Подача документов на государственную регистрацию в электронном виде.

Описание электронного сервиса.

Сервис предназначен для обеспечения взаимодействия между клиентскими приложениеми и сервером ФНС, предоставляющим функционал для подачи документов на государственную регистрацию в электронном виде.

Данный сервис реализован с помощью архитектурного стиля RESTful.

Адрес сервиса для входа через токен: https://service.nalog.ru/regin/rs

Адрес описания: https://service.nalog.ru/regin/rs/application.wadl

Доступ к сервису возможен как через защищенное соединение с сервером с использованием алгоритмов ГОСТ 28147-89 и ГОСТ Р 34.10-2001 (разрешен только авторизованным пользователям), так и через токен авторизации (см. метод "/auth/signin"). Авторизация производится по сертификату ключа подписи, выданному удостоверяющим центром, аккредитованным в сети доверенных УЦ ФНС России.

Ресурсы и методы сервиса.

Сервис предоставляет доступ к следующим ресурсам и методам:

/file *–Файлы с заявками:*
- ***getFileList()*** *– Получить список заявок*
- ***uploadFile()*** *– Отправить новую заявку на регистрацию*
- ***downloadFile()*** *– Загрузить с сервера файл с заявкой*
- ***getFileInfo()*** *– Получить информацию о результатах обработки заявки*
- ***getReplyList()*** *– Получить список ответов на заявку*
- ***downloadReply()*** *– Загрузить с сервера файл с ответом на заявку*

/gp *– Уплата госпошлины:*
- ***createPlatDoc()*** *– Создать платежный документ*

/pdf *– ФормированиеPDF-документа:*
- ***xmlToZippedPdf()*** *– Преобразовать XML в PDF*

/acc *– Заявления на открытие счёта – реализуется в рамках сервиса «Старт бизнеса онлайн», используется исключительно банками-участниками эксперимента:*
- ***getIncompleteACCList()*** *– Получить список незавершенных заявок на открытие счёта*
- ***getAccList()*** *– Получить список заявок на открытие счёта*
- ***downloadAcc()*** *– Загрузить с сервера файл с заявкой на открытие счёта*
- ***setAccStatus()*** *– Передать* состояние *заявки на открытие счёта*
- ***setEdoOfdStatus()*** *– Передать состояние обработки заявки на ЭДО и ОФД*

/auth *– Авторизация:*
- ***getAuthToken()*** *– Получить токен для авторизации через метод rs/auth/signin*
- ***signIn()*** *– Выполнить авторизацию пользователя по токену и его подписи*

Описания методов.

getFileList() – Получить список заявок

Возвращает список заявок, отправленных пользователем.

Ресурс:	/rs/file
HTTP-Метод:	GET
Параметры:	Параметры строки запроса: fileName = **<Имя файла контейнера>**
Результат (код)	200 OK
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	// Поле FILE_LIST содержит список объектов - транспортных контейнеров
	// Если указан параметр запроса fileName, то список объектов будет содержать только один элемент - контейнер с указанным именем файла (при условии, что такой контейнер был загружен ранее)
	{
	"STATUS": "OK",
	"FILE_LIST": [
	{
	"ID": **<Идентификаторфайла>**,
	"CID": **< Идентификатор транспортного контейнера >**,
	"FILE_NAME": **<Имя файла транспортного контейнера>**,
	"DT": **<Дата/время приема файла>**,
	"STATE_CODE": **<Код состояния>**,
	"STATE": **<Описание состояния>**
	},
	...
	]
	}

Список возможных состояний см. Приложение 1.

uploadFile() – Отправить новую заявку на регистрацию

Загружает на сервер подготовленный ранее файл транспортного контейнера

Ресурс:	/rs/file
HTTP-Метод:	POST
Параметры:	Тип передаваемых данных: multipart/form-data Параметры формы: file = **<Файл транспортного контейнера>**

Имя загружаемого файла проверяется на корректность. Это должен быть ZIP-архив с именем, соответствующим формату транспортного контейнера. Описание формата транспортного контейнера: https://service.nalog.ru/regin/doc/tk_registr_20241015.rar.

Внимание! С момента подписания пакета документов на государственную регистрацию до направления его в регистрирующий орган должно пройти не более 10 суток.

Успешный вызов метода

Результат (код)	201 Created
Результат (тип)	Application/json

Результат (описание):	{
	"STATUS": "OK",
	"ID": **<Идентификатор загруженного файла>**
	}
	Дополнительно, возвращается HTTP-заголовок location, в котором указывается ссылка на созданный ресурс - файл транспортного контейнера.

Внимание! Идентификатор, возвращенный этим методом, в дальнейшем передается в качестве значения /**{id}** при вызове методов *downloadFilegetFileInfogetReplyListdownloadReply*.

Ошибка контроля значений

Результат (код)	400 BadRequest
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	// Поле ERRORS содержит список ошибок
	{
	"STATUS": "Bad Request",
	"ERRORS": {
	<Имя ошибочного параметра>: [
	<Сообщение об ошибке контроля>,
	...
	],
	...
	}
	}

downloadFile() – Загрузить с сервера файл с заявкой

Ресурс:	/rs/file/**{id}**
HTTP-Метод:	GET
Параметры:	(отсутствуют)
Результат (код)	200 OK
Результат (тип)	application/x-zip-compressed
Результат (описание):	Файл транспортного контейнера в формате ZIP-архива.

getFileInfo() – Получить информацию о результатах обработки заявки

Ресурс:	/rs/file/**{id}**/info
HTTP-Метод:	GET
Параметры:	(отсутствуют)
Результат (код)	200 OK
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	// Поле INFO содержит объект - транспортный контейнер
	{
	"STATUS": "OK",
	"INFO": {
	"ID": **<Идентификатор файла>**,
	"CID": **<Идентификатор транспортного контейнера>**,
	"FILE_NAME": **<Имя файла транспортного контейнера>**,
	"DT": **<Дата/время приема файла>**,
	"DT_RO": **<Дата/время получения файла регистрирующим органом>**,
	"RO_CODE": **<Код регистрирующего органа>**,
	"RO_NAME": **<Наименование регистрирующего органа>**,
	"STATE_CODE": **<Код состояния>**,
	"STATE": **<Описание состояния>**,
	"MSG": **<Сообщение проверки>**,
	"NAME": **<Наименование ЮЛ либо ФИО ИП>**
	"REPLY_LIST": [
	{
	"ID": **<Идентификатор файла>**,
	"FILE_NAME": **<Имя файла транспортного контейнера>**,
	"FILE_SIZE": **<Размер файла транспортного контейнера>**,
	"STATE_CODE": **<Код состояния>**,
	"STATE": **<Описание состояния>**
	},
	...
	]
	}
	}

Список возможных состояний см. Приложение 1.

getReplyList() – Получить список ответов на заявку

Ресурс:	/rs/file/**{id}**/reply
HTTP-Метод:	GET
Параметры:	(отсутствуют)
Результат (код)	200 OK
Результат (тип)	application/json

Результат (описание):	`// JSON-объект.`
	`// Поле REPLY_LIST содержит список объектов - транспортных контейнеров`
	`{`
	`  "STATUS": "OK",`
	`  "REPLY_LIST": [`
	`  {`
	`  "ID": <Идентификаторфайла>,`
	`"FILE_NAME": <Имя файла транспортного контейнера>,`
	`  "FILE_SIZE": <Размер файла транспортного контейнера>,`
	`  "STATE_CODE": <Код состояния>,`
	`  "STATE": <Описание состояния>`
	`  },`
	`  ...`
	`  ]`
	`}`

Список возможных состояний см. Приложение 1.

downloadReply() – Загрузить с сервера файл с ответом на заявку

Ресурс:	/rs/file/**{id}**/reply/**{replyId}**
HTTP-Метод:	GET
Параметры:	(отсутствуют)
Результат (код)	200 OK
Результат (тип)	application/x-zip-compressed
Результат (описание):	Файл транспортного контейнера в формате ZIP-архива. Описание формата транспортного контейнера: https://service.nalog.ru/regin/doc/ tk_registr_20241015.rar.

{replyId} - значение, возвращенное при вызове метода ***getReplyList***.

createPlatDoc() – *Создать платежный документ*

Данный метод возвращает платежный документ в формате PDF, со всеми заполненными реквизитами.

Ресурс:	/rs/gp
HTTP-Метод:	POST
Параметры:	Тип передаваемых данных: application/x-www-form-urlencoded Параметры формы: codeno = **<Код налогового органа – получателя платежа>** okato = **<ОКАТО>** kbk = **<КБК>** summa = **<Сумма платежа>** innfl = **<ИННФЛ плательщика>** fam = **<Фамилия плательщика>** nam = **<Имя плательщика>** otch = **<Отчество плательщика>** adrfl = **<Адрес плательщика>**

Передаваемые значения контролируются на корректность. Например, «ОКАТО» должен состоять из 11 цифр, а «КБК» из 20 цифр. «Отчество» является необязательным параметром, а «ИННФЛ» и «Адрес» являются условно-обязательными, т.е. необходимо заполнить хотя бы один из этих параметров.

Успешный вызов метода

Результат (код)	200 OK
Результат (тип)	application/pdf
Результат (описание):	{"status": "OK"}

Ошибка контроля значений

Результат (код)	400 BadRequest
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	// Поле ERRORS содержит список ошибок
	{
	"STATUS": "Bad Request",
	"ERRORS": {
	<Имя ошибочного параметра>: [
	<Сообщение об ошибке контроля>,
	...
	],
	...
	}
	}

xmlToZippedPdf() – Преобразовать XMLв PDF

Загружает на сервер XML по формату одной из форм регистрации и формирует печатную форму в формате PDF-документа, упакованного в ZIP-архив. При поступлении XML формы Р11001 на регистрацию ООО с единственным учредителем - ФЛ дополнительно формируется и вкладывается в ZIP-архив решение о создании ЮЛ.

Ресурс:	/rs/pdf
HTTP-Метод:	POST
Параметры:	Тип передаваемых данных: application/xml

XML должен соответствовать формату одной из форм регистрации. Доступные формы: Р11001, Р12003, Р12016, Р13014, Р15016, Р18002, Р19001, Р21001, Р24001, Р26001, Р34001, Р34002, Р38001.

Успешный вызов метода

Результат (код)	200OK
Результат (тип)	application/x-zip-compressed
Результат (описание):	ZIP-архив, содержащий файл печатной формы заявления в формате PDF-документа, при выполнении условий - файл печатной формы решения о создании ЮЛ в формате PDF-документа и подписи к ним. Правила именования файлов в архиве приведены в Приложении 2.

Ошибка контроля значений

Результат (код)	400 BadRequest
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	// Поле ERRORS содержит список ошибок
	{
	"STATUS": "Bad Request",
	"ERRORS": {
	<Имя ошибочного параметра>: [
	<Сообщение об ошибке контроля>,
	...
	],
	...
	}
	}

getIncompleteACCList() – Получить список незавершенных заявок на открытие счёта

Возвращает список заявок, подготовленных пользователем (отправленных на регистрацию ИП/ЮЛ).

Ресурс:	/rs/acc/prev
HTTP-Метод:	GET
Параметры:	dateTime - дата и время, с которого отбираются заявки (если пусто, или ранее 10 суток, берутся за 10 суток). Указывается в одном из форматов: "dd.MM.yyyyHH:mm:ss", "yyyy-MM-dd HH:mm:ss", "yyyy-MM-dd'T'HH:mm:ss", "yyyy-MM-dd", "dd.MM.yyyy"
Результат (код)	200 OK
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	// Поле ACC_LIST содержит список строк - идентификаторов транспортных контейнеров
	{
	"STATUS": "OK",
	"PREV_LIST": [
	<Данные заявки>,
	...
	]
	}
	<Данные заявки>- JSON объект с набором атрибутов:
	ACC_ID - Идентификатор заявки для банка (тот же, что возвращает метод getAccList()), присутствует только в случае успешного проведения рег. действий
	GOSREG_NUM - номер заявки в сервисе госрегистрации
	REG_NUM - номер заявки в регоргане (UID ТК)
	SFRD - код формы: P21001 - ИП; P11001 - ЮЛ
	BNK_NAME - наименование банка
	BNK_INN - ИНН банка
	STATUS_CODE - код статуса заявки
	STATUS - статус заявки
	TK_DATE - дата формирования ТК
	RO_DATE - дата поступления заявки в регорган
	CONFIRM_DATE - дата подтверждения заявки регорганом
	REG_DATE - дата регистрации
	STATUS_DATE - дата последнего статуса
	SURNAME - фамилия заявителя
	NAME - имя заявителя
	PATRONYMIC - отчество заявителя
	FL_INN - ИНН заявителя
	OGRNIP - ОГРНИП ИП
	UL_OGRN - ОГРН ЮЛ
	UL_INN - ИНН ЮЛ
	UL_NAME - наименование ЮЛ
	PHONE - номер телефона заявителя
	UKEP_DATE - дата обращения к сервису выдачи сертификата УКЭП
	EBS - признак наличия ЕБС: 0 - нет информации; 1 - есть; 2 - нет
	REGION - код региона открытия счета
	QES_DT - дата и время последнего события от ВСЭП
	QES_STATUS-последнийстатусот ВСЭП
	QES_MESSAGE-последнеесообщениеот ВСЭП

	MOBILE_SIGN_STATUS-статус подписания мобильной подписью: 000 - создана заявка на подписание; 010 - заявка отправлена; 050 - подписание успешно EDO - признак подключения к электронному документообороту: 0 - нет, 1 -да OFD - признак подключения к ОФД: 0 - нет, 1 -да EMAIL - адресэлектроннойпочты

Пример ответа	{
	"STATUS": "OK",
	"PREV_LIST": [
	{
	"GOSREG_NUM": 74902,
	"REG_NUM": "U6240118514356",
	"SFRD": "P11001",
	"BNK_NAME": "ПУБЛИЧНОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО \"СБЕРБАНК РОССИИ\"",
	"BNK_INN": "7707083893",
	"STATUS_CODE": "A70",
	"STATUS": "ТК получен банком",
	"TK_DATE": "18.01.2024 12:09:00",
	"RO_DATE": "18.01.2024 13:23:33",
	"REG_DATE": "18.01.2024 16:14:55",
	"STATUS_DATE": "20.03.2024 15:24:05",
	"SURNAME": "СИНИЦЫНА",
	"NAME": "ТАТЬЯНА",
	"PATRONYMIC": "МИХАЙЛОВНА",
	"FL_INN": "520800010597",
	"UL_OGRN": "1247700000020",
	"UL_INN": "7703502254",
	"UL_NAME": "ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ \"БИЗНЕС 2\"",
	"PHONE": "+12345678901",
	"EBS": "0"
	},
	{
	"GOSREG_NUM": 79803,
	"REG_NUM": "F6240313519556",
	"SFRD": "P21001",
	"BNK_NAME": "ПУБЛИЧНОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО \"СБЕРБАНК РОССИИ\"",
	"BNK_INN": "7707083893",
	"STATUS_CODE": "A03",
	"STATUS": "Требуется получение сертификата УКЭП",
	"TK_DATE": "13.03.2024 18:35:23",
	"RO_DATE": "13.03.2024 18:46:57",
	"REG_DATE": "13.03.2024 19:53:58",
	"STATUS_DATE": "13.03.2024 19:54:12",
	"SURNAME": "КРЕМИНСКИЙ",
	"NAME": "АЛЕКСЕЙ",
	"PATRONYMIC": "СТАНИСЛАВОВИЧ",
	"FL_INN": "971141574601",
	"OGRNIP": "324774600000295",
	"PHONE": "+79097768765",
	"EBS": "0"
	}
	]
	}

getAccList() – Получить список заявок на открытие счёта

Возвращает список заявок, отправленных пользователем.

Ресурс:	/rs/acc
HTTP-Метод:	GET
Параметры:	(отсутствуют)
Результат (код)	200 OK
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	// Поле ACC_LIST содержит список строк - идентификаторов транспортных контейнеров
	{
	"STATUS": "OK",
	"ACC_LIST": [
	<Идентификатор файла>,
	...
	]
	}

downloadAcc() – Загрузить с сервера файл с заявкой на открытие счёта

Ресурс:	/rs/acc/**{id}**
HTTP-Метод:	GET
Параметры:	(отсутствуют)
Результат (код)	200 OK
Результат (тип)	application/x-zip-compressed
Результат (описание):	Файл транспортного контейнера в формате ZIP-архива.

setAccStatus() – Передать состояние заявки на открытие счёта

Позволяет передать состояние обработки заявки на открытие счета в банке:

- *10 – заявка принята в банке (передается банком в подтверждение получения заявки);*
- *11 – требуются дополнительные действия по идентификации клиента (может быть передан следующий статус);*
- *12 – ошибка проверки УКЭП заявления об открытие счета, требуется повторное подписание (передается банком)*
- *20 – счет открыт (дальнейший прием статусов прекращается);*
- *30 – счет не может быть открыт (дальнейший прием статусов прекращается).*
- *31 – истек срок открытия счета (дальнейший прием статусов прекращается).*

Ресурс:	/rs/acc/**{id}**
HTTP-Метод:	POST
Параметры:	Параметры формы: stateCode= **<Код состояния>** state= **<Описание состояния>**
Результат (код)	200 OK
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	{
	"STATUS": "OK"
	}

setEdoOfdStatus() – Передать состояние обработкизаявки на ЭДО и ОФД

Позволяет передать состояние обработки заявки на ЭДО и ОФД:

stateCode - два знака; первый -статус ЭДО, второй - статус ОФД:

- *0 – статуснеменяется*
- *1 – заявка на услугу в обработке*
- *2 - заключен договор*
- *3 – в договоре на услугу отказано*
- *4 - пользователь отказался от услуги*

Ресурс:	/rs/edo-ofd/**{id}**
HTTP-Метод:	POST
Параметры:	Параметры формы: stateCode= **<Код состояния>**
Результат (код)	200 OK
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	{
	"STATUS": "OK"
	}

getAuthToken(): Получить токен авторизации

Возвращает токен авторизации, необходимый для для авторизации пользователя по токену и подписи (метод /rs/auth/signin).

Ресурс:	/rs/auth/token
HTTP-Метод:	GET
Параметры:	(отсутствуют)

Успешный вызов метода

Результат (код)	200 OK
Результат (тип)	application/json

Результат (описание):	{
	  "status": "OK",
	  "expires_in": 1800000,
	  "type": "AuthToken",
	  "token": **<Токен авторизации>**
	}
	<expires_in> - время в миллисекундах, через которое истечет срок действия токена. Например, 1800000 мс = 30 минут.
	<type> - Тип токена.

signIn(): Выполнить авторизацию пользователя по токену и его подписи

Выполняет авторизацию пользователя, необходимую для работы с сервисом. Результат вызова сервиса возвращает токен авторизации, который в дальнейшем необходимо передавать во все методы сервиса в заголовке **"Authorization: RiToken<Токен, полученный в методе /rs/auth/signin>"**

Ресурс:	/rs/auth/signin
HTTP-Метод:	POST
Параметры:	Тип передаваемых данных: text/xml Параметры формы: token = **<Токен авторизации, полученный через методо /rs/auth/token>** sign = **<Электронно-цифровая подпись токена авторизации в формате CAdES-BES(форматPKCS#7), закодированная в формате BASE64>**

Успешный вызов метода

Результат (код)	200 OK
Результат (тип)	application/json

Результат (описание):	{
	"STATUS": "OK",
	"expires_in": 86400000,
	"type": "RiToken",
	"token": **<Токен авторизации>**
	}
	<expires_in> - время в миллисекундах, через которое истечет срок действия токена. Например, 86400000мс = 24 часа.
	<type> - Тип токена.

Ошибка авторизации

Результат (код)	Код ошибки
Результат (тип)	application/json

Результат (описание):	// JSON-объект.
	{
	"status": **<Статус-сообщение>**,
	"error": **<Сообщение об ошибке>**
	}

Пример авторизации пользователя:

1) Получение токена авторизации, вызов метод "/rs/auth/token":

Запрос:

GEThttps://service.nalog.ru/regin/rs/auth/token

HTTP/1.1

Accept-Encoding: gzip,deflate

Host: service.nalog.ru

Connection: Keep-Alive

```
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

Ответ:

```
HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Content-Type: application/json;charset=UTF-8

Transfer-Encoding: chunked

Date: Mon, 01 Jan 2019 04:42:34 GMT

{"STATUS":"OK","expires_in":1800000,"type":"AuthToken","token":"RjVGM0NBQkRENTQyM0
M4MkRENUU1M0M1MzExQjJGQzQ2MjU0OTcxRThCNEJDNjA2Mjg4MkRCNDgyQTlGRkI1MjBBNUFCNTlENzk0
MkFDRkFDQjRFOUQwMTgzMkUzMkU3NEI0QUU2OTg4RENEQjcxMjVFMTk3NjIyQjg0QzQzNzk\u003d"}
```

2) Получив токен авторизации, его необходимо подписать в формате CadES-BES (формат PKCS#7) и отправить вместе с подписью в качестве параметров метода "/rs/auth/signin":

При подписании токена, его необходимо декодировать из кодировки BASE64, далее уже подписывать декодированный токен, иначе подпись будет неверна.

Общий алгоритм подписания:

а) Декодировать токен из кодировки BASE64.

б) Подписать декодированный токен в формате CAdES-BES(формат PKCS#7) .

в) Закодировать подпись в кодировке BASE64.

Запрос:

```
POSThttp://service.nalog.ru/regin/rs/auth/signin?token=RjVGM0NBQkRENTQyM0M4MkRENUU
1M0M1MzExQjJGQzQ2MjU0OTcxRThCNEJDNjA2Mjg4MkRCNDgyQTlGRkI1MjBBNUFCNTlENzk0MkFDRkFDQ
jRFOUQwMTgzMkUzMkU3NEI0QUU2OTg4RENEQjcxMjVFMTk3NjIyQjg0QzQzNzk=&sign=MIIFXgYJKoZIh
vcNAQcCoIIFTzCCBUsCAQExDDAKBgYqhQMCAgkFADALBgkqhkiG9w0BBwGgggPyMIID7jCCA52gAwIBAgI
KR9qkkAACAAB1GTAIBgYqhQMCAgMwgcAxHjAcBgkqhkiG9w0BCQEWD3VjaW5mb0Bnbml2Yy5ydTELMAkGA
1UEBhMCUlUxFTATBgNVBAcMDNCc0L7RgdC60LLQsDEwMC4GA1UECgwn0KTQk9Cj0J8g0JPQndCY0JLQpiD
QpNCd0KEg0KDQvtGB0YHQuNC4MTAwLgYDVQQLDCfQo9C00L7RgdGC0L7QstC10YDRj9GO0YnQuNC5INGG0
LXQvdGC0YAxFjAUBgNVBAMTDUdOSVZDIEZOUyBSVVMwHhcNMTgxMjA2MDYzMTUyWhcNMTkxMjA2MDY0MTU
yWjCBpzEYMBYGBSqFA2QBEw0xMDIwMjAyODU2Nzk0MRgwFgYIKoUDA4EDAQETCjAyNzYwNjAyNzkxCzAJB
gNVBAYTAlJVMRUwEwYDVQQIHgwEHAQ+BEEEOgQyBDAxFTATBgNVBAceDAQcBD4EQQQ6BDIEMDEfMB0GA1U
ECh4WBBcEEAQeACAELgQdBBgEIQQeBCQEIjEVMBMGA1UEAxMMOTQuMTI1LjkwLjUwMGMwHAYGKoUDAgITM
BIGByqFAwICJAAGByqFAwICHgEDQwAEQMc+/GdPNHP864tgkNnvIkMdDno02aWmPO7LMoJBaeEYQwdhGQw
bP2yvtsvxssO/M3x2+pgALrRIt0iiFzHymBajggGMMIIBiDAOBgNVHQ8BAf8EBAMCBPAwGQYJKoZIhvcNA
QkPBAwwCjAIBgYqhQMCAhUwJwYDVR0lBCAwHgYIKwYBBQUHAwQGCCsGAQUFBwMCBggrBgEFBQcDATAdBgN
VHQ4EFgQUjdyAq+VWcitSiVbHX7cKLQghA+swHwYDVR0jBBgwFoAUj69gHJrhX6V7ItkrU5iyCJmNNQkwb
AYDVR0fBGUwYzBhoF+gXYYraHR0cDovL3d3dy5nbml2Yy5ydS91Yy9HTklWQ0ZOU1JVU18yMDExLmNybIY
uaHR0cDovL2MwMDAwLWFwcDAwNS9nbml2Yy9HTklWQ0ZOU1JVU18yMDExLmNybDCBgwYIKwYBBQUHAQEEd
zB1MDcGCCsGAQUFBzAChitodHRwOi8vd3d3LmduaXZjLnJ1L3VjL0dOSVZDRk5TUlVTXzIwMTEuY3J0MDo
GCCsGAQUFBzAChi5odHRwOi8vYzAwMDAtYXBwMDA1L2duaXZjL0dOSVZDRk5TUlVTXzIwMTEuY3J0MAgGB
iqFAwICAwNBACOKw7oIrdts9ANUuMwwImHtUu4e57dMNI/YYKrXkwmNp0aJ/iowscHmDwkOMp4tK8UpzIC
PgEVa0w2feWD8UzIxggEzMIIBLwIBATCBzzCBwDEeMBwGCSqGSIb3DQEJARYPdWNpbmZvQGduaXZjLnJ1M
QswCQYDVQQGEwJSVTEVMBMGA1UEBwwM0JzQvtGB0LrQstCwMTAwLgYDVQQKDCfQpNCT0KPQnyDQk9Cd0Jj
QktCmINCk0J3QoSDQoNC+0YHRgdC40LgxMDAuBgNVBAsMJ9Cj0LTQvtGB0YLQvtCy0LXRgNGP0Y7RidC40
Lkg0YbQtdC90YLRgDEWMBQGA1UEAxMNR05JVkMgRk5TIFJVUwIKR9qkkAACAAB1GTAKBgYqhQMCAgkFADA
KBgYqhQMCAhMFAARAmwVD3UAQpX3FcCOhKQPi05Vnwdv7RZDlvnTScZ+vOiiu6xhNc+zL20ZXVEdCbPbuh
FGiuInSn0P7FT4+PgeEkQ==

HTTP/1.1

Accept-Encoding: gzip,deflate

Content-Type: text/xml

Content-Length: 0
```

```
Host: service.nalog.ru

Connection: Keep-Alive

User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

Ответ:

```
HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Set-Cookie: JSESSIONID=25C009487BAE4179B721BEB35B2B18E5; Path=/regin/; HttpOnly

Set-Cookie: RiUserLogon=9920acb4-18d4-41bf-afc3-8d73cb61fe6d; Expires=Fri, 15-Feb-2019 07:25:16 GMT; Path=/

Content-Type: application/json;charset=UTF-8

Transfer-Encoding: chunked

Date: Mon, 01 Jan 2019 04:43:29 GMT

{"STATUS":"OK","expires_in":3600000,"type":"RiToken","token":"RUVGOTdEMzM4RjAyQjNB
MTI4RTBDOTFFNzJFRURFOUFEMTI2QjkzREIzQzFEOERFOTE2MjczMDdBMEM1NjExODZFMTgyMUVGQjNCQT
AwNjQyRUFCMThFNUZGRUNGRkIzNjU0NDQ4REZCQUQ4ODE3RjhDMzUxOTVFREJEMzAyQkRFMDhGRUNEQURC
MzFBQzc1RDcyNjJBRkMwM0ZDMzhBRDVCODBFRjg0Mzg4RENEOUM2QTFEMTRGQzVBNDRBMjBEOTA3RTVDMD
JBQjNDQkQzQjMyN0UwQUVCODk0NUFGODZCNzQyRjk2Q0JCMURFMzQxMTRGMjc2QTRFQzU2M0YxMkU0NUU1
QzA2MDRBMUFDNDkzMzlCRjg5NjQ3OEE3RUZERDkyNDkzMTNFOTRDMTk3MTMyOEYxOTU0RTE5M0NGNDFGNE
JDRTNCNUJGNzAxNTJEQzgzNkVCRENGQ0Q0NjA0Njc4NzBDODk3Q0Q3QkJGNTlFNzVBRkU5QjU3NTA4MjlD
QTBGQ0M4MEQ0MjVCQUMyNTk4OTU0ODVCNEJFMzAxNEEwRjQ4NUU3NkUyMDQ1NzkzQUIzODM0RTEzQUVCOD
M4MDVFQ0JGMjY1NTZCMjkwRjEzQjIyMDlBQUE4NDhCMEZDQUEzMjY4MDE3ODNERjFFRUU0NTM3Rjk3QzI2
RTExQjYwMDE5QTQ4QjdDNkY2QkM1MjNDMkRCNDdEOUM2RDE0NDJDQUVCRDM1N0M5MjkwMEMyRkE2OEJDOD
IyN0NDRjVGRUUzNzQ1RTc4NzUzQzExMDVGOTFERDhBQ0RGM0RCMURBRUUxRkE5RjEyREVBRjUwQjVEMjkz
ODMzMTM1MDRENUYxMzBEM0FGNUZFNDg3RjlENUYwNDE1QUZDN0FBRjI2QjdGQ0Q1QjE5Nzc2MzZFODk1Mj
Y0OTZCRUIzMDNFNDY2QUY3RUFGQ0M4MzEyOTI1RTg4MjA3QTQ1QTc2OEI1MUExNjk1QzM0NDZCMTVDMkRB
NUMwMDUzRjRCQ0ZGMkFDQkNCRDg4MTUyQkIyNjJERThCMUM4OEJCNzFBQUM0OTkyQjAxNzhGNTI5RDJDQz
AxNzRCMDYwMzIzRDA5NDQyMQ\u003d\u003d"}
```

3) Далее все методы сервиса необходимо вызывать с передачей токена авторизации заголовке ("Authorization: RiToken<Токен, полученный в методе /rs/auth/signin>"), например:

```
GET https://service.nalog.ru/regin/rs/file/12345/reply HTTP/1.1

Accept-Encoding: gzip,deflate

Authorization: RiToken
RUVGOTdEMzM4RjAyQjNBMTI4RTBDOTFFNzJFRURFOUFEMTI2QjkzREIzQzFEOERFOTE2MjczMDdBMEM1Nj
ExODZFMTgyMUVGQjNCQTAwNjQyRUFCMThFNUZGRUNGRkIzNjU0NDQ4REZCQUQ4ODE3RjhDMzUxOTVFREJE
MzAyQkRFMDhGRUNEQURCMzFBQzc1RDcyNjJBRkMwM0ZDMzhBRDVCODBFRjg0Mzg4RENEOUM2QTFEMTRGQz
VBNDRBMjBEOTA3RTVDMDJBQjNDQkQzQjMyN0UwQUVCODk0NUFGODZCNzQyRjk2Q0JCMURFMzQxMTRGMjc2
QTRFQzU2M0YxMkU0NUU1QzA2MDRBMUFDNDkzMzlCRjg5NjQ3OEE3RUZERDkyNDkzMTNFOTRDMTk3MTMyOE
YxOTU0RTE5M0NGNDFGNEJDRTNCNUJGNzAxNTJEQzgzNkVCRENGQ0Q0NjA0Njc4NzBDODk3Q0Q3QkJGNTlF
NzVBRkU5QjU3NTA4MjlDQTBGQ0M4MEQ0MjVCQUMyNTk4OTU0ODVCNEJFMzAxNEEwRjQ4NUU3NkUyMDQ1Nz
kzQUIzODM0RTEzQUVCODM4MDVFQ0JGMjY1NTZCMjkwRjEzQjIyMDlBQUE4NDhCMEZDQUEzMjY4MDE3ODNE
RjFFRUU0NTM3Rjk3QzI2RTExQjYwMDE5QTQ4QjdDNkY2QkM1MjNDMkRCNDdEOUM2RDE0NDJDQUVCRDM1N0
M5MjkwMEMyRkE2OEJDODIyN0NDRjVGRUUzNzQ1RTc4NzUzQzExMDVGOTFERDhBQ0RGM0RCMURBRUUxRkE5
RjEyREVBRjUwQjVEMjkzODMzMTM1MDRENUYxMzBEM0FGNUZFNDg3RjlENUYwNDE1QUZDN0FBRjI2QjdGQ0
Q1QjE5Nzc2MzZFODk1Mjc0OTZCRUIzMDNFNDY2QUY3RUFGQ0M4MzEyOTI1RTg4MjA3QTQ1QTc2OEI1MUEx
Njk1QzM0NDZCMTVDMkRBNUMwMDUzRjRCQ0ZGMkFDQkNCRDg4MTUyQkIyNjJERThCMUM4OEJCNzFBQUM0OT
kyQjAxNzhGNTI5RDJDQzAxNzRCMDYwMzIzRDA5NDQyMQ==

Host: service.nalog.ru

Connection: Keep-Alive

User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

Приложение 1.

Список статусов заявки на регистрацию

0 Идет предварительная проверка

10 Заявка поставлена в очередь на обработку

19 Ошибка адресации сведений (в описи документов указан некорректный код РО)

20 Заявка не может быть выполнена (выявлены несоответствия ТК требованиям)

21 Заявка не может быть выполнена в РО (выявлены несоответствия заявления и приложенных документов требованиям)

23 Срок жизни заявки истек

30 Документы приняты в регистрирующий орган (отправлена расписка)

40 Принято решение об отказе в регистрации

50 Запись о регистрации внесена в реестр

99 Контейнер загружен не полностью

41 В выполнении заявки отказано

48 Принято решение о приостановлении государственной регистрации

49 Направлено уведомление о дате прихода заявителя

60 Заявка аннулирована из-за неявки заявителя в указанный срок

Приложение 2.

Правила именования файлов в архиве сформированных документов.

из переданного XML берется значение атрибута ИдФайл (далее обозначается {ИдФайл})

в архив вкладываются:

PDF файл заявления с именем {ИдФайл}.pdf

подпись к нему с именем {ИдФайл}.p7s

если формируется решение о создании ЮЛ, то

в значении {ИдФайл} префикс (первые 9 символов) заменяются на 021023 (далее обозначается {ИдРеш})

дополнительно вкладывается

PDF файл решения с именем {ИдРеш}.pdf

подпись к нему с именем {ИдРеш}.p7s

При наличии вкладывается файл протокола ФЛК в формате JSON с именем {ИдФайл}.json

Пример:

ИдФайл="RO_R11001_0000_0000_1066111000449_20260219_12B187CAFF334E85A5B3D7C54C6DB1CA"

В архиве будут файлы:

RO_R11001_0000_0000_1066111000449_20260219_12B187CAFF334E85A5B3D7C54C6DB1CA.pdf

RO_R11001_0000_0000_1066111000449_20260219_12B187CAFF334E85A5B3D7C54C6DB1CA.p7s

021023_0000_0000_1066111000449_20260219_12B187CAFF334E85A5B3D7C54C6DB1CA.pdf

021023_0000_0000_1066111000449_20260219_12B187CAFF334E85A5B3D7C54C6DB1CA.p7s

RO_R11001_0000_0000_1066111000449_20260219_12B187CAFF334E85A5B3D7C54C6DB1CA.json