SRS: Интеграция входящей корреспонденции в Directum RX через REST API

1. Введение

1.1 Назначение

Данный документ описывает функциональные и нефункциональные требования к системе, обеспечивающей автоматическую регистрацию входящей корреспонденции во внутренней системе электронного документооборота Directum RX через REST API.

1.2 Задачи проекта

- Автоматизировать приём и регистрацию входящих писем и вложений в RX;
- Исключить дублирование данных и документов;
- Сократить нагрузку на регистраторов и снизить число ошибок;
- Обеспечить юридическую значимость и целостность хранимых писем.

1.3 Область применения

Система будет использоваться для интеграции с корпоративной почтой, CRM и другими внешними источниками входящих писем. Основная цель — снизить ручной труд по заводу документов и сократить ошибки при регистрации корреспонденции.

1.4 Основные заинтересованные стороны

Роль	Интересы и цели
Руководство	Снижение операционных затрат
IT-служба	Надёжная и масштабируемая интеграция
Секретариат/архив	Упрощение процессов регистрации
Бизнес-пользователи	Быстрый доступ к документам

2. Общие сведения о системе

2.1 Подключение к Directum RX API

- Подключение к REST API осуществляется по HTTPS по адресу: `https://<host>/api/`
- Авторизация производится с использованием Bearer-токена, который должен быть получен заранее через OAuth 2.0 или выдан администратором системы.
- Все запросы должны содержать заголовки:

- `Authorization: Bearer <access_token>`
- `Content-Type: application/json`
- `Accept: application/json`

- Рекомендуется использовать отдельного сервисного пользователя с ограниченными правами только на создание входящих документов.

2.2 Акторы

- **Интеграционный сервис** (внешняя система-источник).
- **REST API Directum RX** (приемник данных).
- **Администратор системы** (настройка прав, справочников).

2.3 Описание работы

1. Внешняя система генерирует JSON-запрос с данными письма.
2. Через защищённый REST API создаётся карточка документа.
3. Вложения загружаются в виде PDF.
4. RX обрабатывает и сохраняет документ.

3. Функциональные требования

1. Автоматический импорт входящих писем в Directum RX. Система должна автоматически получать входящие письма из заданных почтовых ящиков и регистрировать их как объекты входящей корреспонденции в Directum RX.
2. Распознавание и привязка по признакам (email-идентификаторы, шаблоны, адреса). При импорте письма система должна проверять, существует ли уже связанный документ в Directum RX (например, по Thread-ID, Message-ID, теме письма или адресату) и, в случае нахождения, добавлять письмо в соответствующее дело/цепочку сообщений.
3. Разделение писем по типам корреспонденции. В зависимости от домена отправителя, ключевых слов или адреса назначения, письмо должно автоматически классифицироваться как входящее обращение или спам.
4. Сохранение вложений письма. Все вложения писем должны сохраняться в системе в оригинальном формате и быть доступны для просмотра, скачивания и прикрепления к другим документам внутри СЭД.
5. Создание задания исполнителю. При регистрации письма система должна автоматически создавать задание (например, “Рассмотреть входящее письмо”) для сотрудника, указанного по правилам маршрутизации или в карточке организации/контакта.
6. Интерфейс мониторинга и логирования обработки писем. Должен быть реализован интерфейс (раздел в Directum RX) для отображения: состояния синхронизации с почтой, количества обработанных и необработанных писем, ошибок и исключений при обработке, логов событий по каждому письму.
7. Поддержка ручной дообработки. Пользователь СЭД должен иметь возможность вручную: создать документ на основе письма (в случае ошибки автосоздания), изменить получателя задания, изменить привязку письма к документу или делу, повторно запустить обработку письма.

4. Нефункциональные требования

1. Система должна быть масштабируема — возможность подключения дополнительных почтовых ящиков без остановки работы.
2. Интеграция должна работать в режиме 24/7 без необходимости ручного запуска.
3. Поддержка обработки не менее 10 000 писем в сутки.
4. Обработка одного письма не должна превышать 3 секунд при размере вложений до 5 МБ.
5. Хранение логов обработки — не менее 30 дней.
6. Все ошибки должны логироваться и быть доступны в интерфейсе администратора.

5. Диаграммы

5.1 BPMN-диаграмма

Описание: Диаграмма показывает основной поток обработки входящего письма — от получения письма до его регистрации в системе RX.

Сценарий: автоматическая проверка писем, удаление спама, регистрация уникальных документов и уведомление исполнителя.

5.2 UML-диаграмма компонентов

Описание: Диаграмма демонстрирует взаимодействие между внешними системами, интеграционным сервисом и RX API.

6. JSON-схемы

API должен принимать JSON с полями: `subject`, `documentKind`, `sender`, `receivedDate`, `attachments`, `register`, `responsible`.

Описание полей JSON-запроса:

Поле	Тип	Обязательное	Назначение
`subject`	string	да	Тема/краткое описание входящего письма.
`documentKind`	string	да	Вид документа (например, "Входящее письмо").
`sender`	object	да	Объект с данными отправителя.
`sender.name`	string	да	Имя или название организации-отправителя.
`sender.email`	string	нет	Email отправителя.
`receivedDate`	string (ISO8601)	да	Дата и время получения письма.
`attachments`	array	нет	Массив вложений в формате PDF.

Поле	Тип	Обязательное	Назначение
`fileName`	string	да (внутри attachments)	Имя файла вложения.
`contentPDF`	string	да (внутри attachments)	PDF-файл, закодированный в base64.
`register`	string	нет	Название регистра для регистрации документа.
`responsible`	string	нет	Логин/имя ответственного сотрудника.

Обработка ошибок:

- `400` — ошибка валидации
- `401` — неавторизованный доступ
- `403` — недостаточно прав
- `409` — дублирование

7. Тестирование

7.1 Успешные кейсы и коды ответов

№	Название	Описание запроса	Ожидаемый результат
1	Успешное создание документа	Корректный JSON с одним вложением	`201 Created`, возвращается ID документа
2	Успешно без вложений	JSON без поля `attachments`	`201 Created`, документ создан
3	Повторная отправка с тем же ID	Повторный запрос с тем же `externalId`	`200 OK`, возвращается существующий ID
4	Проверка существующего документа	GET-запрос по ID	`200 OK`, возвращается JSON документа
5	Обновление токена	Обновление access token через OAuth	`200 OK`, возвращается новый токен
6	Вложение с кириллическим именем	PDF с именем файла на русском языке	`201 Created`, файл корректно сохраняется

7.2 Проблемные кейсы и коды ответов

№	Название	Описание запроса	Ожидаемый результат
1	Пропущено поле `subject`	JSON без `subject`	`400 Bad Request`

№	Название	Описание запроса	Ожидаемый результат
2	Неверный формат даты	`"receivedDate": "01-01-2025"`	`400 Bad Request`
3	Невалидный email	`"email": "not-an-email"`	`400 Bad Request`
4	Неизвестный `documentKind`	`"documentKind": "неизвестный"`	`400 Bad Request`
5	Пустой `contentPDF`	Вложение с пустым контентом	`400 Bad Request`
6	Отсутствует токен авторизации	Без заголовка `Authorization`	`401 Unauthorized`
7	Недостаточно прав	Пользователь без прав на создание документов	`403 Forbidden`
8	Повторная отправка с тем же ID	Повторный запрос с тем же `externalId`	`409 Conflict`
9	Неверный Content-Type	`Content-Type: text/plain` вместо JSON	`415 Unsupported Media Type`
10	Превышен размер вложения	Вложение больше 10 МБ	`413 Payload Too Large`
11	Лишнее поле в JSON	`"unexpected": "value"`	`400 Bad Request`
12	`attachments` не массив	`"attachments": {}`	`400 Bad Request`
13	`sender` не объект	`"sender": "ООО Альфа"`	`400 Bad Request`
14	Пустой JSON	`{}`	`400 Bad Request`

8. Примеры API-запросов

Ниже приведены примеры API-запросов, которые демонстрируют создание входящего документа (POST) и получение информации о нём (GET). Эти запросы можно использовать как образец при ручном тестировании через Swagger или автоматизации в Postman.

POST /api/documents/incoming

Описание: Создаёт новый входящий документ в Directum RX с указанными метаданными и вложениями.

```
POST /api/documents/incoming HTTP/1.1
Host: rx.example.com
Authorization: Bearer <access_token>
Content-Type: application/json

{
  "subject": "Письмо от клиента по договору №123",
  "documentKind": "Входящее письмо",
```

```
  "sender": {
    "name": "ООО Бета",
    "email": "contact@beta.ru"
  },
  "receivedDate": "2025-06-04T13:00:00",
  "attachments": [
    {
      "fileName": "contract_123.pdf",
      "contentPDF": "<base64_pdf_data>"
    }
  ],
  "register": "Входящий реестр",
  "responsible": "petrov.p"
}
```

GET /api/documents/incoming/{id}

Описание: Возвращает информацию о ранее созданном входящем документе по его уникальному идентификатору `id`.

```
GET /api/documents/incoming/12345 HTTP/1.1
Host: rx.example.com
Authorization: Bearer <access_token>
Accept: application/json
```

Ответ:

```
{
  "id": "12345",
  "subject": "Письмо от клиента по договору №123",
  "documentKind": "Входящее письмо",
  "receivedDate": "2025-06-04T13:00:00",
  "sender": {
    "name": "ООО Бета",
    "email": "contact@beta.ru"
  },
  "attachments": [
    {
      "fileName": "contract_123.pdf"
    }
  ],
  "register": "Входящий реестр",
  "responsible": "petrov.p"
}
```

---

9. Ответственные лица (RACI)

Роль	Ответственность
Аналитик	Описание требований (R)
Разработчик	Реализация API-интеграции (A)
Тестировщик	Проверка работы (C)
Администратор RX	Настройка справочников (C)