Хост PgDocuments

Обзор

Хост `PgDocuments` предоставляет набор сервисов для управления информационными объектами (документами) в системе MorphCluster. Основная функциональность включает:

- Управление структурой типов документов (DocKind) — свойства, статусы, действия, права доступа
- Работа с документами — создание, чтение, обновление, удаление
- Импорт/экспорт метаданных и данных в XML/JSON
- Генерация и синхронизация таблиц документов в PostgreSQL
- Выполнение бизнес-логики через функции и действия

Структура сервисов

Клиенты (ServiceNats)

Клиенты **не описываются** в данной документации, но они используются сервисами для межсервисного взаимодействия:

- **Eventer** — отправка событий, управление пользовательскими соединениями
- **SysProcesses** — управление фоновыми бизнес-процессами

Основные сервисы

1. DockindStructure

Ответственность: Управление структурой типов документов (DocKind): статусы, свойства, функции, действия, права доступа, правила именования.

Зависимости: `QueriesHelper`, `OraLongTransactions`, `SysProcesses`

События:

- `onChanged` — генерируется при любом изменении структуры типа документа

Основные методы:

Метод	Описание

`getInfo({ docKind })`	Получить базовую информацию о типе документа (ID, описание)
`setMain({ oldDocKind, newDocKind, description, parentId })`	Обновить основную информацию (только описание и родительскую папку)
`getStatuses({ docKind, docKindId })`	Получить список статусов (событий) типа документа
`setStatuses({ docKind, deleteIds, statuses, newOrder })`	Обновить статусы: удалить, изменить, добавить, переупорядочить
`getProps({ docKind, docKindId })`	Получить список свойств типа документа
`setProps({ docKind, props, deleteIds })`	Обновить свойства: удалить, изменить, добавить
`setProperty({ docKindId, property })`	Внутренний метод для сохранения одного свойства с его статусными масками и ссылками
`getFunctions({ docKind, docKindId })`	Получить функции, привязанные к типу документа
`setFunctions({ docKind, funcs, deleteIds })`	Обновить функции (DB или CSP) с их привязкой к статусам и свойствам
`setActions({ docKind, deleteIds, actions })`	Обновить действия (переходы между статусами)
`setNames({ docKind, names })`	Обновить правила именования документов
`setPermissions({ docKind, roles, deleteRoleIds })`	Установить права доступа для ролей (создание, видимость/редактирование статусов и свойств)
`getIdByName({ docKind })`	Получить ID типа документа по имени
`getNameById({ docKindId })`	Получить имя типа документа по ID
`updateDockindNames({ dockindId })`	Отложенное обновление описаний всех документов типа
`createDbFunction({ schema, name })`	Создать пустую PL/pgSQL-функцию, если она не существует

---

2. Documents

Ответственность: Низкоуровневая работа с документами — чтение/запись значений, информация о документе, проверка обязательных полей.

Зависимости: `QueriesHelper`, `DockindCache`

Основные методы:

Метод	Описание
`getInfo({ documentId })`	Получить основную информацию о документе (тип, статус, дата, создатель)
`getDescr({ documentId })`	Получить описание документа (сгенерированное по правилам именования)

Метод	Описание
`getValues({ dockind, documentId, propNames })`	Получить значения указанных свойств документа (поддерживает простые и ссылочные поля)
`setValues({ dockind, documentId, values })`	Установить значения свойств документа в рамках транзакции
`insertRefs({ dockind, documentId, propName, values })`	Добавить ссылки на другие документы в множественное ссылочное поле
`deleteRefs({ dockind, documentId, propName, values })`	Удалить указанные ссылки из множественного ссылочного поля
`checkNull({ dockind, documentId })`	Проверить, заполнены ли обязательные для текущего статуса поля

---

3. DocumentLists

Ответственность: Поиск и выборка списков документов по значениям свойств.

Зависимости: `QueriesHelper`, `DockindCache`, `Documents`

Основные методы:

Метод	Описание
`listIdByProps({ dockind, values, statuses, notStatuses, count })`	Найти ID документов по значениям свойств (AND-условия)
`listPropsByProps({ dockind, props, filter, statuses, notStatuses, count })`	Выбрать документы с указанными полями для вывода
`listByUniqueFields({ dockind, values })`	Найти документы по уникальным полям (автоматически определяются по конфигурации)
`getUniqueCollisions({ dockind, documentId })`	Найти документы с конфликтами по уникальным полям (исключая переданный ID)

---

4. DocumentFuncs

Ответственность: Выполнение бизнес-функций, привязанных к статусам и свойствам документов.

Зависимости: `QueriesHelper`, `DockindCache`, `Documents`, `GlobalServices`, `Eventer`

Основные методы:

Метод	Описание
`processStatus({ dockind, documentId, statusName, params, session, postprocess })`	Выполнить все функции, привязанные к указанному статусу (pre- или post-process)

Метод	Описание
`processProperty({ dockind, documentId, propName, session })`	Выполнить функции, привязанные к изменению свойства
`processAnykindFuncs({ dockind, documentId, params, session })`	Выполнить глобальные (anykind) функции для документа
`runFunc({ func, dockind, documentId, params, session, trxId })`	Выполнить конкретную функцию (DB или CSP)

Поддерживаемые типы функций:

- **DB** — вызов хранимой процедуры PostgreSQL (формат `schema.function`)
- **CSP** — вызов метода удалённого сервиса (формат `ServiceName.method`)

---

5. DocCardValues

Ответственность: Высокоуровневая работа со значениями документов в контексте карточки (UI-транзакции).

Зависимости: `QueriesHelper`, `DockindCache`, `Documents`, `DocumentFuncs`, `DocumentLists`

Основные методы:

Метод	Описание
`getValDisplay({ documentId, propName })`	Получить отображаемое значение поля (например, имя справочника вместо ID)
`getValues({ documentId, propName })`	Получить массив значений простого поля
`setValues({ cardId, documentId, propName, value/values })`	Установить значения простого поля в рамках карточной транзакции
`getRefs({ documentId, propName })`	Получить список связанных документов для ссылочного поля
`setRefs({ cardId, documentId, propName, refDocumentIds })`	Заменить весь список связанных документов
`clearRefs({ cardId, documentId, propName })`	Очистить ссылочное поле
`insertRefs({ cardId, documentId, propName, refDocumentIds })`	Добавить ссылки в множественное поле
`deleteRefs({ cardId, documentId, propName, refDocumentIds })`	Удалить указанные ссылки
`createRefDoc({ cardId, documentId, propName, parentId })`	Создать новый документ, привязанный к ссылочному полю
`deleteRefDocs({ cardId, documentId, propName, refDocumentIds })`	Удалить связанные документы (полное удаление)
`commit({ cardId })`	Зафиксировать транзакцию карточки (проверка обязательных полей и уникальности)
`rollback({ cardId })`	Откатить транзакцию карточки

---

6. DocCardActions

Ответственность: Выполнение действий (переходов между статусами) с документами.

Зависимости: `QueriesHelper`, `DockindCache`, `Documents`, `DocumentFuncs`, `DocCardValues`, `SysProcesses`, `DocumentLists`

Основные методы:

Метод	Описание
`runActions({ dockind, documentIds, actionId, actParams, cardId, session })`	Выполнить действие над одним или несколькими документами
`runAction({ dockind, documentId, action, actParams, cardId, session })`	Внутренний метод выполнения одного действия (проверка статуса, смена статуса, вызов функций)
`runGlobalAction({ action, actParams, cardId, session })`	Выполнить глобальное действие (без привязки к документу)
`create({ dockind, parentId, classifPropId, classifId, session })`	Создать новый документ с опциональным родителем и классификатором
`deleteDocs({ dockind, documentIds, session })`	Пометить документы как удалённые и запустить фоновую очистку
`processDelete({ dockind, documentId, session })`	Отложенная постобработка удаления (вызов функций статуса "deleted", физическое удаление)

Логика действия:

1. Проверка допустимости текущего статуса (`FROM_STATUS_IDS`)
2. (Опционально) Смена статуса на `TO_STATUS_ID` с вызовом pre-process функций
3. (Опционально) Вызов функции, привязанной к действию
4. Запуск post-process функций для нового статуса
5. Запуск глобальных (anykind) функций

---

7. DocTables

Ответственность: Генерация и синхронизация структур таблиц документов в PostgreSQL для быстрого поиска и отчетов.

Зависимости: `QueriesHelper`, `OraLongTransactions`, `DockindCache`

Основные методы:

Метод	Описание
`recreate({ docKind })`	Полностью пересоздать таблицу, представление и процедуру синхронизации для типа, затем синхронизировать все данные

Метод	Описание
`recreateStruct({ docKind })`	Пересоздать только структуру таблицы (без представления и данных)
`recreateView({ docKind })`	Пересоздать представление для типа документа (объединяет основную таблицу и словари)
`recreateAll({})`	Пересоздать структуры для всех типов документов (последовательно)
`createSyncProcedure({ dockind })`	Сгенерировать PL/pgSQL-процедуру синхронизации для типа
`documentSync({ docKind, documentId })`	Синхронизировать один документ с его таблицей
`documentSyncKind({ docKind })`	Синхронизировать все документы указанного типа (пакетная обработка)

Архитектура таблиц:

- Основная таблица: `doc_tables.<dockind>_t` — содержит single-свойства
- Дочерние таблицы: `doc_tables.<dockind>_t_<prop>` — для multi-свойств
- Представление: `doc_tables.<dockind>_v` — объединяет данные для удобного чтения

---

8. DockindImporter

Ответственность: Импорт метаданных из XML и JSON в базу данных.

Зависимости: `DockindStructure`, `QueriesHelper`, `DocumentsHelper`, `OraLongTransactions`

Основные методы:

Метод	Описание
`importXml({ root, vocabs, types })`	Импорт структуры из XML (словари + типы документов)
`importJson({ root, data })`	Импорт структуры из JSON

Внутренние классы:

- **XmlImporter** — парсинг XML, создание/обновление словарей, типов, статусов, свойств, действий, имён, функций
- **JsonImporter** — аналогичный функционал для JSON (более современный формат)

Поддерживаемые сущности при импорте (JSON):

- Словари (`vocabs`)
- Типы документов (`name`, `descr`)
- Статусы (`statuses` с цветами, функциями, правами)

- Свойства (`props` с типом, мульти, уникальностью, статусными масками, ссылками, правами, функциями)
- Правила именования (`names`)
- Действия (`actions` с условиями, переходами, правами, функциями)
- Права доступа (`permissions` на уровне ролей)

---

9. DockindExporter

Ответственность: Экспорт метаданных типа документа в JSON.

Зависимости: `DockindStructure`, `QueriesHelper`, `DocumentsHelper`, `OraLongTransactions`

Основные методы:

Метод	Описание
`exportJson({ dockind })`	Экспортировать полную структуру типа документа в JSON

Внутренний класс: `JsonExporter`

Экспортируемые данные:

- Основная информация (`name`, `descr`)
- Используемые словари (`vocabs` со значениями)
- Статусы (`statuses` с цветами, правами, функциями)
- Свойства (`props` с типами, параметрами, статусными масками, ссылками, правами, функциями)
- Правила именования (`names`)
- Действия (`actions` с условиями, переходами, правами, функциями)

---

10. DocDataImporter

Ответственность: Импорт данных документов из внешних источников (XML).

Зависимости: `QueriesHelper`, `DocumentsHelper`, `DockindCache`, `Documents`

Основные методы:

Метод	Описание
`importXml({ dataXml, uniqueProps })`	Импортировать данные документов из XML

Логика импорта:

1. Парсинг XML, извлечение структуры документов
2. Для каждого документа:

- Поиск существующего документа по уникальным полям (если указаны)
- Если найден — применение правил слияния (добавление, перезапись, стирание)
- Если не найден — создание нового документа

3. Заполнение свойств документа (включая ссылки на подчинённые документы)
4. Установка статуса

Правила слияния:

- `0` — значение не меняется
- `2` — добавление новых значений в множество
- `3` — перезапись значения
- `4` — стирание значения

Вспомогательные модули

db-functions.mjs

Содержит определения SQL-функций для `csp_documents` (схема CSP-документов). Эти функции используются сервисами для работы с базой данных:

- `get_all_dockinds` — получить все типы документов
- `get_document_info` — получить информацию о документе
- `get_dockind_id_by_name` / `get_dockind_name_by_id` — преобразование ID/имени
- `get_dockind_props` — свойства типа документа
- `get_dockind_prop_events` — статусные маски свойств
- `get_document_values` / `get_document_refs` — чтение значений и ссылок
- и другие

DocFunctionsInstaller

Наследуется от `DbFunctionsInstaller` и устанавливает функции из `db-functions.mjs` при старте сервиса.

Revision #1
Created 26 June 2026 15:30:36 by Admin
Updated 26 June 2026 15:31:29 by Admin