## Изисквания към интеграцията на системи в рамките на електронното управление

# Съдържание

1. [Интерфейси](#1-Интерфейси)
2. [Сигнализиране за грешки](#2-Сигнализиране-за-грешки)
3. [Диаграма на действията](#3-Диаграма-на-действията)
4. [Електронна идентификация и журнал на действията от страна на АИС](#4-Електронна-идентификация-и-журнал-на-действията-от-страна-на-АИС)
5. [Допълнителни изисквания към първичните регистри](#5-Допълнителни-изисквания-към-първичните-регистри)
6. [Описание на централния координатор](#6-Описание-на-централния-координатор)
7. [Описание на централния регистър](#7-Описание-на-централния-регистър)

Този документ предсдтавя изискванията към интеграцията на системи, включвайки конкретни интерфейси и формати на заявки и отговори.

### 1. **Интерфейси**

Функционалността на приложенията трябва да е достъпна и през уеб-интерфейси.

Следват конкретни интерфейси и структури, които първичните регистри и консуматорите (АИС-и) трябва да предоставят и извикват.

Всички примери са дадени в JSON формат, но XML със същата структура също следва да се поддържа.

Разграничаването на JSON от XML следва да става с използване на Accept HTTP header: съответно application/json или text/xml.

**За първични регистри:**

Първичните регистри следва да предоставят следните интерфейси:

Заб: препоръчва се адресът на API-то да бъде достъпно чрез поддомейн api.<url>/. Параметърът {version} показва версията, като първоначално тя е 1.

1. GET /api/{version}/docs - документация на всички предоставени уеб-услуги (вж. секция „Документация на интерфейсите“ в изискванията към качеството). Всеки първичен регистър трябва да има стандартните, описани по-долу уеб-услуги. Както те, така и всички останали уеб-услуги, трябва да бъдат детайлно документирани - адрес, метод, параметри, поддържани сериализационни формати, начин на извикване (синхронно, асинхронно, абонамент), очаквано време (timeout) за асинхронни заявки и т.н.
2. GET /api/{version}/data - списък с данни, които предоставят регистърът. Съществуват предефинирани основни типове, като “имена” или “адрес”. Вж. т.4. Например:

* {  
   "data": [  
   {  
   "key": "names",  
   "description": "Три имена",  
   "version": 1,  
   "type": "Names",  
   "idTypes": [ "ЕГН", "ЛНЧ" ]  
   },  
   {  
   "key": "current\_address",  
   "description": "Настоящ адрес",  
   "version": 2,  
   "type": "Address",  
   "idTypes": [ "ЕГН", "ЛНЧ" ]  
   },  
   {  
   "key": "current\_address",  
   "description": "Настоящ адрес",  
   "version": 1,  
   "type": "Address",  
   "idTypes": [ "ЕГН", "ЛНЧ" ],  
   "deprecated": true  
   }  
   ]  
  }

1. GET /api/{version}/services - връща списък на уеб-услуги, които предоставя АИС-ът, с техните описания (XSD/json schema)
2. GET /api/{version}/dataType/{key} - използвайки Json-schema ([*http://json-schema.org/examples.html*](http://json-schema.org/examples.html)) или XSD, дефинира типовете структурите на данните, реферирани в полето type на горния списък. Параметърът “key” отговаря на стойността в type. Тези типове се извличат автоматично от централния регистър и стават де-факто “регистър на информационните обекти” според ЗЕУ.
3. GET /api/{version}/idTypes - дава списък с поддържане типове идентификатор (на физически лица, юридически лица, автомобили и т.н.):

* {  
   "idTypes" : [  
   {  
   "idType": "ЕГН",  
   "description": "Единен граждански номер на физически лица"  
   }  
   ]  
  }

1. POST /api/{version}/requestData/<txId> - заявка за данни.
   * txId е идентификатор на транзакция, който следва да бъде проверен в централния компонент
   * тялото на заявката следва да съдържа следното:

{  
 "clientId" : "5c24acb8-e001-4d92-bb0c-5a02072ca1e9",  
 "destinationId" : "b25b24c2-17f2-4ac3-885e-261506a8c693",  
 "requestType" : "SYNCHRONOUS",  
 "endpointType" : "DATA",  
 "requestedDataFields" : [  
 {  
 "key": "names",  
 "version": 1  
 },  
 {  
 "key": "current\_address",  
 "version": 1  
 }  
 ],  
 "idType" : "ЕГН",  
 "id" : "8112120511",  
 "requestingPerson" : "Иван Иванов",  
 "requestingPersonAuthenticationToken" : "a43vsfda5342",  
 "serviceId": "123445"  
}

* Проверката на заявка в централния компонент се извършва, като се изпрати заявка към него, съдържаща полученото тяло на заявката от заявителя към

POST https://<central-service-URL>/api/{version}/transaction/verify?txId=<txId>

* При успешна верификация, централният компонент отговаря с true и първичният администратор на данни следва да предостави отговор на заявителя.
* Типове заявки:
  + SYNCHRONOUS - изисква се синхронен отговор
  + ASYNC - изисква се асинхронен отговор да бъде изпратен на адрес, посочен в полето callbackUrl.
  + SUBSCRIBE - всяка промяна в заявените данни следва да бъде изпращане на адрес, посочен в полето callbackUrl
  + UNSUBSCRIBE - отказ от заявен SUBSCIRBE
* Структурата на синхронния отговор, както и на асинхронните отговори, изпратени към callbackUrl следва да бъде:

{  
 "txId": "5c24acb8-e001-4d92-bb0c-5a02072ca1e9",  
 "timestamp": "2016-01-01T17:43:19+02:00",  
 "success": true,  
 "data": [  
 {  
 "key": "names",  
 "value": {  
 "firstName": "Петър",  
 "lastName": "Петров"  
 }  
 }  
 ]  
}

* Известия за промяна при заявени данни с тип SUBSCRIBE следва да се изпращат на callbackUrl със следния формат”

{  
 "requestTxId": "5c24acb8-e001-4d92-bb0c-5a02072ca1e9",  
 "timestamp": "2016-01-01T17:43:19+02:00",  
 "dataChanged": "current\_address"  
}

Заявителят следва да разбере за кой идентификатор (напр. ЕГН) се отнася известието на база на requestTxId, което следва да пази в системата си при SUBSCRIBE заявки.

**За АИС (консуматор)**

Консуматорът поддържа само два интерфейса, на дефиниран от него адрес.

1. интерфейс за приемане на асинхронен отговор (вж. 1.1.6, типове заявки)
2. интерфейс за приемане на известия за промени на данни (вж. 1.1.6, типове заявки)

Преди изпращане на заявка до първичен регистър, всеки консуматор трябва да създаде транзакция, извиквайки <координатор>/transaction/create Примерна заявка:

{  
 "clientId" : "5c24acb8-e001-4d92-bb0c-5a02072ca1e9",  
 "destinationId" : "b25b24c2-17f2-4ac3-885e-261506a8c693",  
 "destinationServiceId" : "b25b24c2-17f2-4ac3-885e-261506a8c693",  
 "requestType" : "SYNCHRONOUS",  
 "endpointType" : "DATA",  
 "requestedDataFields" : [ "names", "address" ],  
 "idType" : "ЕГН",  
 "id" : "8112120511",  
 "requestingPerson" : "Иван Иванов",  
 "requestingPersonAuthenticationToken" : "a43vsfda5342",  
 "serviceId": "123445"  
}

Необходимо е и да бъдат предоставени два HTTP header-a:

* Signature - електронно подписаното тяло на заявката
* Signature-Algorithm - алгоритъмът, използван за съставянето на Signature, напр. SHA256withRSA

Полетата за всяка заявка за данни са:

Задължителни:

* clientId - идентификатор на администрацията-заявител (от административния регистър)
* destinationId - идентификатор на администрацията, предоставяща данните/услугата (от административния регистър)
* destinationServiceId - идентификатор на първичния регистър / АИС, предоставящ данните/услугата. Всеки АИС (вкл. регистър) избира собствен идентификатор и го публикува в централния компонент. АИС-заявител може или да конфигурира предварително този индентификатор (вземайки го от списъка на централния регистър), а може да го заявява периодично, питайки централния регистър “кой е идентификатора на регистъра, който предодставя данна X”
* requestType - тип на заявката - синхронна/асинхронна/заявка за уведомяване
* endpointType - тип на крайната точка на транзакцията - дали предоставя данни (/requestData) (DATA) или е пълноправна услуга (SERVICE)
* requestedDataFields - данните, които се заявяват, като key съдържа името на данната, а version – версията, която се заявява
* idType - типът на използвания идентификатор, поддържан от първичния администратор (/api/idTypes)
* id - идентификатора на лицето/автомобила/имота/и т.н., за които се искат данни
* originatingDocumentId - идентификатор на документ (преписка) на заявителя, на база на който се изисква достъпа до данните

Опционални:

* serviceId - идентификатор на услугата (от регистъра на услугите), за изпълнението на която са нужни данните.
* legalReason - правно основание за заявяването на данните, когато те не са заявени за конкретна услуга
* originatingDocumentType - тип на документ на заявителя, на база на който се изисква достъпа до данните, в свободен текст, дефиниран от АИС-заявител.
* requestingPerson - имената на длъжностното лице, което инициира заявката. В случай на автоматизирана заявка не се попълва. Вж. т.10.
* requestingPersonAuthenticationToken - token/идентификатор, издаден от валидиращият орган за електронна идентификация (Identity Provider), по който може да бъде идентифицирано длъжностното лице, извършило заявката. В случай на автоматизирана заявка не се попълва. Вж. т.10.
* callbackUrl - адрес (URL), на който да бъде изпратен отговора на ASYNC или SUBSCRIBE заявка

### 2. **Сигнализиране за грешки**

Всяка заявка към приложно-програмния интерфейс на системата се смята за успешно изпълнена, освен ако HTTP Status кодът на отговора ѝ не е код за грешка (4XX или 5XX).

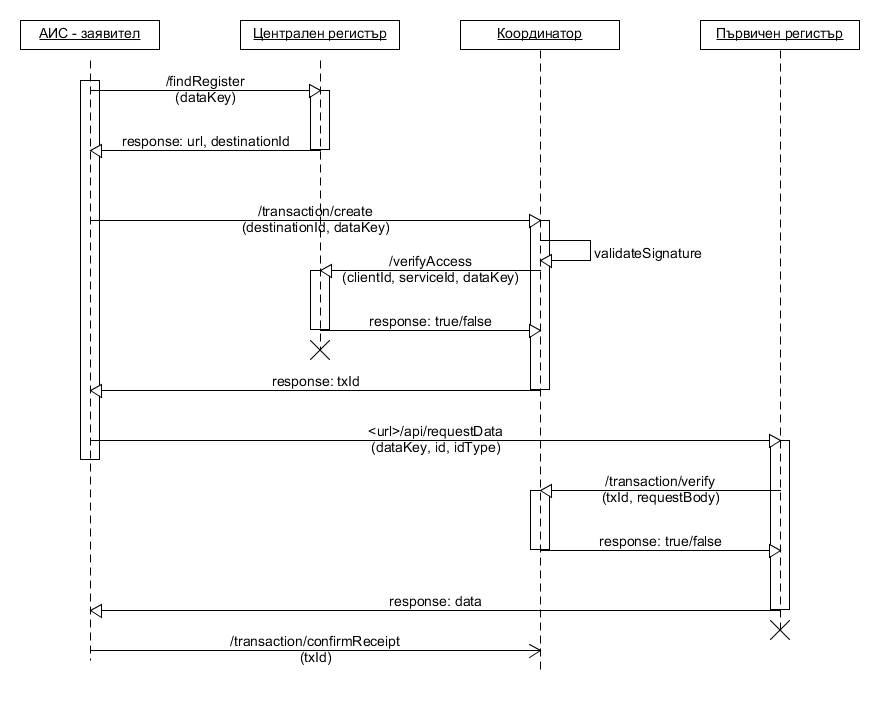
При възникване на грешка по време на заявка, системата трябва да върне подходящ HTTP Status код и в тялото на отговора трябва да присъства следната структурирана информация за възникналата грешка (id и description са задължителни):

{  
 "id": "<идентификатор на грешката – числов или символен>",  
 "description ": "подробно и лесно за разбиране обяснение в какво се  
изразява грешката",  
 "userMessage": "кратко обяснение на грешката, подходящо запоказване на краен потребител",  
 "details ": {  
 "<идентификатор на обект>": [  
 "информация за грешка 1",  
 "информация за грешка 2",  
 ],  
 ...  
 }  
}

При грешка в отговора на асинхронна заявка (ASYNC), регистърът изпраща следното съобщение към callbackUrl-а:

{  
 "txId": "5c24acb8-e001-4d92-bb0c-5a02072ca1e9",  
 "success": false,  
 "error": <Error Object>  
}

### 3. **Диаграма на действията**



### 4. **Електронна идентификация и журнал на действията от страна на АИС**

Всеки АИС (вкл. първичните регистри) трябва да поддържат вход за длъжностни лица в администрациите с националната схема за електронна идентификация. Преди тя да бъде достъпна, за временен преходен период, това може да става и с квалифициран електронен подпис (напр. чрез системата за електронна автентикация).

Всеки път, когато длъжностно лице се идентифицира електронно пред системата, центърът за електронна идентификация (Identity Provider) предоставя token за достъп до данни и token, идентифициращ уникално текущата сесия. Вторият token трябва да бъде включен в ръчни заявки за данни, за може впоследствие еднозначно (чрез корелиране на този token между централния координатор и валидиращия орган) да се определи дали заявителят наистина е този, който се твърди.

Отделно от това, всеки АИС трябва да пази журнал на действията. В него трябва да се записва длъжностното лице, извършеното от него действие, заедно с датата и часа (използвайки системата определяне на единно време).

### 5. **Допълнителни изисквания към първичните регистри**

Първичните регистри трябва да:

* обявят своето съществуване на координатора, като при стартирането си извикат POST /registers с тяло със следната структура:

{  
 "administrationId" : "b25b24c2-17f2-4ac3-885e-261506a8c693",  
 "serviceId" : "b25b24c2-17f2-4ac3-885e-261506a8c693:BRRA"  
 "serviceName" : "Търговски регистър",  
 "serviceRootUrl": "http://api.brra.bg"  
}

HTTP header-ите Signature и Signature-Algorithm трябва да се използват и при тази заявка.

* administrationId е идентификатора от ИИСДА на администрацията, предоставяща регистъра.
* serviceId е идентификатора на АИС (регистъра, в случая), който се образува от идентификатора на администрацията, двоеточие, и уникален ключ за система в рамките на администрацията.
* Освен продукционната среда, всеки първичен регистър трябва да поддържа и поне една тестова/staging среда. Данните в тестовата среда не трябва да бъдат реални, а трябва да бъдат документирани в /api/docs/test
* Поддържат услуга за проверка на законно представителство (напр. Търговският регистър поддържа проверка дали дадено физическо лице е законен представител на юридическо лице, база данни население поддържа проверка дали дадено лице е законен представител на малолетно или непълнолетно лице)

### 6. **Описание на централния координатор**

Централният координатор е интеграционен компонент. През него минават само метаданни, а не цялата комуникация. В диаграмата по-горе ясно се вижда неговото участие. Основните му функции са:

* да провери автентичността на заявката и заявителя
* да провери дали заявителят има достъп (базиран на право основание) до данните, които е заявил.
* да запише информация за заявката в журнал, използвайки сигурно време, предоставено от Time Stamping Authority (т.нар. СОЕВ), така че да гарантира противопоставимост в съда.
* да предостави механизъм на системата на първичния регистър да провери дали всяка, получена от него заявка, е оторизирана.

### 7. **Описание на централния регистър**

Централният регистър е логически компонент, реализиран от една страна чрез ИИСДА (Интегрирана Информационна Система на Държавна Администрация), от друга страна с отделна база данни към централния координатор.

Данните в ИИСДА се попълват от администрациите и утрърждават от администрацията на министерския съвет, в съответствие с Наредбата за административния регистър.

За всяка услуга (без значение дали към момента на въвеждане тя е електронизирана) се води списък с данни, до които администрацията, която извършва услугата, има нормативно-обоснован достъп. При попълване на тези данни от администрациите, те се вземат от публикуваните типове данни от всеки първичен регистър, чрез интерфейса /api/data, и се представят в удобен за търсене вид (напр. първичен регистър + описание на данната)

Ако даден тип данни не е публикуван от никой регистър към момента на въвеждането на данни в административния регистър, то този избор може да бъде направен впоследствие.

Централният регистър поддържа и PKI-инфраструктурата.

Освен функционалностите на ИИСДА, с която основно комуникира координаторът, АИС-ите имат достъп до няколко списъка:

Намиране на регистър, който предоставя дадена данна:

GET https://<central-registry-url>/api/findRegister?dataKey={dataKey}

Намиране на всички регистри на дадена администрация:

GET https://<central-registry-url>/api/findRegister?destinationId={destinationId}