

CM-Sochi-Server   
и CM-AdminGUI

Архитектура

Содержание

[1 Введение 4](#_Toc414277469)

[2 Назначение документа 4](#_Toc414277470)

[3 Соглашения 4](#_Toc414277471)

[4 Назначение подсистемы 5](#_Toc414277472)

[5 Концепция 7](#_Toc414277473)

[6 Логические Компоненты 8](#_Toc414277474)

[6.1 EntityMapper 8](#_Toc414277475)

[6.1.1 Кэширование сущностей 9](#_Toc414277476)

[6.1.2 Идентификаторы сущностей (UNID) 10](#_Toc414277477)

[6.2 CollectionProvider 10](#_Toc414277478)

[6.3 UI-Collections 11](#_Toc414277479)

[6.4 JS 11](#_Toc414277480)

[6.5 NamedCode 11](#_Toc414277481)

[6.6 Notification 11](#_Toc414277482)

[6.7 PKD 12](#_Toc414277483)

[6.8 FT-Search 12](#_Toc414277484)

[6.9 SoService 12](#_Toc414277485)

[6.10 Reporter 12](#_Toc414277486)

[6.11 AgentRunner 12](#_Toc414277487)

[6.12 Data-CSV 12](#_Toc414277488)

[6.13 AclRules 12](#_Toc414277489)

[6.14 Authentification 13](#_Toc414277490)

[6.15 CM-Agents 13](#_Toc414277491)

[6.16 Notes-adapter 13](#_Toc414277492)

[6.17 Cmj-beans 13](#_Toc414277493)

[6.18 Storage-SPI 13](#_Toc414277494)

[6.19 CM-AdminGUI 13](#_Toc414277495)

[7 Состав артефактов поставки 14](#_Toc414277496)

[7.1 Зависимости 17](#_Toc414277497)

[8 Схема развертывания 17](#_Toc414277498)

CM-sochi-Server

Архитектура. Назначение. Применение.

# Введение

Данный документ описывает подсистему CM-Sochi-Server корпоративной информационной системы CompanyMedia®[[1]](#footnote-1): архитектуру, назначение, возможности и применение. Подсистема позволяет получить конфигурацию CompanyMedia® на открытом ПО, базирующуюся на платформе ActiveFrame 5®[[2]](#footnote-2), полностью на стеке Java EE технологий и РСУБД.

Это руководство является частью комплекта документации по архитектуре CompanyMedia®.

# Назначение документа

Документ предназначен для формирования высокоуровневого представления о подсистеме CM-Sochi-Server, её назначении, внутренней структуре, месте в общем программном комплексе CompanyMedia®, взаимодействии с другими подсистемами. Документ ориентирован на персонал, выполняющий развертывание системы, поддержку её работоспособности и функциональное развитие.

# Соглашения

|  |  |
| --- | --- |
| **Сокращение** | **Расшифровка** |
| SSRV | Подсистема CM-Sochi-Server |
| AF5 | Платформа ActiveFrame 5 |
| CM | CompanyMedia |
| РСУБД | Реляционная система управления базами данных |
| ПО | Программное обеспечение |
| АРМ | Автоматизированное рабочее место |
| ДОП | Доменный объект платформы AF5. Является элементарной единицей хранения. |

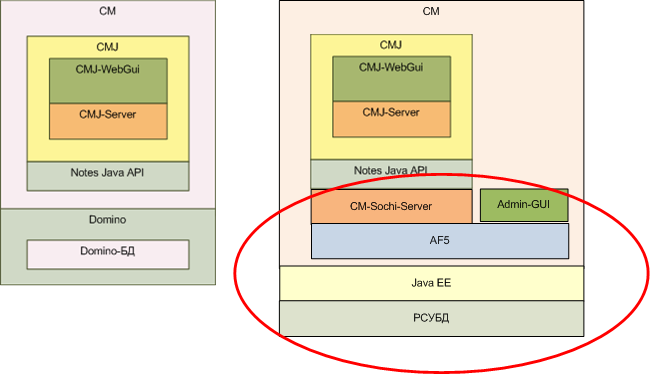
# Назначение подсистемы

Предыдущие версии CompanyMedia® имели конфигурацию компонент, основывающихся на подсистеме хранения, обеспечиваемой проприетарной системой IBM Lotus Domino®.

Подсистема CM-Sochi-Server (SSRV) и платформа ActiveFrame 5® (AF5) в данной версии CompanyMedia® позволяют получить конфигурацию, которая:

* реализована на открытом ПО,
* реализована полностью на стеке Java EE технологий,
* использует в качестве хранилища РСУБД.

В такой конфигурации, вместо Domino, используется связка открытого ПО: Java EE – сервера приложений JBoss EAP, РСУБД PostgreSQL и платформа AF5 (тоже разработка компании InterTrust®).



Платформа AF5 обеспечивает базовые операции для:

* работы с хранилищем на РСУБД,
* получения списков сущностей в хранилище,
* контроля доступа к хранимым сущностям,
* построения UI-интерфейсов для Web,
* уведомления пользователей.

Подсистема CMJ:

* Реализует Web-клиента CompanyMedia – CMJ-WebGUI.
* Серверная часть (CMJ-Server) обеспечивает обмен данными по HTTP с CMJ-WebGUI и содержит часть бизнес-логики.

Назначение подсистемы SSRV:

* Обеспечить взаимодействие CMJ-Server с подсистемами хранения и контроля доступа платформы AF5, без их модификации.
* Предоставить настройки для платформы AF5, специфичные CM.
* Повторить бизнес-логику, реализованную внутри компоненты Domino (агенты).
* Предоставить CMJ альтернативу языка выражений @-формул.

Назначение приложения CM-AdminGUI:

* Реализует АРМы, портированные с Domino и не реализованные в CMJ-WebGUI. Такие, как АРМ администратора информационного обеспечения, АРМ администратора НСД и т.п.

# Концепция

В конфигурации СМ на Domino, CMJ-Server взаимодействует с ним через набор интерфейсов, задекларированных в Notes Java API, являющейся частью Domino. Основная идея, заложенная в SSRV – сделать альтернативную реализацию этих интерфейсов, которая будет соблюдать тот же контракт, что и в СМ на Domino, но делегировать вызовы в AF5. Этим обеспечивается неизменность кода CMJ-Server и возможность получения конфигурации на платформе альтернативной Domino.

Бизнес-логика в Domino реализована с помощью программных задач, запускаемых по-расписанию (Domino-агенты по-расписанию) и задач, запускаемых CMJ-Server'ом синхронно через Notes Java API (агенты c контекстным документом). В СМ на AF5 бизнес-логика переписана на Java. Для задач по-расписанию разработан пользовательский интерфейс для определения расписаний запуска группы задач, а также механизм для их запуска. Интерфейс доступен через отдельное Web-приложение «CM-AdminGUI».

Domino-агенты, осуществляющие уведомление пользователей, переделаны на использование подсистемы «уведомления» платформы AF5. Подсистема оправляет уведомления задачей по-расписанию, либо реагирует на изменения значений атрибутов сущности.

Выражения на @-формулах решено заменить на JavaScript-выражения. CMJ-Server все @-выражения получает из хранимых в справочниках данных, либо захаркоженных. Поэтому СМ на AF5 поставляется со справочными данными по-умолчанию с портированными на JavaScript-выражениями. Захаркоженные выражения CMJ-Server получает через программные интерфейсны. Реализация интерфейсов для СМ на AF5 подменена для возврата портированных на JavaScript-выражений.

Классы, выполняющие поиск через запросы на @-формулах заменяется на реализацию, которая основана на подсистеме «коллекций» платформы AF5. Подсистема позволяет определять поисковые выражения через язык, очень похожий на SQL.

Классы, выполняющие полнотекстовый поиск через язык запросов Domino, заменены на реализацию, использующую подсистему «полнотекстового поиска» платформы AF5.

АРМы, которые в CM на Domino доступны только через пользовательский интерфейс Domino, например «АРМ предметного администратора», в СМ на AF5 реализованы с использованием UI-средств платформы в приложении CM-AdminGUI.

# Логические Компоненты



## EntityMapper

* Содержит xml-описания структуры хранимых сущностей (типы ДОП платформы AF5).
* Реализует механизмы преобразования элементов хранения данных платформы AF5 (ДОП) в формат Notes-Document и обратно.

Все сущности CMJ-Server сохраняет и читает через объект с интерфейсом lotus.domino.Document из состава Notes Java API. Интерфейс содержит методы для чтения и модификации атрибутов сущности.

При отображении Document в объекты платформы, как правило, задействуется множество разнотипных доменных объектов платформы (ДОП), например, для сохранения множественных значений полей. Т.е. одному типу Document соответствует N типов ДОП, которые связаны между собой в древовидную структуру. Корень дерева будем называть «корневым ДОП» (КДОП).

При сохранении Document, на основании ключевых атрибутов сущности определяется её тип (РКК, КР и т.п.). По имени типа сущности, имени типа и «комплекта» модуля, к которому относится Document, ищется именованный Spring-бин с интерфейсом EntityMapper, описывающий логику конвертации Document в иерархию ДОП и обратно. Если бин не найден, ищется бин по имени типа сущности и типа модуля. Если и такой не найден, ищется по имени типа сущности. Такой подход позволяет иметь единые правила преобразования однотипных сущностей, логически расположенных в разных бизнес-модулях или комплектах.

Компонента EntityMapper названа по имени её ключевого интерфейса. Любой объект EntityMapper выполняет сопоставление поступивших для сохранения данных, с уже хранимыми данными. Модификация ДОП выполняется только если данные изменены. Если хотя бы один ДОП из иерархии изменён, выполняется фиктивная модификация корневого ДОП для того, чтобы изменилась хранимая дата его последней модификации. Дата модификации корневого ДОП используется как дата модификации всей сущности в целом.

При запросе на извлечение сущности из хранилища, ищется её корневой ДОП. Любой КДОП имеет ссылку на модуль, к которому относится сущность. По типу КДОП, типу модуля и «комплекту» модуля ищется EntityMapper, который и выполняет наполнение Document данными.

AF5 отслеживает конкурентную модификацию ДОП. Но не считает конфликтом, если первый поток изменил один ДОП из иерархии сущности, а второй поток изменил другой ДОП той же сущности. Обязательность сохранения КДОП при изменении любого ДОП из иерархии, решает и эту проблему. Когда AF5 фиксирует конфликт модификации КДОП, бросается эксепшен, который приводит к откату транзакции в «проигравшем» потоке, а значит и откату модификаций всех ДОП из иерархии сущности.

### Кэширование сущностей

AF5 обеспечивает кэширование ДОП в рамках одной транзакции. Транзакция открывается на каждый HTTP-запрос и закрывается после формирования HTTP-ответа.

Для экономии операций конвертации иерархии ДОП в Document, SSRV имеет LFU-кэш считанных из хранилища сущностей. Ключом кэша является идентификатор сущности и идентификатор потока исполнения. Идентификатор потока в ключе предотвращает получение одного java-объекта разными потоками и избавляет от необходимости управления конкурентным доступом к java-объекту на уровне SSRV. Использование пула потоков обеспечивает хорошее кэш-попадание. Конфликты конкурентной модификации ДОП отслеживает AF5.

При развертывании СМ на нескольких серверах приложений с общей БД, возможна модификация ДОП другим экземпляром SSRV. Для обеспечения получения актуальных, не закешированных данных на первом экземпляре SSRV, при считывании сущности всегда извлекается её КДОП и сравнивается дата его модификации с датой в кэше. Если дата совпадает, данные берутся из кэша. Не совпадает – выполняется полное считывание иерархии ДОП из БД и конвертация в Document. Этот принцип обеспечивает и актуальность данных после модификации сущности разными потоками одного экземпляра SSRV.

### Идентификаторы сущностей (UNID)

CMJ-Server применяет подход, когда идентификатор сущности (UNID) генерируется и назначается сущности ещё до её первого сохранения. Платформа AF5 назначает идентификатор только после первого сохранения ДОП. Для установления соответствия сгенерированного UNID идентификатору КДОП, использованы служебные ДОП типа «nunid2punid\_map», которые хранят

• назначенный UNID

• соответствующий id ДОП

При первом сохранении сущности, после сохранения её КДОП, создаётся и ДОП типа «nunid2punid\_map». Если CMJ-Server запрашивает сущность по назначенному UNID, по «nunid2punid\_map» определяется id её КДОП, и по полученному id уже извлекается КДОП средствами AF5. При поиске сущности не по id, например через коллекции, выполняется обратное преобразование id найденных КДОП в назначенный UNID.

## CollectionProvider

* Содержит xml-описания списков хранимых сущностей (коллекции AF5).
* Реализует механизм извлечения списков, функционально эквивалентный Notes-View.
* Базируется на подсистеме «коллекций» платформы, добавляя возможность преобразования нескольких колонок в одну (виртуальные колонки), группировки по значениям колонок и определения сортировок по-умолчанию.

При запросе на получение именованного списка через CollectionProvider, выполняется поиск коллекции платформы по имени списка, типу модуля и «комплекту» модуля, к которому относится данный CollectionProvider. Если коллекция не найдена, ищется по имени списка и типу модуля. Если не найдено, то только по имени списка. К найденной коллекции применяется фильтр с зарезервированным именем MODULE, который должен отбирать сущности, приписанные заданному модулю. После нахождения точного имени коллекции, ищется Spring-бин с именем коллекции плюс суффикс «Metadata». Из этого бина с метаданными, CollectionProvider получает правила формирования виртуальных колонок, сортировки и категоризации по-умолчанию. Такой подход позволяет иметь единые описания списков сущностей, логически расположенных в разных модулях или комплектах.

Поля виртуальной колонки вычисляются в отложенном режиме, в момент, когда запрашивается значение виртуального поля. Этим достигается экономия на вычислении значений виртуальных полей в больших списках.

Средствами CollectionProvider моделируются методы получения респонсов Notes API. Для этого у модуля должна быть коллекция с именем «document\_responses», которая должна иметь фильтр «parent». Фильтр должен отбирать респонсы первого уровня для заданной родительской сущности.

Для получения локализованных списков, у коллекции должен быть фильтр «byLocale», который отбирает локализованные значения сущностей. CollectionProvider сначала считает все имена локалей, сохраненные БД. Имя таблицы с локализованными значениями и имя колонки с именами локалей берутся из бина с метаданными списка. Затем получает имя локали из настроек браузера пользователя. Среди хранимых локалей находит наиболее подходящую пользовательской локали. Полученное имя локали CollectionProvider подставляет как значение параметра фильтру «byLocale».

## UI-Collections

* Формирует иерархию списков сущностей в формате CMJ, в соответствии с настройками, заданными через CM-AdminGUI и отображаемую в интерфейсе CMJ-WebGUI в виде иерархии папок.

## JS

* Реализует движок исполнения JavaScript
* Формирует контекст исполнения для скриптов

## NamedCode

* Реализует синхронный запуск именованных Notes-агентов.

## Notification

* Содержит описание типов e-mail уведомлений
* Содержит локализованные шаблоны текстов уведомлений
* Описывает события, по которым должно отправляться уведомление заданного типа

## PKD

* Формирует персональные задачи и уведомления, отображаемые в интерфейсе CMJ-WebGUI, отслеживает их статус.

В CMJ-Server на Domino реализована функциональность хранения PKD-уведомлений и PKD-задач в реляционной СУБД. Их источником являлась notes-БД СПКД. Передача данных между СПКД и реляционным хранилищем выполняется по-расписанию. В конфигурации CM на AF5 данные для ПКД поставляются через её API, которое выполняет их сохранение средствами ORM Hibernate, т.е. без использования средств платформы AF5. Интеграция с ПКД в СМ на AF5 выполнена путём создания специального канала уведомлений платформы AF5. Обработчик канала выполняет сопоставление типа уведомления типу PKD-уведомления или PKD-задачи и через API PKD выполняет их создание или обновление.

## FT-Search

* Реализует полнотекстовый поиск сущностей

## SoService

* Реализует операции по работе со справочниками СО, СпО, СпП

## Reporter

* Содержит настройки с шаблонами и параметрами отчетов
* Выполняет построение отчетов
* Реализован с использованием JasperReports компоненты платформы AF5.

## AgentRunner

* Реализует запуск программных задач по-расписанию или при изменении сущностей

## Data-CSV

* Выполняет инициализацию БД значениями по-умолчанию, предоставленными в CSV-формате.

## AclRules

* Содержит описание правил предоставления доступа к хранимым сущностям
* Базируется на подсистеме «прав доступа» платформы AF5

Проверку прав на создание и модификацию сущностей осуществляет CMJ-Server, в соответствии с настройками в профиле пользователя и назначением его на роли. Поэтому компонент AclRoles описывает лишь правила предоставления доступа «на чтение» сущностей. Помимо авторизации, правами также обеспечивается, чтобы в списки попадали только доступные сущности.

В конфигурации CM на Domino, после изменений в СО, таких, как назначение персоны на должность, назначение замещающего, передача дел, происходит пересчёт прав доступа. При пересчёте происходит модификация сущности. Такая техническая по сути модификация, значительно повышает вероятность появления конфликта сохранения пользовательских изменений. AF5, при пересчёте прав, не модифицирует ДОП, а значит пересчет не может привести к неуспешному сохранению пользовательских изменений.

## Authentification

* Реализует аутентификацию пользователей в системе
* Базируется на JAAS подсистеме Java EE

## CM-Agents

* Реализует бизнес-логику в виде программных задач, запускаемых через AgentRunner

## Notes-adapter

* Реализует программные интерфейсы Notes Java API с использованием подсистем EntityMapper, CollectionProvider, NamedCode, JS, Authentification

## Cmj-beans

* Содержит альтернативную имплементацию компонент CMJ, которые задействуют специфичные Domino операции, такие, как поиск по @-формуле, полнотекстовый поиск.

## Storage-SPI

* Интерфейсы для абстрагирования от реализации подсистемы хранения

## CM-AdminGUI

* Реализует пользовательский интерфейс для АРМ предметного администратора средствами AF5.
* Содержит серверную логику обработки сохранения сущностей через UI

# Состав артефактов поставки



ssrv-sochi-fellow.jar и ssrv-sochi-fellow-shared.jar содержат компоненты, которые должны деплоиться в едином java-приложении с платформой AF5. Такие, как настройки для AF5, реализации точек расширения AF5 и т.п. Это требование платформы AF5. Дело в том, что потенциально возможна схема развертывания, когда AF5 и Sochi-fellow разворачиваются на одном хосте, а CM-Sochi-Server на другом. Взаимодействие между ними тогда реализуется средствами Remote EJB. Такая схема не рекомендуется из-за низкой производительности.

ssrv-sochi-fellow-shared.jar должна также входить и в состав приложения с ssrv-agents.jar

ssrv-agents.jar, и ряд библиотек платформы AF5 и CMJ-Server, могут быть скомпанованы в отдельное java-приложение и развернуты на отдельном сервере приложений для решения задач масштабирования. Взаимодействие с компонентами ssrv-sochi-fellow при этом происходит через JMS.

ssrv-admin-gui.jar и AF5 потенциально могут компоноваться в отдельное java-приложение для деплоя на отдельный сервер.



Приложение ssrv-overall.ear содержит компановку, предназначенную для развертывания всех компонент системы на одном сервере приложений. Содержит 2 web-модуля ssrv-war.war и ssrv-agents.war. Ssrv-war.war включает в себя также AF5, CMJ-Server и CM-AdminGUI.

## Зависимости



# Схема развертывания

Пример развертывания CM для двух «системных организаций».



Используется единая БД для автоматизации процессов между организациями и переиспользования справочников СО и СС. Для каждой организации обязательно наличие отдельного сервера приложений, т.к. сервер конфигурируется параметрами, специфичными организации. Платформа AF5 должна быть настроена на режим работы «в кластере серверов приложений»

Приложение cm-web-gui.war может разворачиваться и на отдельном сервере приложений, но по одному на каждую организацию.

1. CompanyMedia является зарегистрированной торговой маркой ЗАО «Компания «ИнтерТраст» [↑](#footnote-ref-1)
2. ActiveFrame является зарегистрированной торговой маркой ЗАО «Компания «ИнтерТраст» [↑](#footnote-ref-2)