Ниже представлена подробная инструкция по использованию вашего SOAP‑/JSON‑прокси. Эта инструкция описывает, как отправлять запросы к прокси, на какие URL отправлять, какие параметры должны быть в теле запроса, какие заголовки требуются и что ожидается в ответе. Инструкция рассчитана на пользователей с любым уровнем подготовки.

# **Инструкция по использованию SOAP‑/JSON‑прокси**

## **1. Общее описание**

Наш прокси-сервер предназначен для пересылки запросов от клиентов к целевому API, расположенного по адресу:  
 **http://gw.chel.mnogomed.ru:9095/HubServiceJson.svc**.

Прокси работает по схеме «прозрачного моста»:

* Клиент обращается по URL вашего сервера, например,  
   **http://<IP\_сервера>:3001/proxy/<MethodName>**
* Прокси получает запрос, добавляет к базовому URL целевого API имя метода (в параметре <MethodName>) и пересылает запрос дальше.
* Ответ, полученный от целевого API, возвращается клиенту без изменений.

На данном сервере прокси запущен на порту **3001**.

## **2. Структура URL и маршрутов**

Формат запросов к прокси:

* **Базовый адрес прокси:** http://<IP\_сервера>:3001
* **Путь для запроса:** Все запросы должны начинаться с префикса:  
   **/proxy/**
* Далее добавляется имя метода целевого API. Например, если вы хотите вызвать метод *GetDistrictList*, то полный URL для запроса будет:  
   **http://51.250.34.77:3001/proxy/GetDistrictList**.

Прокси автоматически формирует конечный URL для пересылки запроса, добавляя имя метода к базовому URL API. В нашем случае, целевой URL получается следующим образом:  
 **http://gw.chel.mnogomed.ru:9095/HubServiceJson.svc/<MethodName>**.

## **3. Требования к запросу**

### **3.1 HTTP‑метод**

Большинство запросов отправляются методом **POST**. Если целевой API требует другой метод (например, GET для получения данных), это можно настроить, но в наших примерах используется POST.

### **3.2 Заголовки**

Обязательно включайте следующий заголовок:

* **Content-Type:** При использовании JSON‑формата указывайте:  
   application/json

### **3.3 Тело запроса**

Тело запроса должно быть корректно оформленным JSON-объектом с параметрами, необходимыми для конкретного метода. Обязательные и дополнительные поля зависят от метода целевого API. Например:

* Для метода **CheckPatient** — параметры: guid, idLpu, и объект pat с данными пациента.
* Для метода **GetDistrictList** — параметры: guid, idHistory (можно передать 0, если не требуется сессионный идентификатор).

Прокси пересылает тело запроса без изменений (при необходимости пытаясь распарсить текстовое содержимое в JSON).

## **4. Примеры использования**

Ниже приведены примеры отправки запросов для различных методов. Обратите внимание, что строку "ваш-guid" следует заменить на действительный идентификатор, а числовые значения — на актуальные данные.

### **4.1. GetDistrictList**

**Назначение:** Получение списка районов.  
 **Параметры (пример):**

* guid: Ваш GUID-токен.
* idHistory: Идентификатор сессии (например, можно передавать 0, если не требуется).

**URL запроса:**

http://51.250.34.77:3001/proxy/GetDistrictList

**Пример запроса через curl (Windows CMD):**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idHistory\": 0}" "http://51.250.34.77:3001/proxy/GetDistrictList"

**Ожидаемый ответ:** В ответ вы получите JSON, содержащий список районов, например:

{

"GetDistrictListResult": {

"ErrorList": [],

"IdHistory": null,

"Success": true,

"Districts": [

{"DistrictName": "Челябинская обл", "IdDistrict": 74, "Okato": null},

{"DistrictName": "Новгородская обл", "IdDistrict": 53, "Okato": null},

{"DistrictName": "Белгородская обл", "IdDistrict": 31, "Okato": null},

{"DistrictName": "Оренбургская обл", "IdDistrict": 56, "Okato": null}

]

}

}

### **4.2. CheckPatient**

**Назначение:** Идентификация пациента.  
 **Параметры (пример):**

* guid: Ваш GUID-токен.
* idLpu: Идентификатор медицинской организации.
* pat: Объект с данными пациента, например:  
  + Surname: Фамилия.
  + Name: Имя.
  + Birthday: Дата рождения (в формате ISO 8601).

**URL запроса:**

http://51.250.34.77:3001/proxy/CheckPatient

**Пример запроса через curl:**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idLpu\": 101, \"pat\": {\"Surname\": \"Иванов\", \"Name\": \"Иван\", \"Birthday\": \"1985-01-01T00:00:00Z\"}}" "http://51.250.34.77:3001/proxy/CheckPatient"

**Ожидаемый ответ:** Ответ будет содержать информацию об успешности идентификации пациента, например:

{

"CheckPatientResult": {

"Success": true,

"IdPat": "уникальный-идентификатор",

"ErrorList": []

}

}

### **4.3. AddNewPatient**

**Назначение:** Регистрация нового пациента.  
 **Параметры (пример):**

* guid
* idLpu
* surname, name, birthday, phone и другие поля, необходимые для регистрации.

**URL запроса:**

http://51.250.34.77:3001/proxy/AddNewPatient

**Пример запроса через curl:**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idLpu\": 101, \"surname\": \"Петров\", \"name\": \"Пётр\", \"birthday\": \"1990-05-15T00:00:00Z\", \"phone\": \"0987654321\"}" "http://51.250.34.77:3001/proxy/AddNewPatient"

**Ожидаемый ответ:** В ответе должно быть подтверждение регистрации, например:

{

"AddNewPatientResult": {

"Success": true,

"IdPat": "новый-идентификатор",

"ErrorList": []

}

}

### **4.4. GetLPUList**

**Назначение:** Получение списка медицинских учреждений.  
 **Параметры (пример):**

* guid
* (Опционально) idDistrict для фильтрации по району.

**URL запроса:**

http://51.250.34.77:3001/proxy/GetLPUList

**Пример запроса через curl:**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idDistrict\": 74}" "http://51.250.34.77:3001/proxy/GetLPUList"

**Ожидаемый ответ:** Ответ будет содержать список ЛПУ:

{

"GetLPUListResult": {

"Success": true,

"LPUList": [

{"idLpu": 101, "LPUFullName": "Медицинский центр №1", "LPUShortName": "Центр №1"},

{"idLpu": 102, "LPUFullName": "Медицинский центр №2", "LPUShortName": "Центр №2"}

],

"ErrorList": []

}

}

### **4.5. GetDoctorList**

**Назначение:** Получение списка врачей по специальности в определённом ЛПУ.  
 **Параметры (пример):**

* guid
* idLpu
* idSpesiality
* idPat (может быть использован для фильтрации или подстановки значений)

**URL запроса:**

http://51.250.34.77:3001/proxy/GetDoctorList

**Пример запроса:**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idLpu\": 101, \"idSpesiality\": 1001, \"idPat\": 123}" "http://51.250.34.77:3001/proxy/GetDoctorList"

**Ожидаемый ответ:** Ответ вернёт список врачей:

{

"GetDoctorListResult": {

"Success": true,

"Doctors": [

{"idDoc": 201, "FIO": "Иванов И.И.", "CountFreeTicket": 5},

{"idDoc": 202, "FIO": "Петров П.П.", "CountFreeTicket": 3}

],

"ErrorList": []

}

}

### **4.6. GetAvailableDates**

**Назначение:** Получение доступных дат для приема у врача.  
 **Параметры (пример):**

* guid
* idLpu
* idDoc
* idPat
* visitStart и visitEnd — диапазон дат в формате ISO 8601 (например, "2025-04-20T00:00:00Z").

**URL запроса:**

http://51.250.34.77:3001/proxy/GetAvailableDates

**Пример запроса:**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idLpu\": 101, \"idDoc\": 2001, \"idPat\": 123, \"visitStart\": \"2025-04-20T00:00:00Z\", \"visitEnd\": \"2025-04-30T00:00:00Z\"}" "http://51.250.34.77:3001/proxy/GetAvailableDates"

**Ожидаемый ответ:** Ответ содержит список доступных дат:

{

"GetAvailableDatesResult": {

"Success": true,

"Dates": [

"2025-04-21T00:00:00Z",

"2025-04-22T00:00:00Z"

],

"ErrorList": []

}

}

### **4.7. SetAppointment**

**Назначение:** Запись пациента на прием.  
 **Параметры (пример):**

* guid
* idLpu
* idPat
* idAppointment (идентификатор выбранного талона)
* Дополнительные параметры, если требуются.

**URL запроса:**

http://51.250.34.77:3001/proxy/SetAppointment

**Пример запроса:**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idLpu\": 101, \"idPat\": 123, \"idAppointment\": 456}" "http://51.250.34.77:3001/proxy/SetAppointment"

**Ожидаемый ответ:** Ответ будет содержать подтверждение записи:

{

"SetAppointmentResult": {

"Success": true,

"IdHistory": 789,

"ErrorList": []

}

}

### **4.8. UpdatePhoneByIdPat**

**Назначение:** Обновление телефона пациента.  
 **Параметры (пример):**

* guid
* idPat
* Phone

**URL запроса:**

http://51.250.34.77:3001/proxy/UpdatePhoneByIdPat

**Пример запроса:**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idPat\": 123, \"Phone\": \"1234567890\"}" "http://51.250.34.77:3001/proxy/UpdatePhoneByIdPat"

**Ожидаемый ответ:** Ответ подтвердит успешное обновление:

{

"UpdatePhoneByIdPatResult": {

"Success": true,

"ErrorList": []

}

}

### **4.9. CreateClaimForRefusal**

**Назначение:** Отмена записи на прием (создание жалобы на отказ).  
 **Параметры (пример):**

* guid
* idPat
* idAppointment
* Дополнительный параметр (например, reason с описанием причины отмены).

**URL запроса:**

http://51.250.34.77:3001/proxy/CreateClaimForRefusal

**Пример запроса:**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idPat\": 123, \"idAppointment\": 456, \"reason\": \"Пациент не смог прийти\"}" "http://51.250.34.77:3001/proxy/CreateClaimForRefusal"

**Ожидаемый ответ:** Ответ содержит информацию об успешном создании жалобы:

{

"CreateClaimForRefusalResult": {

"Success": true,

"ErrorList": []

}

}

### **4.10. SendNotificationAboutAppointment**

**Назначение:** Уведомление пациента о записи (например, отправка SMS или PUSH-уведомления).  
 **Параметры (пример):**

* guid
* idPat
* messageText (текст уведомления)
* method (метод отправки, например, "SMS" или "PUSH")

**URL запроса:**

http://51.250.34.77:3001/proxy/SendNotificationAboutAppointment

**Пример запроса:**

curl -X POST -H "Content-Type: application/json" --data "{\"guid\": \"ваш-guid\", \"idPat\": 123, \"messageText\": \"Ваша запись подтверждена\", \"method\": \"SMS\"}" "http://51.250.34.77:3001/proxy/SendNotificationAboutAppointment"

**Ожидаемый ответ:** Ответ будет содержать статус отправки уведомления:

{

"SendNotificationAboutAppointmentResult": {

"Success": true,

"ErrorList": []

}

}

## **5. Советы и рекомендации**

* **Проверка перед отправкой:** Если вы используете curl, всегда проверяйте правильность форматирования JSON. В Windows командной строке требуется экранирование двойных кавычек внутри JSON (например, "{\"guid\": \"ваш-guid\"}").
* **Отладка:** Если запрос не возвращает ожидаемых результатов, сначала проверьте, что вы получаете правильный ответ напрямую от целевого API (если доступно тестовое окружение). Используйте логи PM2 или server-side console.log, чтобы видеть, какие запросы обрабатываются.
* **Использование Postman:** Это удобный инструмент для создания, отладки и тестирования API-запросов. С его помощью можно быстро сформировать правильный формат запроса, проверить корректность заголовков и тела запроса, а также проанализировать ответ.
* **Замена параметров:** Всегда заменяйте тестовые значения (например, "ваш-guid", числовые идентификаторы) на реальные значения, предоставленные администратором API.
* **Ошибки и обработка:** Если в ответе от API есть поле ErrorList или параметр Success: false, обратитесь к документации целевого API для понимания возможных ошибок.

# **Заключение**

Этот документ подробно описывает, как отправлять запросы на ваш прокси-сервер, какие параметры включать, как форматировать запросы (с примерами для curl и Postman) и чего ожидать в ответе. Следуя приведённой инструкции, любой пользователь сможет правильно работать с прокси, отладить возможные ошибки и использовать его функционал для интеграции с целевым API.

Если возникнут дополнительные вопросы или потребуется помощь — обратитесь к документации API или свяжитесь с технической поддержкой.