алгоритма.

Если в аргументах операции требуется использование пробельных символов, они должны заключаться в символы кавычек ("). Если в значении аргумента требуется использование кавычек, они должны экранироваться символом обратного следа (\).

Поддерживаемые строковые операции:

1. Lower – перевод символов строки в нижний регистр;
2. Upper – перевод символов строки в верхний регистр;
3. Capitalize – каждое слово начинается с символа в верхнем регистре и продолжается символами в нижнем регистре;
4. Trim – удаление пробелов в начале и конце строки;
5. LTrim – удаление пробелов в начале строки;
6. RTrim – удаление пробелов в конце строки;
7. SpaceNorm – нормализация пробелов, т.е. удаление пробелов в начале и конце строки, с последующим преобразованием последовательности пробелов в единичный пробел, а также замена пробельных символов (символа табуляции, возврата каретки и аналогичных) на пробелы;
8. Translit – транслитерация кириллических букв на латинские буквы, исходя из особенностей произношения русского языка;
9. Replace – замена подстрок, заданных регулярным выражением, на указанное подстановочное значение.

Примеры функций маскирования с использованием алгоритма строковых преобразований:

```
<!-- нормализация ФИО -->
<function name="name-norm" type="StringOp">
<![CDATA[
    SpaceNorm
    Capitalize
]]>
</function>

<!-- удалить все нечисловые символы -->
<function name="remove-non-digits" type="StringOp">
<![CDATA[
    Replace "[^\d]" ""
]]>
</function>
```

7.9 Конкатенация строк

Идентификатор алгоритма: Concat

Поддержка итераций: нет (не требуется)

Параметры алгоритма приведены в таблице ниже.

| Параметр | Описание |
|-----------|---|
| SEPARATOR | Разделитель, используемый при склеивании строк. Опциональный параметр, при отсутствии используется единичный пробел (значение « »). |

Алгоритм осуществляет конкатенацию переданных аргументов, интерпретируемых как строки, в одну выходную строку. Между конкатенируемыми строками вставляется настраиваемая строка-разделитель. Значения NULL пропускаются (строка-разделитель не добавляется). Пустые входные строки обрабатываются аналогично непустым (не пропускаются).

Пример определения функции с использованием алгоритма конкатенации строк:

```
<function name="name-concat" type="Concat"/>
```

7.10 Разделение строк

Идентификатор алгоритма: Split

Поддержка итераций: нет (не требуется)

Параметры алгоритма приведены в таблице ниже.

| Параметр | Описание |
|----------|---|
| RX | Регулярное выражение, определяющее разделитель строк. Опциональный, по умолчанию используется значение «[\s\r\n]». |

Алгоритм разделения строк принимает одну входную строку и возвращает одну или несколько выходных строк, являющихся результатом разделения входной строки на части.

Передача сразу нескольких входных строк в алгоритм не допускается.

Примеры определения функций с использованием алгоритма разделения строк:

```
<!-- простой пример со стандартными разделителями -->
<function name="name-split" type="Split"/>
```

```
<!-- разделить строки на основе символа '@' -->
<function name="email-split" type="Split">
<![CDATA
  RX "@"
]]>
</function>
```

7.11 Подстановка символов

Идентификатор алгоритма: CharSubst

Поддержка итераций: нет (не требуется)

Параметры алгоритма приведены в таблице ниже.

| Параметр | Описание |
|----------|--|
| TABLE | Имя используемой алгоритмом таблицы подстановок символов. Обязательный параметр |

Алгоритм выполняет замену символов во входном значении, интерпретируемом как строка, на другие символы в соответствии с заданной таблицей подстановок. Имя таблицы подстановок используется для поиска и загрузки XML-документа, сопоставляющего входные и выходные символы алгоритма. При подстановке поддерживается работа с составными Unicode-последовательностями.

Алгоритм применяется для дополнительной трансляции кодировок в случае, когда языковые настройки исходной базы данных не позволяют корректно читать хранимые данные в кодировке Unicode.

Таблицы трансляции символов загружаются из файлов, размещённых в каталоге, заданном параметром «DictPath» оператора маскирования (см. раздел 4.4.3 *Настройки оператора маскирования «DsMask»*). Каждая таблица трансляции символов настраивается в виде файла в формате XML, имя которого формируется по следующему шаблону: «charz-ИмяТаблицы.xml», где ИмяТаблицы определяется параметром TABLE алгоритма.

Пример таблицы трансляции символов с прямым сопоставлением символов грузинского алфавита в кодировке Unicode и нестандартной изменённой кодировке Latin-1:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<char-trans-table name='ge-x2u'>
  <char-map
    dst='აბგდეზზთიკლმნოპჟრსტუფჯღყზხც'
    src='ÀÁÂÃÄÅÆÈÉÊËÌÍÎÏÐÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäå'
  />
</char-trans-table>
```

Пример таблицы трансляции символов с использованием диапазонов значений (технически полностью эквивалентна предыдущему примеру):

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<char-trans-table name='ge-x2u'>
  <char-range size="7" src-char="À" src-hex="c0" dst-char="ა" dst-hex="10d0"/>
  <char-range size="6" src-char="È" src-hex="c8" dst-char="თ" dst-hex="10d7"/>
  <char-range size="2" src-char="İ" src-hex="cf" dst-char="ო" dst-hex="10dd"/>
  <char-range size="3" src-char="Ò" src-hex="d2" dst-char="რ" dst-hex="10e0"/>
  <char-range size="1" src-char="Õ" src-hex="d5" dst-char="ჟ" dst-hex="10df"/>
  <char-range size="12" src-char="Ö" src-hex="d6" dst-char="უ" dst-hex="10e3"/>
  <char-range size="2" src-char="ä" src-hex="e3" dst-char="ჯ" dst-hex="10ef"/>
</char-trans-table>
```

Пример определения функции с использованием алгоритма трансляции символов:

```
<function name="ge-x2u" type="CharSubst">
<![CDATA
    TABLE ge-x2u
]]>
</function>
```

7.12 Проекция

Идентификатор алгоритма: Project

Поддержка итераций: нет (не требуется)

Алгоритм копирует набор входных значений в набор выходных значений, сохраняя последовательность переданных значений.

Алгоритм используется для того, чтобы сформировать необходимый конечный результат правила маскирования путём копирования значений, полученных ранее на различных шагах этого правила.

Примеры определения функции с использованием алгоритма проекции:

```
<function name="project" type="Project"/>
```

8 Первоначальная установка прототипа решения

Первоначальная установка прототипа решения включает в себя следующие действия:

1. Установка серверных компонентов IBM Information Server
2. Установка Windows-клиента IBM Information Server
3. Корректировка настроек операционной системы
4. Корректировка настроек серверных компонентов IBM Information Server
5. Донастройка IBM Information Analyzer
6. Установка необходимых драйверов СУБД
7. Установка дополнительных компонентов решения

Для планирования и выполнения работ по установке необходимо знакомство с платформой IBM Information Server, знание порядка установки её компонентов, вариантов развёртывания.

Приведённая ниже информация не является пошаговой инструкцией по установке, и концентрируется преимущественно на специфике решения по маскированию данных.

8.1 Установка серверных компонентов IBM Information Server

Установка серверных компонентов IBM Information Server осуществляется с использованием штатной программы инсталляции из дистрибутива продукта. Необходим административный доступ в систему (локальный администратор в среде Microsoft Windows, пользователь «root» в среде Linux и AIX).

Дистрибутив продукта и лицензионный ключ к нему (license specification) необходимо распаковать в один и тот же каталог, после чего будет возможен запуск программы установки.

Доступ к интерактивному интерфейсу программы установки осуществляется через Web-браузер (рекомендовано использование браузера Google Chrome и производных от него).

Необходимый состав устанавливаемых компонентов включает в себя:

- репозиторий метаданных (Repository Tier) – базу данных для ведения метаданных;
- Information Governance Catalog (Services Tier);
- Information Analyzer (Services Tier, Engine Tier);
- DataStage / QualityStage (Engine Tier);
- опционально – DataStage Flow Designer (Services Tier).

При использовании СУБД IBM Db2 для развёртывания репозитория метаданных рекомендуется установить СУБД заранее, используя наиболее актуальный (поддерживаемый для работы Information Server) пакет обновлений СУБД. Рекомендуется установить СУБД на отдельный сервер, чтобы исключить конкуренцию за вычислительные ресурсы между СУБД и другими службами.

Не рекомендуется устанавливать компоненты, относящиеся к Engine Tier и к Services Tier, на одном сервере, поскольку такая конфигурация не оптимальна с точки зрения использования лицензируемых вычислительных мощностей.

При установке Services Tier рекомендуется выбрать вариант сервера приложений WebSphere Liberty, поскольку он отличается несколько меньшим потреблением ресурсов и проще в администрировании.

Перед установкой в среде Windows может потребоваться корректировка настроек операционной системы. Программа установки обнаружит требуемые изменения при проверке соблюдения системных требований, и отобразит ссылку на статью с инструкциями.

8.2 Установка Windows-клиента IBM Information Server

Установка Windows-клиента IBM Information Server осуществляется с использованием штатной программы инсталляции из дистрибутива продукта. Необходим административный доступ в систему (локальный администратор в среде Microsoft Windows).

Состав устанавливаемых компонентов должен соответствовать составу серверных компонентов (необходимы, как минимум, клиент и административный клиент DataStage, а также консоль сервисов IBM Information Server).

При установке Windows-клиента потребуются реквизиты для подключения к серверным компонентам (имя сервера и номер порта, а также административный логин и пароль), поэтому установку клиента следует производить после установки сервера IBM Information Server.

8.3 Корректировка настроек операционной системы

Необходимо обеспечить синхронизацию системного времени на всех узлах, входящих в состав системы маскирования данных. Рассинхронизация системного времени может приводить к различным трудно диагностируемым ошибкам, включая сбои аутентификации и авторизации, некорректную работу средств диагностики, протоколирования и построения отчётов, а также разнообразные «плавающие» сбои в работе сервисов.

В среде Linux и AIX для пользователей root, dsadm и db2inst1 (владелец экземпляра Db2) необходимо установить лимит файловых дескрипторов в значение 16384 или выше (как «мягкий», так и «жесткий» лимиты). Проконтролировать установленный лимит можно вызовом «ulimit -a».

В современных дистрибутивах Linux соответствующие настройки могут быть выполнены путём создания файла /etc/security/limits.d/99-iis.conf со следующим содержанием:

| | | | |
|-----------|------|--------|-------|
| root | soft | nofile | 16384 |
| root | hard | nofile | 16384 |
| dsadm | soft | nofile | 16384 |
| dsadm | hard | nofile | 16384 |
| @db2iadml | soft | nofile | 32768 |
| @db2iadml | hard | nofile | 32768 |

В примере выше настройки для пользователя db2inst1 сделаны через указание имени группы системных администраторов Db2 для экземпляра по умолчанию (db2iadml). Если реальное имя группы отличается от указанного, настройку лимита файловых дескрипторов необходимо сделать для реального имени группы.

8.4 Корректировка настроек серверных компонентов IBM Information Server

После изменения настроек перечисленных ниже параметров необходимо перезапустить сервисы IBM Information Server (см. Раздел 8.10 *Запуск и остановка компонентов IBM Information Server*).

8.4.1 Настройки IBM Db2

При использовании IBM Db2 для организации репозитория метаданных рекомендовано установить переменную реестра Db2 DB2_FMP_COMM_HEAPSZ в значение 131072, а затем перезапустить сервис Db2. Это можно сделать путем выполнения от имени пользователя-владельца экземпляра Db2 (по умолчанию – пользователя db2inst1) следующих команд:

```
db2set DB2_FMP_COMM_HEAPSZ=131072
db2stop force
db2start
```

8.4.2 Настройки IBM DataStage

В среде Linux необходимо дополнить файл <IISDIR>/Server/DSEngine/dsenv следующими настройками (можно указать сразу после строки, содержащей команду «set +u»):

```
TZ=/etc/localtime
export TZ
ulimit -n 10240
```

При отсутствии корректно настроенной сети IPv6 рекомендуется отключить использование протокола IPv6 заданиями IBM DataStage. Для этого необходимо установить переменную окружения APT_USE_IPV4 в значение 1. Для ОС Windows это можно сделать на общесистемном уровне, для ОС Linux и AIX – путём настройки переменной на уровне параметров каждого из

проектов DataStage (см. подробнее <https://www.ibm.com/support/pages/new-informationserver-91-install-check-configuration-fails>).

8.4.3 Включение операционного мониторинга IBM DataStage

В файле настроек операционного мониторинга (файл Server/DSODB/DSODBConfig.cfg) необходимо активировать сбор данных о выполнении заданий DataStage путём установки параметра DSODBON в значение «1». Сбор данных о выполнении заданий DataStage используется для работы Web-интерфейса Operations Console, а также требуется для построения отчётности по результатам запусков заданий маскирования.

8.5 Донастройка IBM Information Analyzer

Для начала работы с IBM Information Analyzer необходимо выполнить дополнительные действия по настройке, а именно:

- а) открыть Web-интерфейс инструмента Metadata Asset Manager, и выполнить следующие операции:
 1. выбрать «Import», затем «New Import Area»;
 2. ввести название области импорта «IADB»;
 3. выбрать тип коннектора для рабочей БД Information Analyzer (обычно – Db2);
 4. указать параметры соединения (хост, порт, логин и пароль), при этом для Db2 должен использоваться пользователь с логином «iauser», созданный при установке серверных компонентов;
 5. отключить импорт всех видов активов (требуется только описание подключения);
 6. выполнить экспресс-импорт с публикацией результатов в репозиторий;
- б) открыть Windows-приложение Information Server Console, и выполнить следующие действия:
 1. возможно, запуск приложения успешно произойдёт только со второго раза из-за известной ошибки;
 2. выбрать в левом верхнем кругу «Home» (кружок с пиктограммой), затем «Configuration», затем «Analysis Settings»;
 3. опционально – указать вариант «Static engine credentials», введя логин «dsadm» и соответствующий ему (указанный при установке) пароль;
 4. выбрать операцию «Validate Analysis Database connection», и убедиться в отсутствии возвращаемых ошибок;
 5. выбрать операцию «Validate Analysis Engine settings», и убедиться в отсутствии возвращаемых ошибок.

До успешного выполнения перечисленных выше действий работа с инструментом Information Analyzer невозможна.

Для повышения стабильности работы IBM Information Analyzer с большими объёмами данных (только для платформ Linux и AIX) рекомендовано выполнить следующие изменения стандартных настроек:

```
cd /opt/IBM/InformationServer/ASBServer/bin
```

```
./iisAdmin.sh -set -key com.ibm.iis.ia.engine.javaStage.heapSize -value 4096  
./iisAdmin.sh -set -key com.ibm.iis.ia.jdbc.connector.heapSize -value 4096
```

Приведённый выше пример команд устанавливает предельный размер оперативной памяти для использования отдельными заданиями Information Analyzer в значение 4 Гбайт. Для использования указанных настроек необходимо наличие достаточного объёма оперативной памяти на сервере (например, при одновременном конкурентном выполнении 4 заданий Information Analyzer с указанными настройками необходимо наличие не менее 24 Гбайт оперативной памяти, с учётом потребностей других сервисов).

На платформе Microsoft Windows при работе IBM Information Analyzer используется 32-битная виртуальная машина Java, для которой допустимое значение указанных выше параметров обычно не превышает 768 Мбайт.

8.6 Настройка подавления предупреждений Information Analyzer (опционально)

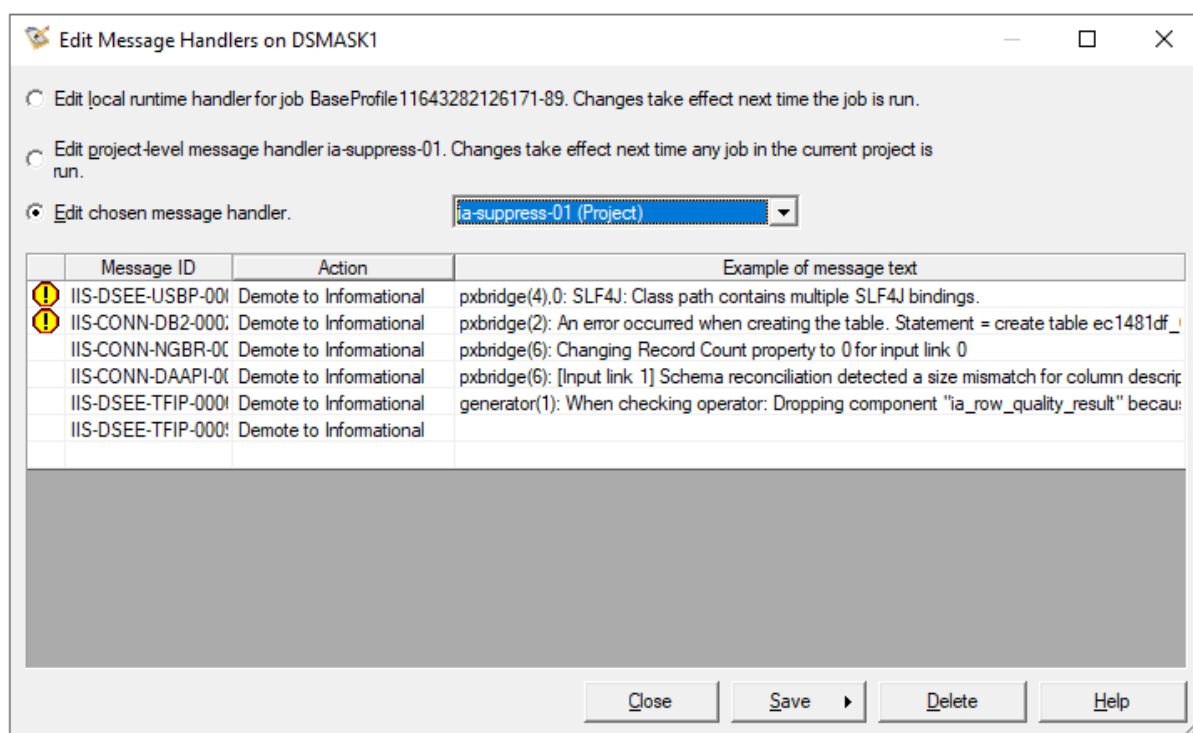
При нормальной работе инструмента IBM Information Analyzer запускаемые им задания DataStage генерируют некоторое количество предупреждений, которые по умолчанию отображаются в операционной консоли IBM Information Server.

Для устранения отображения «штатных» предупреждений рекомендуется настроить их подавление. Для настройки подавления необходимо открыть инструмент «Director Client», и выбрать в нём опцию «Tools / Message Handler Management». Далее требуется ввести идентификаторы подавляемых сообщений:

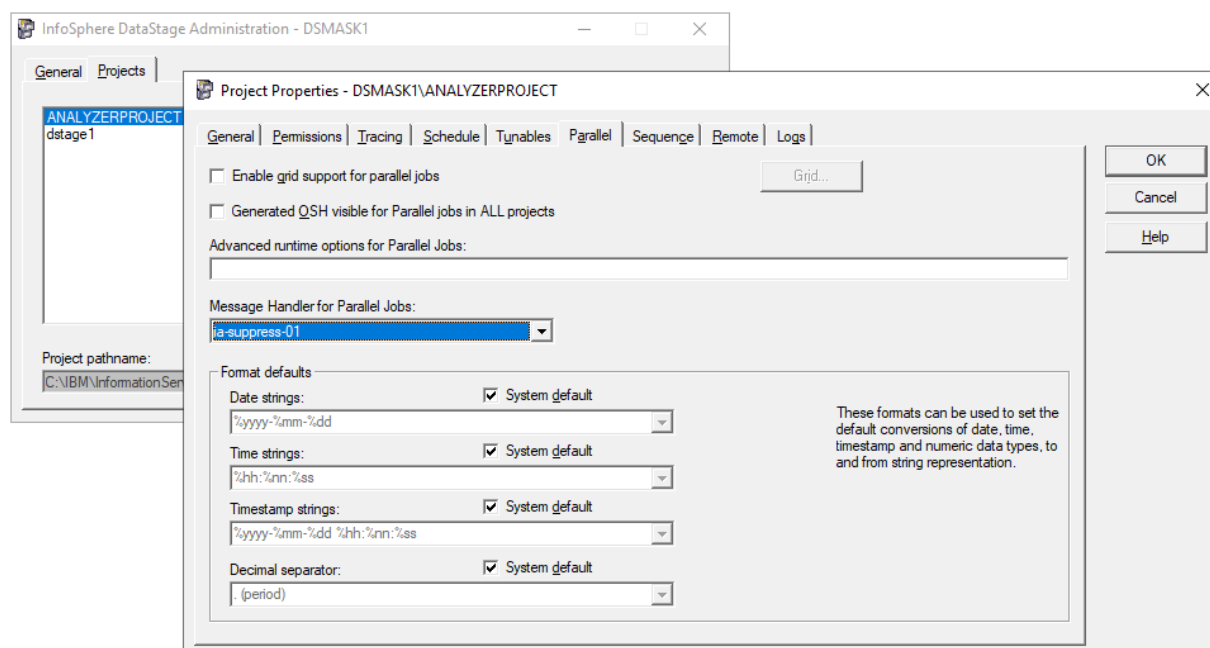
- IIS-CONN-DB2-00027
- IIS-CONN-NGBR-00072
- IIS-CONN-DAAPI-00398
- IIS-DSEE-TFIP-00063
- IIS-DSEE-TFIP-00095
- IIS-DSEE-USBP-00002

Для каждого сообщения выбирается один из режимов «Demote to Informational» (рекомендуется к использованию) либо «Suppress from Logs». Результат настройки необходимо сохранить в виде объекта типа «Message handler», дав ему новое имя (например, «ia-suppress-01»).

Пример выполненных настроек можно увидеть на снимке экрана ниже.



Далее необходимо открыть инструмент «Administrator Client», и установить для проекта ANALYZERPROJECT обработчик сообщений, обеспечивающий подавление выбранного набора предупреждений (опция «Parallel / Message Handler for Parallel Jobs»), как показано на снимке экрана ниже.



8.7 Установка необходимых драйверов СУБД

Для работы высокопроизводительных коннекторов (например, для СУБД Oracle Database и Microsoft SQL Server) на сервере или серверах, где установлен Engine Tier, требуется наличие установленной и настроенной клиентской части соответствующих СУБД.

Для работы с СУБД Oracle Database технически достаточно наличие установленного Oracle Instant Client, с указанием переменных окружения ORACLE_HOME, TNS_ADMIN и NLS_LANG в файле Server/DSEngine/dsenv.

Для использования JDBC-драйверов соответствующие библиотеки должны быть размещены в файловой системе сервера или серверов, где установлен Engine Tier. JDBC-драйверы при этом должны быть зарегистрированы в файле Server/DSEngine/isjdbc.config, пример содержимого приведён ниже:

```
CLASSPATH=/jdbc/h2-2.0.206.jar;/jdbc/mssql-jdbc-8.4.0.jre8.jar;  
/jdbc/postgresql-42.2.14.jar; /jdbc/ojdbc8.jar;/jdbc/db2jcc4.jar  
CLASS_NAMES=org.h2.Driver;com.microsoft.sqlserver.jdbc.SQLServerDriver;  
org.postgresql.Driver;oracle.jdbc.driver.OracleDriver;  
com.ibm.db2.jcc.DB2Driver
```

Значения переменных CLASSPATH и CLASS_NAMES должны быть указаны в одну строку, без пробелов, для разделения элементов используется символ ';' (точка с запятой).

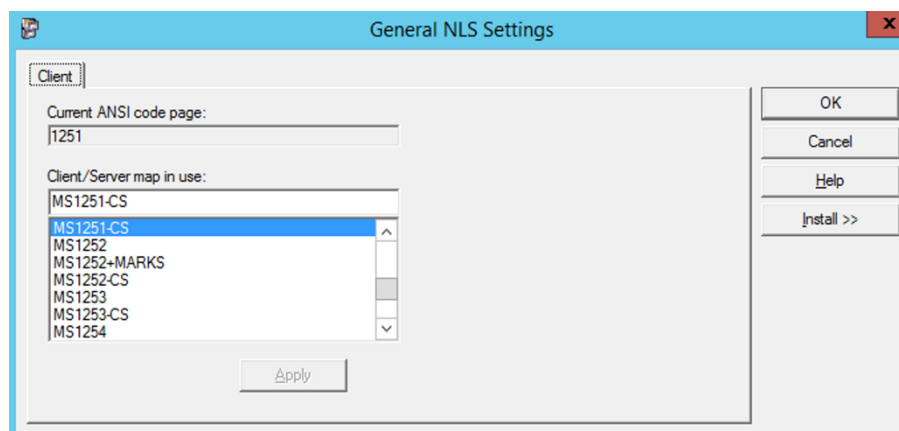
Более подробная информация по настройке взаимодействия с источниками данных, включая обеспечение поддержки Kerberos-аутентификации и защиты трафика на базе протокола TLS, приведена в официальной документации на IBM Information Server.

8.8 Настройка проектов IBM DataStage

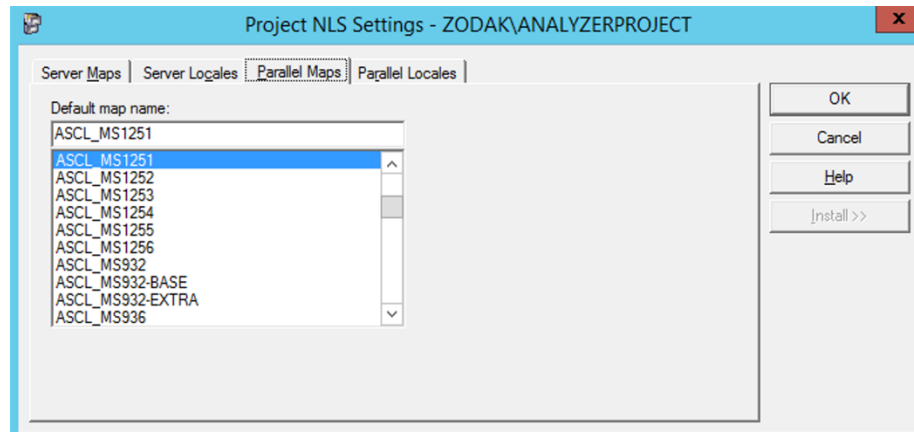
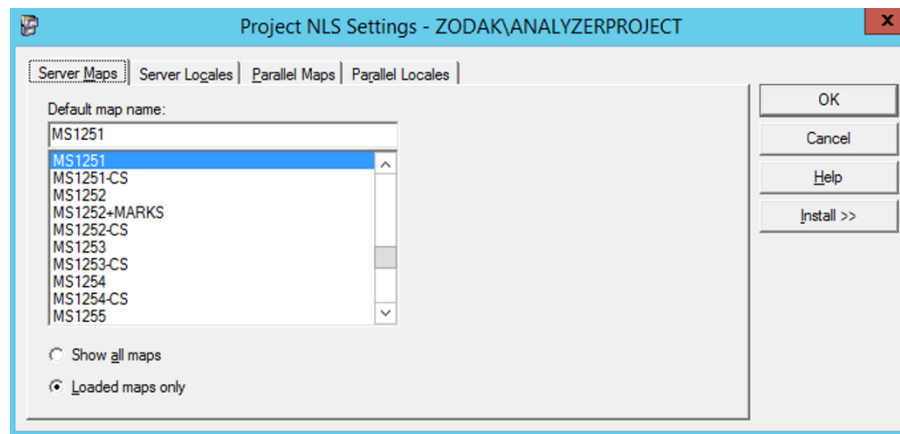
Перед подготовкой заданий маскирования необходимо выполнить ряд настроек IBM DataStage в среде инструмента «Administrator Client». Рисунки в текущем разделе являются снимками экрана, полученными при работе с этим инструментом.

Необходимо обеспечить корректную работу с текстовыми символами в используемых системах-источниками кодировках. Наиболее универсальным вариантом является настройка для работы в кодировке UTF-8. При работе с данными, использующими смесь латинских и кириллических символов, часто используют кодировку CP1251.

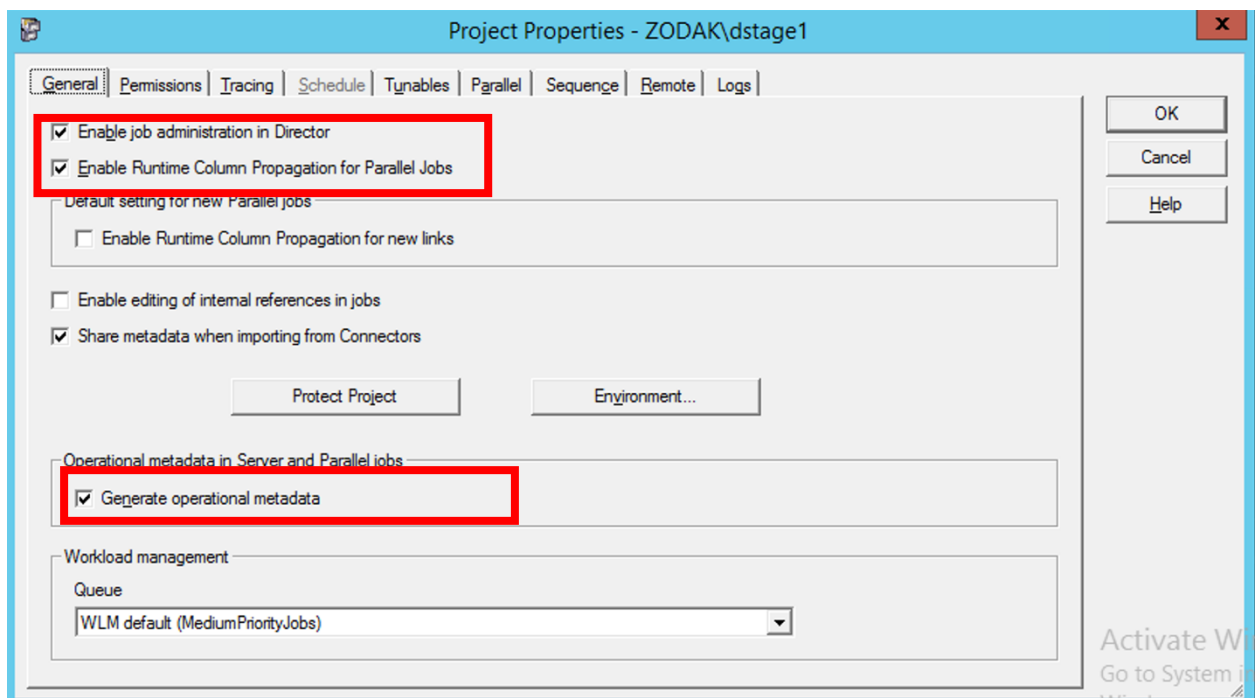
Пример настроек локализации для кодировки CP1251 на общесистемном уровне приведён на рисунке ниже.



Пример настроек локализации для кодировки CP1251 на уровне проекта (настраивается для проекта ANALYZERPROJECT и проекта «по умолчанию» для пользовательских заданий dstage1).



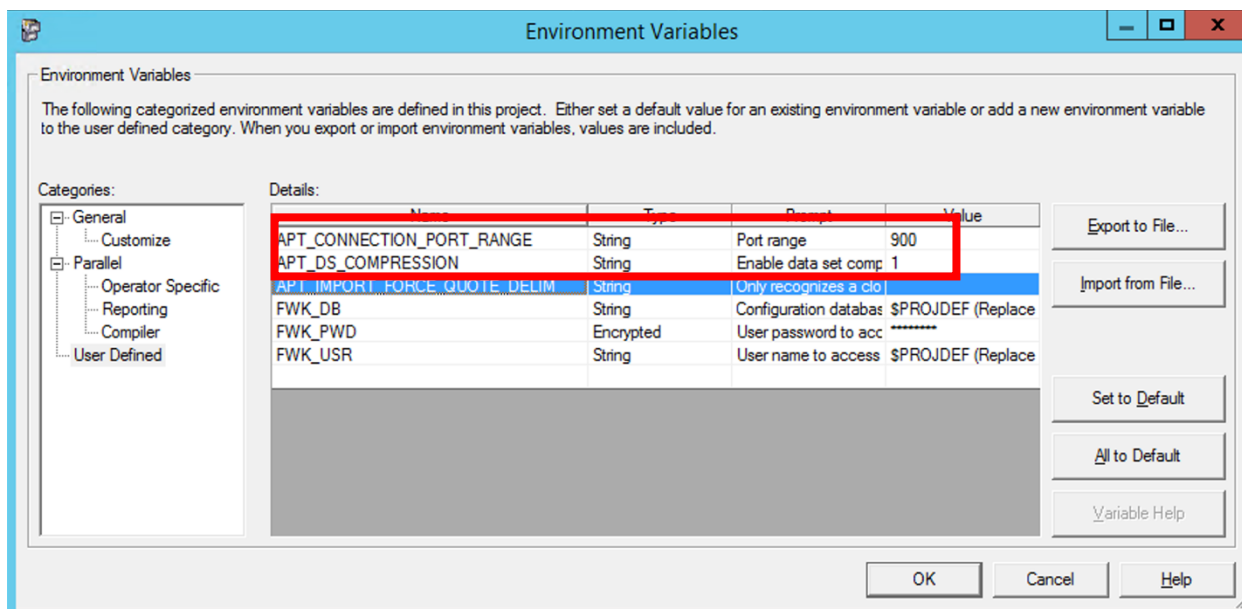
На уровне проекта «dstage1» необходимо настроить возможность использования функции Runtime Column Propagation. Дополнительно рекомендуется разрешить управление заданиями через инструмент Director Client, и включить по умолчанию генерацию операционных метаданных (см. рисунок ниже).



Для ускорения запуска заданий DataStage рекомендовано установить на уровне проекта переменную окружения APT_CONNECTION_PORT_RANGE в значение от 900 до 1000 (использование значений больше 1000 требует корректировки других связанных параметров).

Для использования механизма сжатия данных при хранении структурированных наборов данных DataStage следует на уровне проекта установить переменную окружения APT_DS_COMPRESSION в значение 1.

Пример выполненных настроек показан на рисунке ниже.



8.9 Установка дополнительных компонентов решения

Дополнительные компоненты решения поставляются в виде набора архивных файлов:

- dsmask-jconf-bin.zip – содержит конфигурационную программу «DsMask», примеры правил маскирования, а также генератор справочников для подстановок ФИО;
- dsmask-jmask-bin.zip – содержит оператор маскирования «DsMask», примеры заданий DataStage для решения задач маскирования, пример скрипта для запуска пакетного маскирования (MaskBatcher), а также примеры запросов для построения операционных отчётов по результатам работы заданий маскирования;
- ia-bundle-ru-bin.zip – содержит примеры пользовательских классов данных;
- dsmask-uniq-bin.zip – содержит сервис контроля уникальности подстановок.

Установка дополнительных компонентов заключается в распаковке архивных файлов на сервере (или серверах) системы маскирования с последующей настройкой дополнительных компонентов.

8.9.1 Распаковка архивных файлов

Архив с оператором маскирования «DsMask» необходимо распаковать на серверах с установленным Engine Tier, при этом путь установки должен совпадать на всех серверах (стандартное значение – /opt/IBM/Masker).

Конфигурационную программу «DsMask» необходимо установить на том сервере (тех серверах), где планируется осуществлять её запуск. Рекомендовано наличие доступа к общему каталогу (размещённому в общей сетевой либо кластерной файловой системе) для сохранения конфигурационных баз данных с профилями маскирования, а также баз данных со справочниками подстановок.

Архив с примерами пользовательских классов данных необходимо распаковать на одном из серверов, где установлены компоненты Services Tier, поскольку для регистрации классов данных используется входящая в состав этого слоя утилита IAAdmin.sh. Кроме того, распакованный архив должен быть доступен на каждом из серверов Engine Tier для доступа к справочникам (см. описание настройки переменной OPTIM_DCS_DICT в следующем подразделе).

Внешний сервис контроля уникальности подстановок необходимо распаковать в отдельный каталог, имеющий достаточный объём дискового пространства для размещения рабочих файлов (от 10 Гбайт) и высокую производительность ввода-вывода (рекомендовано использование SSD). Оптимально разместить сервис контроля уникальности на выделенном сервере для исключения конкуренции с другими сервисами за вычислительные ресурсы.

8.9.2 Регистрация примеров определения классов данных

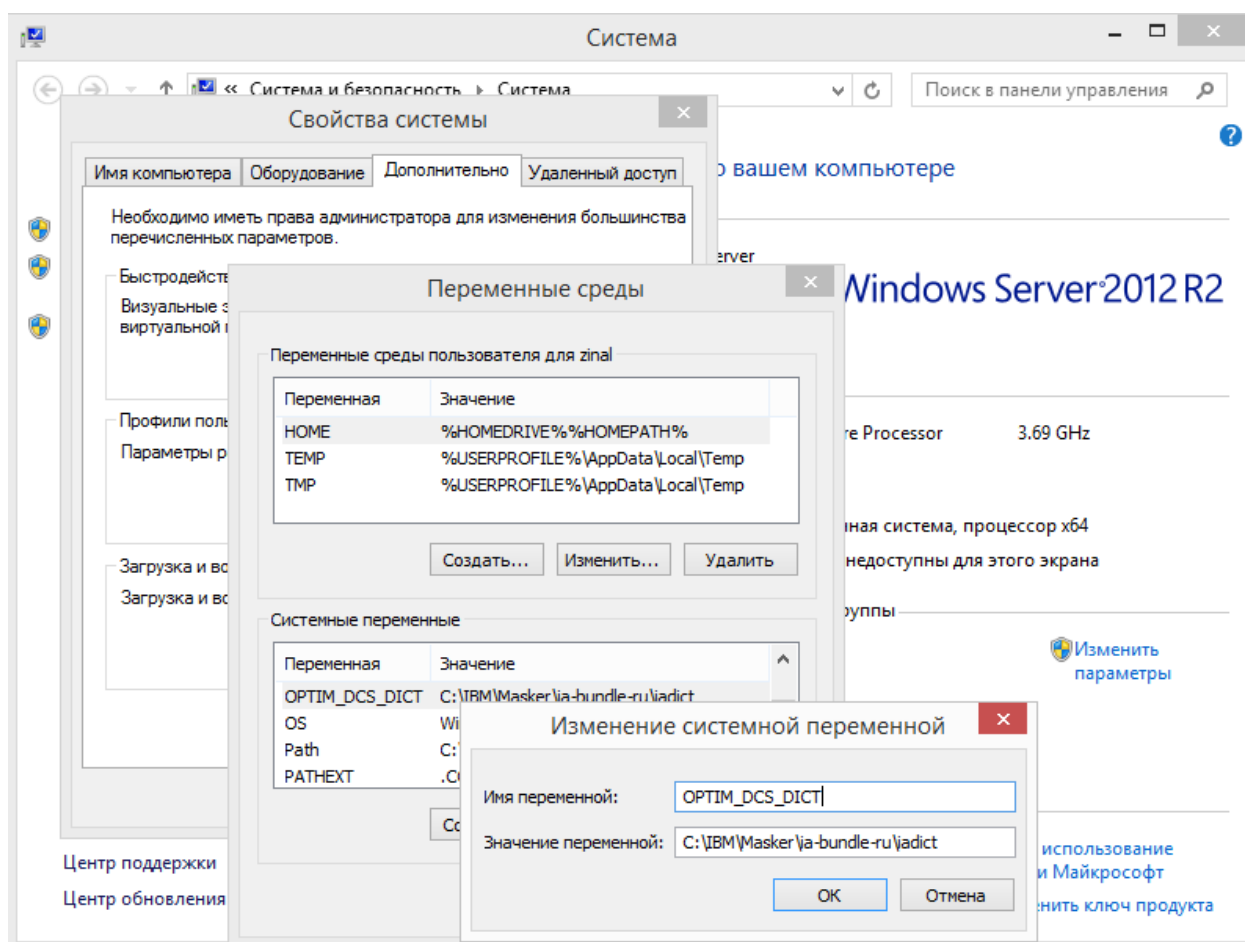
После распаковки необходимо выполнить регистрацию примеров определений классов данных. Для этого используется специальный скрипт, который представлен в том же архивном файле (см. файлы dcs-install.cmd и dcs-install.sh, в зависимости от платформы). Необходимые настройки (путь к инсталляции IBM Information Server, URL для доступа к сервисам, логин и пароль пользователя), которые необходимо скорректировать перед запуском скрипта регистрации классов данных, устанавливаются в отдельном файле (см. файлы dcs-config.cmd и dcs-config.sh).

Для установки пути к каталогу с файлами справочников, используемых примерами определений классов данных, необходимо установить переменную окружения OPTIM_DCS_DICT.

Установка переменной OPTIM_DCS_DICT на общесистемном уровне для платформы Linux может быть обеспечена редактированием файла /etc/profile, либо (предпочтительно) созданием дополнительного файла с необходимой настройкой в каталоге /etc/profile.d. Пример настройки для платформы Linux:

```
OPTIM_DCS_DICT=/opt/IBM/Masker/ia-bundle-ru/iadict
export OPTIM_DCS_DICT
```

Для платформы Windows установку переменной окружения необходимо сделать на общесистемном уровне (см. скриншот ниже).



8.9.3 Регистрация оператора маскирования

Оператор маскирования «DsMask» необходимо зарегистрировать в файле операторов Java (файл Server/DSEngine/JavaStage.config). Пример строки для регистрации оператора:

```
classpath:com.ibm.ds.mask.DsMask = /opt/IBM/Masker/dsmask-jmask/lib/*
```

Аналогичный пример для платформы Microsoft Windows:

```
classpath:com.ibm.ds.mask.DsMask = C:\\IBM\\Masker\\dsmask-jmask\\lib\\*
```

Перезапуск сервисов после изменения содержимого файла JavaStage.config не требуется.

8.9.4 Настройка сервиса контроля уникальности

Сервис контроля уникальности необходимо настроить путём редактирования параметров в файле uniq-service-config.xml.

Описание параметров сервиса контроля уникальности приведено в таблице ниже.

| Параметр | Описание |
|------------|---|
| workDir | Каталог для размещения рабочих файлов. Полный либо относительный путь к каталогу |
| shardCount | Количество рабочих файлов. Рекомендовано использовать значение по умолчанию (100) |

| Параметр | Описание |
|------------------|---|
| cacheSizeMB | Объём кэш-области в оперативной памяти на каждый рабочий файл, в мегабайтах. Определяет требования сервиса к доступной оперативной памяти, увеличение этого параметра может потребовать корректировок скрипта запуска сервиса |
| lazyCommitPeriod | Периодичности принудительного подтверждения транзакций, в секундах. Рекомендовано использовать значение по умолчанию (60) |
| svcPort | Номер порта для приёма клиентских соединений. По умолчанию используется значение 27500. |
| svcHost | Адрес, на котором сервис должен принимать соединения. При использовании значения по умолчанию (127.0.0.1) сервис должен быть запущен на узле Engine Tier |
| svcSecret | Ключ для подключения клиентов к сервису |

Для работы сервиса контроля уникальности необходимо создать отдельную технологическую учётную запись, сделать её владельцем распакованных файлов и разрешить использование необходимых системных ресурсов: оперативной памяти (по умолчанию не менее 10 Гбайт), процессорного времени, файловых дескрипторов (по умолчанию не менее 300), открытие сетевого порта. Также для сервиса контроля уникальности необходимо обеспечить автоматический либо ручной запуск, что может быть реализовано через средства управления системными сервисами (под Linux – systemd).

8.9.5 Проверка работоспособности

Для проверки работоспособности установленных компонентов можно использовать примеры заданий DataStage и правил маскирования, поставляемых с решением. Проверка осуществляется следующим образом:

1. В тестовой исходной БД выбирается таблица для маскирования.
2. Метаданные таблицы импортируются средствами Metadata Asset Manager в репозиторий метаданных.
3. Поля выбранной таблицы размечаются классами данных, для которых есть примеры правил маскирования.
4. Подготавливаются значения наборов параметров «DbParams» и «DbOutParams» для входной и выходной баз данных.
5. Выполняется тестовый запуск задания маскирования и контроль его результатов.

После успешного выполнения тестового запуска задания маскирования решение готово к применению.

8.10 Запуск и остановка компонентов IBM Information Server

8.10.1 Общие сведения о порядке запуска и остановки компонентов

В штатном режиме работы запуск компонентов IBM Information Server осуществляется автоматически при загрузке соответствующего сервера или серверов, а остановка происходит при завершении работы или перезагрузке.

Процедура ручного перезапуска для платформ Linux/UNIX описана в следующем документе:
<https://www.ibm.com/support/pages/how-manually-stopstart-ibm-infosphere-information-server-services-unixlinux>

При необходимости основные операции ручного перезапуска могут быть собраны в скрипты ручной остановки и запуска, которые будут платформенно-зависимыми (в частности, состав вызываемых команд отличается для Linux и AIX).

8.10.2 Отключение автоматического запуска компонентов (опционально)

Часто на стендах разработки и сопровождения конфигураций маскирования данных оказывается желательным ручное управление запуском и остановкой сервисов IBM Information Server.

Для отключения автоматического запуска сервисов IBM Information Server при загрузке компьютера под управлением Microsoft Windows может использоваться скрипт «iis-services-manual.cmd», включённый в состав дистрибутива модуля dsmask-jmask. После перевода сервисов в режим ручного запуска с помощью этого скрипта, запуск и остановка всех сервисов IBM Information Server может выполняться скриптами «iis-services-start.cmd» и «iis-services-stop.cmd».

В случае установки сервера IBM Information Server на платформе Microsoft Windows запуск компонента DataStage Flow Designer осуществляется при входе пользователя в систему. Отключить автоматический запуск этого сервиса можно путём удаления значения «IBM_DataStage_Flow_Designer» из следующего ключа реестра Windows:
«HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run».