УТВЕРЖДАЮ
Генеральный директор
ООО «Кейсистемс»
_____ А. А. Матросов
«___» _____ 2022 г.

ПРОГРАММНЫЙ КОМПЛЕКС «ИНТЕГРАЦИЯ-КС»

ВЕРСИЯ 3.9

Руководство пользователя

Разработка сервиса обмена сообщениями RMS (Remote Messaging Service)

ЛИСТ УТВЕРЖДЕНИЯ

Р.КС. 09010-01 34 05-ЛУ

СОГЛАСОВАНО
Заместитель генерального директора
ООО «Кейсистемс»
_____ С. В. Панов
«___» _____ 2022 г.
Руководитель ДСР
_____ Д. Г. Пахомов
«___» _____ 2022 г.

2022

Литера А

Подп и дата	
Инв.N дубл	
Взам.инв.N	
Подп и дата	
Инв.N подл	

УТВЕРЖДЕНО
Р.КС. 09010-01 34 05-ЛУ

информационные технологии КЕЙСИСТЕМС

ПРОГРАММНЫЙ КОМПЛЕКС «ИНТЕГРАЦИЯ-КС»

ВЕРСИЯ 3.9

Руководство пользователя

Разработка сервиса обмена сообщениями RMS (Remote Messaging Service)

Р.КС. 09010-01 34 05

Листов 10

2022

Литера А

Подп и дата	
Инв.N дубл	
Взам.инв.N	
Подп и дата	
Инв.N подл	

АННОТАЦИЯ

Настоящий документ является частью руководства пользователя программного комплекса «Интеграция-КС» версии 3.9 от 01.03.2022 г. и содержит описание операций по разработке сервиса обмена сообщениями RMS (Remote Messaging Service).

Руководство состоит их двух разделов:

- Описание операций.
- Рекомендации по освоению.

Раздел *«Описание операций»* содержит описание всех выполняемых функций, задач, описание операций по администрированию сервиса обмена сообщениями.

Раздел *«Рекомендации по* освоению*»* содержит рекомендации и разъяснения по использованию сервиса обмена сообщениями пользователем.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящее руководство пользователя содержит описание разработки сервиса обмена сообщениями программного комплекса «Интеграция-КС» (далее – «программный комплекс»).

Условные обозначения

В документе используются следующие условные обозначения:

	Уведомление	– Важные сведения о влиянии текущих действий пользователя на выполнение других функций, задач программного комплекса.
	Предупреждение	– Важные сведения о возможных негативных последствиях действий пользователя.
	Предостережение	– Критически важные сведения, пренебрежение которыми может привести к ошибкам.
	Замечание	– Полезные дополнительные сведения, советы, общеизвестные факты и выводы.
	[Выполнить]	– Функциональные экранные кнопки.
	<F1>	– Клавиши клавиатуры.
	«Чек»	– Наименования объектов обработки (режимов).
	Статус	– Названия элементов пользовательского интерфейса.
	ОКНА => НАВИГАТОР	– Навигация по пунктам меню и режимам.
	п. 2.1.1 *рисунок 5*	– Ссылки на структурные элементы, рисунки, таблицы текущего документа, ссылки на другие документы.

1. ОПИСАНИЕ ОПЕРАЦИЙ

Сервис состоит из трех проектов:

Keysystems.RemoteMessaging

Содержит описания, структуры данных, адаптеры, которые используются сервисом и могут быть использованы в клиентском ПО.

Keysystems.RemoteMessaging.Service

Классы, применение которых характерно только для сервиса.

Keysystems.RMS

Реализация Web-приложения RMS-сервиса.

Использование сторонних библиотек:

AutoMapper – преобразование объектов друг в друга;
(https://github.com/AutoMapper/AutoMapper)

ChilkatDotNet2 – работа с эл.почтой;
(http://www.chilkatsoft.com)

NHibernate – ORM для доступа к БД;
(http://nhforge.org)

FluentNHibernate – надстройка над NHibernate;
(https://github.com/jagregory/fluent-nhibernate)

Iesi.Collections – коллекции для NHibernate;
(https://github.com/nhibernate/iesi.collections)

Newtonsoft.Json – Json сериализация;
(http://json.codeplex.com)

NLog – файловый логер;
(http://nlog-project.org)

Ionic DotNetZipLib – библиотека для работы с zip архивами;
(http://dotnetzip.codeplex.com)

ASP.NET MVC3 – расширение MS ASP.NET MVC архитектуры 3-й версии;
(http://www.asp.net/mvc/mvc3)

ExtJs 3.4 – JavaScript компоненты;
(http://www.sencha.com/products/extjs3/)

1.1. Сервис

Програмная модель RMS-сервиса имеет структуру в виде дерева, корнем которого является класс **MessageService**. Он включает в себя модуль конфигурации (**ServiceConfig**), подсистему логирования (**ServiceLogger**), авторизации (**AuthManager**), менеджер каналов (**ChannelManager**) и др. Они все имеют ссылку на сервис. Благодаря этому, из любой части приложения можно получить доступ, например, к настройкам сервиса.

MessageService имеет статическое свойство **DefaultInstance**, в котором хранится экземпляр сервиса, создаваемый при запуске приложения. При создании экземпляра **MessageService** -а, в его конструктор передается строковый идентификатор, который фиксируется в служебной БД. По нему сервис и его БД однозначно связываются. Этот идентификатор должен быть уникальным и постоянным для создаваемого экземпляра сервиса.

Класс **ServiceInfo** описывает характеристики, текущее состояние и параметры сервиса. Некоторая информация о сервисе запоминается в его БД, другая – нужна только в runtime.

После создания экземпляра сервиса, он конфигурируется. Существует класс **ConfigFileSettings**, который берет настройки из файлов **Rms.config** и **Web.config**. При

конфигурировании указывается путь, где будут располагаться файлы журнала логирования, а также задается реализация модуля кэширования - **ServiceWebCache.** Если ядро сервиса будет использовано вне Web-приложения, то необходимо реализовать соответствующие интерфейсы: **IConfigSettings**, **IServiceCache**, которые будут специфичны для конкретного сценария работы.

WebApplication – реализация сценария использования сервиса в виде Web-приложения. **WebApplication** наследуется от **HttpApplication**. Запуск Web-приложения происходит при первом к нему обращении по http запросу.

Сервис стартует вызовом метода **Start().** При старте сервиса проверяется подключение к его служебной БД и ее структура таблиц. Далее активируется менеджер каналов (**ChannelManager**). Вне зависимости от того произошла ли ошибка на данном этапе, Web-приложение все равно запускается, но ошибка фиксируется и затем используется как сигнал о нерабочем состоянии сервиса. Это позволяет зайти на страницу администрирования и получить информацию об ошибке для последующего ее устранения.

При активации менеджера каналов, список каналов считывается из БД, после чего создаются их экземпляры в памяти, которые храняться в коллекции каналов. Если канал доступен (**Enabled**) и имеет установленное свойство автозапуска (**Autorun**), он стартует. Последующее обращение и доступ к каналам происходит через **ChannelManager**. Он управляет их поиском, созданием, удалением.

Для обработки поступающих http запросов реализованы конечные точки. admin.ashx – реализация интерфейса администратора сервиса, service.ashx – Web API точка доступа для различных клиентов.

1.2. Каналы

Канал сообщений определяется его типом, который зависит от вида источника данных, к которому он подключен. Каналы наследуются от абстрактного класса **WebappRutimeBase**, который задает базовый функционал. В зависимости от типа канала реализованы соответствующие классы наследники. Методы канала можно разделить на несколько групп:

- Информация: получение, сохранение и обновление информации о канале.
- Управление: инициализация, запуск и остановка канала.
- Диагностика: проверка подключения, Ping канала.
- Работа с командами;
- Работа с сообщениями;
- Работа с контактами;
- Работа с журналом.

1.3. Сообщение

Класс **Message** описывает сообщение. Его тело **MessageBody** и контент **MessageContent** хранятся отдельно от сообщения, а сообщение имеет лишь их описания **MessageBodyInfo** и **MessageContentInfo**. Это позволяет отделить «легкое» описание от «тяжелого» контента. Чтобы можно было описать любое сообщение, оно имеет коллекцию дополнительных свойств **MessageProperty.**

Статус сообщения показывает состояние сообщения на определенный момент времени. Класс **MessageStatus** содержит все возможные значения статуса.

Для хранения сообщения в виде отдельного файла разработан **ZipMessageContainer**. В него добавляется описательная часть сообщения, его тело и контент:

- message.json (message.xml или message.bin) – описание сообщения;
- message.body – файл тела сообщения (может отсутствовать);
- Contents – каталог для хранения вложений;

В контейнере может храниться только одно сообщение. Т.к. **ZipMessageContainer** – это обычный zip архив, то его можно открыть любым архиватором.

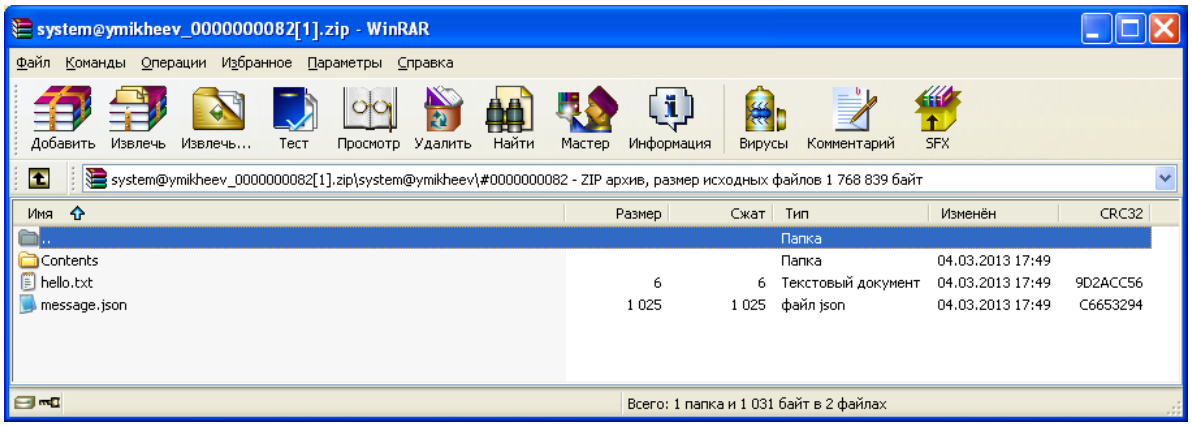

Рисунок 1. Контейнер хранения сообщения

1.4. Адаптеры

Каналы подключаются к своим источникам данных с помощью адаптеров.

MessageDataAdapter – адаптер для подключения к БД POSTGRESQL. Данный адаптер реализован на базе технологии ORM NHibernate с применением FluentNHibernate. Т.е. взаимодействие с БД происходит не SQL командами в явном виде, а с помощью сущностей объектов доступа к данным. Т.е. такие классы как ChannelInfo, ServiceInfo, Message и др. – это и есть такие сущности. Их признаком является свойство **LINK**, которое представляет собой первичный ключ в таблице БД.

RemoteChannelAdapter – адаптер для подключения к каналу, расположенному на другом RMS-сервисе. Он использует **HttpClient**, который может посылать **GET** и **POST** запросы на удаленный сервис. Некоторые данные передаются в Json формате («application/json»), некоторые в виде QueryString, а некоторые кодируются как «application/x-www-form-urlencoded».

SmevChannelAdapter – адаптер на базе **HttpClient,** посылает POST запросы с Content-Type: «text/xml» и передает в них XML SOAP сообщения.

2. РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В документе используются следующие сокращения:

ПК – программный комплекс.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Номер версии	Примечание	Дата	ФИО исполнителя
01	Начальная версия	02.03.2016	Пахомов Д.Г.