ООО «Цитадель»
127015, г. Москва, ул. Новодмитровская, д. 2Б
+7 (495) 666 2 333, info@ctdl.ru

ПО ТС ОРМ «ОЛИМП-MUX»

Руководство по установке и настройке ПО

Содержание

1 Общая информация

Программное обеспечение ТС ОРМ «ОЛИМП-MUX» (далее – ПО ТС ОРМ «ОЛИМП-MUX») в составе программно-аппаратного комплекса «ОЛИМП-MUX» (далее – ПАК «ОЛИМП-MUX») предназначено для консолидации данных, получаемых от коммутационного оборудования на сетях GSM/UMTS, построенных на базе оборудования Huawei, а также информации, поступающей по интерфейсам СОРМ от ТС ОРМ коммутационного оборудования, не входящего в состав MSoftX3000. Полученные и обработанные данные ПО ТС ОРМ «ОЛИМП-MUX» передаёт на один или несколько ПУ ОРМ, осуществляя взаимодействие в соответствии с требованиями Приказов Минкомсвязи России №645 от 12.12.2016 (далее – Приказ №645) и №174 от 11.07.2011 (далее – Приказ №174).

Виды услуг, с которыми работает ПО ТС ОРМ «ОЛИМП-MUX»:
- Голосовые вызовы;
- Сообщения SMS, USSD;
- Дополнительные виды обслуживания (ДВО).

ПО ТС ОРМ «ОЛИМП-MUX» поддерживает кодек G.711.

Общая схема работы ПО ТС ОРМ «ОЛИМП-MUX» в составе ПАК ТС ОРМ «ОЛИМП-MUX» приведена на рисунке ниже.

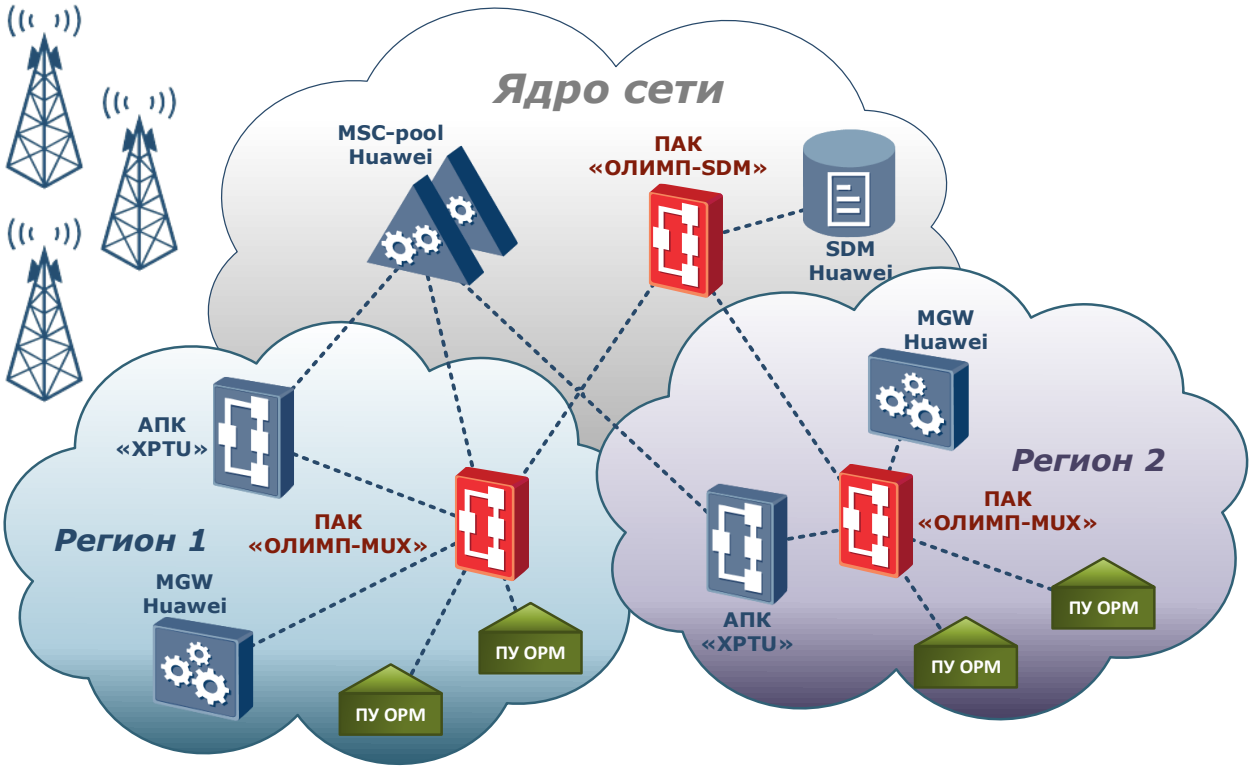

Рисунок 1. Общая схема работы ПО ТС ОРМ «ОЛИМП-MUX» в составе ПАК «ОЛИМП-MUX» на сетях GSM/UMTS, построенных на базе оборудования Huawei

2 Программное обеспечение ТС ОРМ «ОЛИМП-MUX»

ПО ТС ОРМ «ОЛИМП-MUX» включает в себя следующие программные компоненты:
- **tropa**;
- **sorm_xptu**;
- **mmx**;
- **installer**.

Структурная схема и взаимодействие ПО ТС ОРМ «ОЛИМП-MUX» в составе ПАК «ОЛИМП-MUX» с внешними системами представлена на рисунке 2.

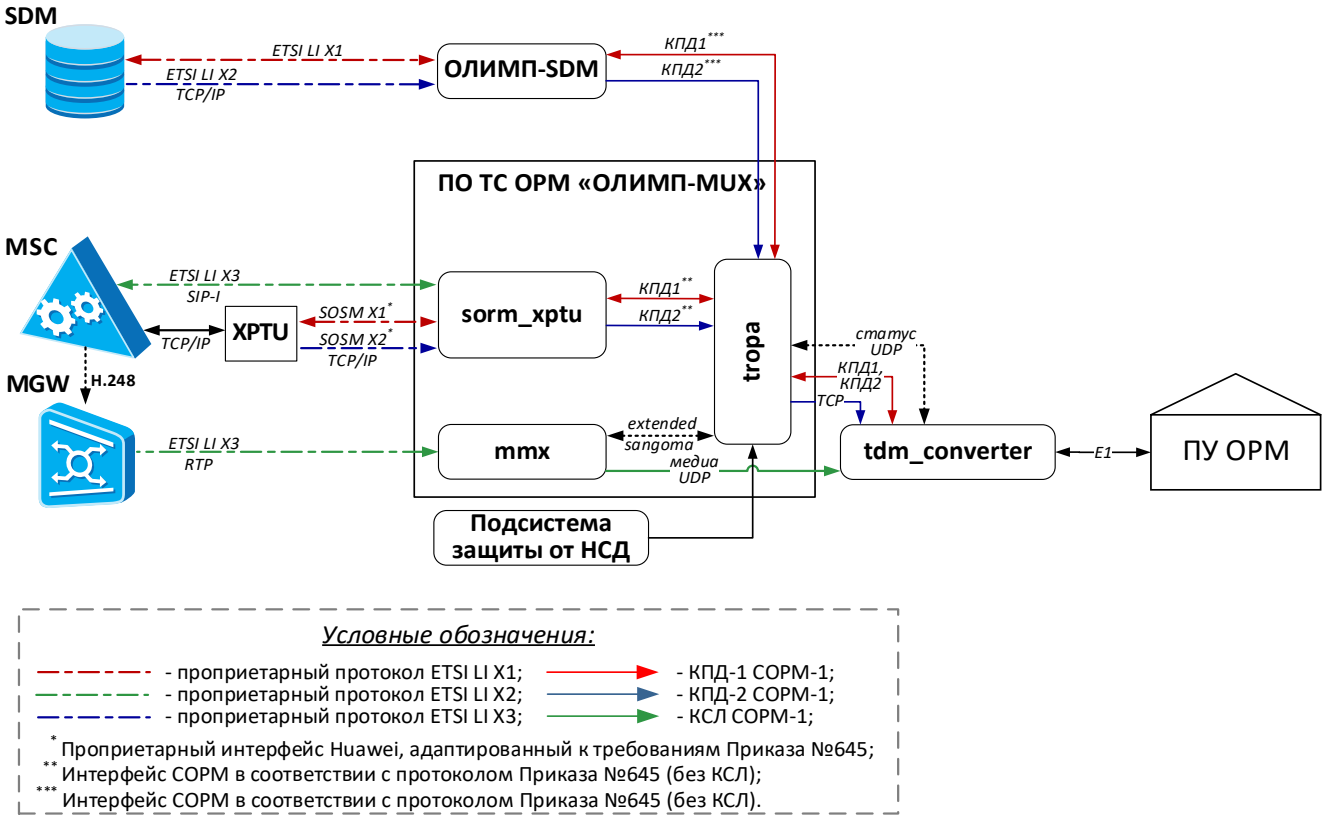

Рисунок 2. Структурная схема ПО ТС ОРМ «ОЛИМП-MUX» в составе ПАК «ОЛИМП-MUX»

2.1 Компонент tropa

Компонент **tropa** выполняет следующие функции:
- сопряжение нескольких интерфейсов ТС ОРМ и нескольких ПУ ОРМ;
- консолидация информации, поступающей от всех подключенных элементов ТС ОРМ для последующей передачи на ПУ ОРМ в едином интерфейсе;
- формирование интерфейса к ПУ ОРМ в соответствии с требованиями Приказа №645.

Конфигурирование компонента **tropa** приведено в разделе 4.1.

2.2 Компонент sorm_xptu

Компонент **sorm_xptu** выполняет следующие функции:
– реализация интерфейса LI в части X1, X2 и сигнализации X3 (SIP-I) для взаимодействия с XPTU/MSC;
– реализация специфичных сценариев взаимодействия с XPTU (например, при MSC-pool).

Конфигурирование компонента **sorm_xptu** приведено в разделе 4.2.

2.3 Компонент mmx

Компонент **mmx** состоит из двух программных модулей - **MmxProxy** и **MmxService**, взаимодействующих между собой по протоколам TCP и UDP.

Взаимодействие компонента **mmx** со сторонними компонентами происходит следующим образом:

1) от MGW сети оператора принимается RTP/UDP;

2) компонент **tropa** инициирует TCP-подключение;

3) на компонент **tdm_converter** (не входит в состав ПО «ОЛИМП-MUX») данные отправляются по UDP.

Модулей **MmxService** может быть несколько - столько же, сколько и ПУ ОРМ, т.к. каждый модуль **MmxService** обслуживает конкретный ПУ ОРМ.

Модуль **MmxProxy** выполняет следующие функции:
– обеспечивает приём команд от компонента **tropa**;
– принимает медиаданные от MGW сети оператора;
– проксирует команду и медиаданные для конкретного ПУ ОРМ на соответствующий модуль **MmxService**.

Модуль **MmxService** выполняет следующие функции:
– приём команд от модуля **MmxProxy**;
– приём медиаданных от модуля **MmxProxy**;
– микширование (в случае необходимости);
– упаковка медиаданных и формирование пакетов в соответствующие форматы:
  • для компонента **tdm_converter** (не входит в состав ПО ТС ОРМ «ОЛИМП-MUX») - в формат пакетов Sangoma;
  • для ПУ ОРМ - в формат пакетов в соответствии с требованиями Приказа №645;
– отправка сформированных пакетов на компонент **tdm_converter** (не входит в состав ПО ТС ОРМ «ОЛИМП-MUX»), либо на ПУ ОРМ.

Конфигурирование компонента **mmx** приведено в разделе 4.3.

2.4 Зависимости от сторонних программных модулей

Для ведения журналов событий работы компонентов подсистемы используется библиотека **liblog4cplus-1.2.so.5.1.6**, поставляемая в составе дистрибутива.

Для работы подсистемы должна быть установлена стандартная библиотека **libstdc++.so.6.0.25** или выше, входит в состав поставляемого образа ОС.

2.5 Компонент installer

2.5.1 Назначение и область применения

Компонент **Installer** – это установщик компонентов, призванный заменить привычные RPM-пакеты. Основное преимущество: возможность установки N независимых инстансов на сервер.

2.5.2 Описание работы с компонентом. Запуск

Внимание! Название инстансов и компонентов является примером и не является обязательным.

Запуск осуществляется исполняемым файлом находящимся в каталоге установщика **./installer** и передачей ему аргументов.

2.5.3 Описание аргументов

Аргумент help/h

Используется для вывода справки. Пример использования:

```
./installer -h
./installer --help
```

Также возможно использование после аргумента **in/un/up**.

Аргумент install/in

Используется для установки компонента и всех зависимостей в системе. Пример использования:

```
./installer in
./installer install
```

Аргумент uninstall/un

Используется для удаления компонента и всех зависимостей в системе.

```
./installer un
./installer uninstall
```

Аргумент updade/up

Используется для обновления компонента и всех зависимостей в системе.

```
./installer up
./installer updade
```

2.5.4 Примеры инсталляции (установка, обновление, удаление)

Установка:

```
./installer in /home/tropa-x.x-x.x.zip  /home/mux/ tropa
./installer install /home/tropa-x.x-x.x.zip  /home/mux/ tropa
```

Удаление:

```
./installer un /home/mux/ tropa
./installer uninstall /home/mux/ tropa
```

Обновление:

```
./installer up /home/tropa-x.x-x.x.zip  /home/mux/ tropa
./installer update /home/tropa-x.x-x.x.zip  /home/mux/ tropa
```

2.5.5 Структура инсталляции

Все компоненты **Installer** по умолчанию расположены в каталоге **/home**. Рекомендуется создавать директории по следующему правилу:

/
|- home
** |- product_name**
** |- module_name**
** |- instance_name_1**
** |- bin**
** |- cfg**
** |- lib**
** |- temp**
...
** |- instance_name_N**
** |- bin**
** |- cfg**
** |- lib**
** |- temp**

Директории **bin/cfg/lib/temp** создаются инсталлятором автоматически.

Пример иерархического списка с описанием подкаталогов и файлов приведен ниже:

/
|- home
** |- mux**
** |- mmx**
** |- mmx_proxy**
** |- ...**
** |- mmx_service_E1**
** |- ...**
** |- mmx_service_IP**
** |- ...**
** |- sorm_xptu**
** |- ...**
** |- tropa**
** |- ...**
...

2.5.6 Примеры управления модулем

Запуск модуля осуществляется командой **start**, пример ниже:

```
systemctl start tropa.service
systemctl start sorm_xptu.service
```

Остановка модуля осуществляется командой **stop**, пример ниже:

```
systemctl stop tropa.service
systemctl stop sorm_xptu.service
```

Перезапуск модуля осуществляется командой **stop/start**, пример ниже:

```
systemctl stop tropa.service
systemctl start tropa.service
systemctl stop sorm_xptu.service
systemctl start sorm_xptu.service
```

Запрос версии осуществляется командой **version**, пример ниже:

```
tropa version
sorm_xptu version
```

Запрос статуса осуществляется командой **status**, пример ниже:

```
tropa status
sorm_xptu status
```

2.5.7 Установка автозапуска

Указание опции автозапуска и удаление программы из списка автозапуска осуществляется с помощью утилиты **systemctl**:

```
# Добавить экземпляр список автозапуска
sudo systemctl enable <имя_экземпляра>.service

# Удалить экземпляр из списка автозапуска
sudo systemctl disable <имя_экземпляра>.service

# Проверить статус автозапуска экземпляра
sudo systemctl is-enabled <имя_экземпляра>.service

# Просмотреть список всех имеющихся сервисов на машине
sudo systemctl list-unit-files
```

3 Установка ПО ТС ОРМ «ОЛИМП-MUX»

3.1 Установка компонентов tropa, sorm_xptu, mmx

Дистрибутив представляет собой папку вида **X.XX**, где X.XX - номер релиза, содержащую архивы дистрибутивов для их установки или обновления.

Папка с дистрибутивами копируется на сервер. Сначала распаковывается дистрибутив компонента **installer** в заранее созданный каталог.

Пример команды распаковки:

```
tar xaf installer-x.x-x.x.tar.gz
```

Далее производится установка компонентов комплекса, примеры описаны выше в пункте 2.4.4 описания компонента installer.

Структура инсталяции и примеры управления модулями так же описаны выше в пунктах 2.4.5 и 2.4.6 соответственно.

Количество устанавливаемых и запускаемых **mmx_service** определяется количеством подключаемых ПУ ОРМ.

4 Настройка и описание конфигурационных файлов ПО ТС ОРМ «ОЛИМП-MUX»

4.1 Настройка компонента tropa

Для настройки компонента **tropa** используются конфигурационные xml-файлы, которые располагаются в каталоге, где установлен модуль **tropa**. Пример: **/home/mux/tropa/cfg/**.

Конфигурационные файлы разбиты на группы параметров. Название каждого параметра имеет следующий вид: **<имя_параметра>значение</имя_параметра>**

Пример правильного названия параметра: **<Daemon>false</Daemon>**

Значение параметра не должно содержать символы:

- пробел, табуляция;
- перевод строки.

В названиях параметров и групп параметров необходимо соблюдать регистр букв.

Список конфигурационных файлов, подлежащих изменению/корректировке:

– **MainConfig.xml** – конфигурационный файл, содержащий основные параметры настройки компонента **tropa**;
– **SystemsConfig.xml** – конфигурационный файл, включающий в себя параметры настройки "подсистем" компонента **tropa;**
– **alarm_detector.cfg** – файл настройки клиента для подключения к серверу НСД (сервер НСД не входит в состав ПО «ОЛИМП-MUX»);
– **MclConfig.xml** – конфигурационный файл, описывающий распределение КСЛ на ТС ОРМ.

Список конфигурационных файлов, не подлежащих изменению/корректировке:

– **Order645ProtocolConfig.xml** – конфигурационный файл, описывающий формат команд и сообщений в соответствии с требованиями Приказа №645;
– **Order174ProtocolConfig.xml** – конфигурационный файл, описывающий формат команд и сообщений в соответствии с требованиями Приказа №174;
– **ExtendedOrder645ProtocolConfig.xml** – конфигурационный файл, описывающий формат команд и сообщений для взаимодействия с **sorm_xptu**.
– **SangomaProtocolConfig.xml** – конфигурационный файл, описывающий формат сообщений для взаимодействия с компонентами **tdm_converter** (не входит в состав ПО ТС ОРМ «ОЛИМП-MUX») и **mmx**.

4.2 Настройка компонента sorm_xptu

4.2.1 Конфигурационный файл модуля sorm_xptu

Файл конфигурации **sorm_xptu.cfg** расположен в каталоге с установленным модулем **sorm_xptu**. Пример: /home/mux/sorm_xptu/cfg/

4.3 Настройка компонента mmx

4.3.1 Конфигурационный файл MmxProxy

Файл конфигурации **mmx_proxy.xml** расположен в каталоге с установленным модулем **mmx_proxy**. Пример: /home/mux/mmx_proxy/cfg

Корневым элементом является Mmx, а вложенные в него элементы определяют конфигурационные параметры и смысловые группы конфигурационных параметров.

4.3.2 Конфигурационный файл MmxService

Файл конфигурации **mmx_service.xml** расположен в каталоге с установленным модулем **mmx_service**. Пример: /home/mux/mmx_service/cfg

Корневым элементом является **MmxService**, а вложенные в него элементы определяют конфигурационные параметры и смысловые группы конфигурационных параметров.

5 Обновление ПО ТС ОРМ «ОЛИМП-MUX»

5.1 Обновление компонента tropa

Для обновления компонента **tropa** необходимо произвести обновление с помощью модуля **installer** и полученного дистибутва модуля **tropa**. Пример приведен выше в пункте 2.4.4 описания компонента **installer**..

5.2 Обновление компонента sorm_xptu

Для обновления компонента **sorm_xptu** необходимо произвести обновление с помощью модуля **installer** и полученного дистибутва модуля **sorm_xptu**. Пример приведен выше в пункте 2.4.4 описания компонента **installer**.

5.3 Обновление компонента mmx

Для обновления компонента **mmx** необходимо произвести обновление с помощью модуля **installer** и полученных дистибутивов модулей **mmx_proxy** и **mmx_service**. Пример приведен выше в пункте 2.4.4 описания компонента **installer**.

6 Работа с программными компонентами ПО ТС ОРМ «ОЛИМП-MUX»

Для работы с ПО ТС ОРМ «ОЛИМП-MUX» используется два инструмента:

1) Управление каждым модулями через systemd после установки с помощью модуля **installer.** Примеры описаны выше в пункте 2.4.6 описания компонента **installer**.
2) скрипт <имя_скрипта>**.sh** (см. раздел 0);

6.1 Скрипт <имя_скрипта>.sh

В архиве компонента **installer** идёт скрипт-алиас для сервисов **generate_alias.sh**.
При запуске скрипт выводит список установленных ранее на сервер сервисов и предлагает выбрать N сервисов, далее для N сервисов создаётся скрипт <имя_скрипта>.sh управляющий через system всеми вложенными сервисами.

Пример:

1) запуск скрипта

```
# ./generate_alias.sh
Installed services:
0: mmx_proxy
1: mmx_service
2: sorm_xptu
3: tropa
Enter services' numbers you wish to control in start order (space separated):
```

2) выбор номеров сервисов в том порядке, в каком они будут запускаться (через пробел)

```
Enter services' numbers you wish to control in start order (space separated): 2 0 1 3
```

3) выбор номеров сервисов в том порядке, в каком они будут останавливаться (через пробел)

```
Enter services' numbers you wish to control in stop order (space separated): 3 2 1 0
```

4) пишем название скрипта, после чего появляется скрипт **<имя_скрипта>.sh** для управления ранее установленными сервисами

```
Enter name for alias script: example
# ls
example.sh
```

Скрипт **<имя_скрипта>.sh** в ПО ТС ОРМ «ОЛИМП-MUX» возможно использовать для следующих компонентов:

- **tropa**;
- **mmx**;
- **sorm_xptu**.

С помощью скрипта поддерживаются следующие команды работы перечисленных выше компонентов:

{start|stop|restart|version|status}

- Запуск

Для запуска всех модулей с помощью скрипта необходимо выполнить команду:

```
# ./examle.sh start
```

- Остановка

Для остановки всех модулей с помощью скрипта необходимо выполнить команду:

```
# ./examle.sh stop
```

- Перезапуск

Для перезапуска всех модулей с помощью скрипта необходимо выполнить команду:

```
# ./examle.sh restart
```

- Проверка версии

Для вывода версий всех модулей с помощью скрипта необходимо выполнить команду:

```
# ./examle.sh version
```

- Проверка статуса

Для вывода статуса работы всех модулей с помощью скрипта необходимо выполнить команду:

```
# ./examle.sh status
```