Чек-лист. Поток «Получение доступа к информации по счету»

№	Взаимодействие	Действие	Проверяемая функция	Результат
Получение токена доступа в рамках типа доступа client_credentials с использованием client_assertion				
1	Пользователь → СПИУ	Установка соединения TLS 1.2	Пользователь, используя Web браузер устанавливает соединение с приложением Стороннего поставщика и авторизуется в нем.	
2	Пользователь → СПИУ	Запрос на получение информации о счетах/транзакциях	Пользователь в приложении Стороннего поставщика выбирает Банк, обслуживающий счет по которому он хочет получить информацию.	
3	СПИУ	Создание и подписание client_assertion	Создается client_assertion, где • "sub" = client_id • "iss" = client_id • "aud" = "token_endpoint" Сервера авторизации Для подписания client_assertion использовать сертификат [Sign] СПИУ/СППУ Песочница (сертификат подписи для авторизации)	
4	СПИУ	Создание запроса к конечной точке токена (тип доступа client_credential)	Запрос должен включать параметры: • grant_type: client_credentials • scope: openid accounts • client_assertion • client_assertion_type: urn:ietf:params:oauth:client-assertion-type:jwt-bearer	
5	СПИУ↔ Сервер авторизации	Установка соединения TLS 1.2 МА	Использовать элементы проверок для установки соединения TLS 1.2 МА	
6	СПИУ → Сервер авторизации	Запрос к конечной точке токена POST /token (grant_type, scope, client_assertion, client_assertion_type)	Запрос к конечной точке токена • request_method POST • request_headers • request_body	

			• request_mutual_tls ("cert": "key":"ca":)	
7	Сервер авторизации	Загрузка и проверка TLS сертификата клиента	Определение параметров сертификата: • CN • x5t#S256	
8	Сервер авторизации	Проверка параметров представленного запроса	• grant_type = "client_credentials" • scope = "openid accounts" • client_assertion = "представлено" • client_assertion_type = "urn:ietf:params:oauth:client-assertion-type:jwt-bearer"	
9	Сервер авторизации	Проверка структуры client_assertion	• "sub" = client_id • "iss" = client_id • "aud" = "token_endpoint" Сервера авторизации • "exp" - имеет актуальное значение (не просрочен) • "jti" - имеет уникальное значение, ранее не предъявлялось	
10	Сервер авторизации	Получение сертификата клиента с JWKS по kid и client_id	Выбран сертификат [Sign] СПИУ/СППУ Песочница (сертификат подписи для авторизации): "certificateOID"= 1.2.643.6.57.42.21.3.2.1 "x5c" = тело сертификата.	
11	Сервер авторизации	Актуальность подписи client_assertion	• "client_assertion" • "x5c"	
12	Сервер авторизации	Представленный TLS сертификат принадлежит Клиенту с client_id из запроса	Связка CN TLS сертификата с client_id	
13	Сервер авторизации	Генерация токена доступа (access-token)	Генерации токена доступа в привязке к TLS сертификату клиента. "client_id": "	

			"cnf": { "x5t#S256":	
14	Сервер авторизации → СПИУ	HTTP 200 (OK) access-token (scope:accounts)	Получение ответа от конечной точки токена • response_status_code 200 OK • response_status_text OK • response_headers • response_body	
15	СПИУ	Проверка ответа от конечной точки токена	• response_status_code = 200 OK • присутствуют параметры (access_token, scope, token_type, expires_in) • scope="accounts" • access_token имеет требуемую энтропию • срок актуальности (expires_in) не истек • token_type = Bearer	
16	СПИУ	Извлечение токена доступа из ответа от конечной точки токена	Токена доступа и его тип: • access_token • token_type	
Запрос к конечной точке ресурса для создания намерения (ConsentId)				
17	СПИУ	Создание запроса ресурса	Создание параметров запроса к конечной точке ресурсов. request_headers: • x-jws-signature (для подписания использовать сертификат [Enc] СПИУ/СППУ Песочница (сертификат для подписи полезной нагрузки) 1.2.643.6.57.42.21.3.3.1 • Authorization: Bearer - полученный токен доступа request_body: • Permissions	

18	СПИУ → Сервер ресурсов	Установка соединения TLS 1.2 МА		
19	СПИУ → Сервер ресурсов	Запрос к конечной точке ресурса для типа доступа client_credentials	• request_method POST • request_headers • request_body • request_mutual_tls ("cert": "key":"ca":)	
20	Сервер ресурсов	Проверка наличия ошибки в HTTP response ("response_status_code")		
21	Сервер ресурсов	Проверка актуальности токена доступа (access-token)	• "nbf": -актуальная дата • "exp»: -актуальная дата • "iss": -URI сервера авторизации • "aud": -URI сервера ресурсов • "jti": - соответствует значению при генерации	
22	Сервер ресурсов	Проверка связи scope: accounts с представленным токеном	"scope": "accounts"	

23	Сервер ресурсов	Токен доступа связан с представленным TLS сертификатом клиента	Проверка значения параметра JWT токена доступа
			"cnf": {
			"x5t#S256"=хэш-код X.509 TLS сертификата клиента
			}	

24	Сервер ресурсов	Проверка связки токена с идентификатором клиента client_id	Проверка значения параметра JWT токена доступа "client_id":	
25	Сервер ресурсов	Проверка полезной нагрузки запроса ресурса	Проверка полезной нагрузки на соответствие прикладным стандартам	
26	Сервер ресурсов	Создание нового ресурса с идентификатором consentId		

27	Сервер ресурсов	Связывание идентификатора ресурса consentId с client_id		
28	Сервер ресурсов	Создание x-fapi-interaction-id		
29	Сервер ресурсов	Генерация и подписание полезной нагрузки. Подпись передается в ответе в качестве HTTP заголовка x-jws-signature.	Генерация и подписание полезной нагрузки. Подпись передается в ответе в качестве HTTP заголовка x-jws-signature. Для подписания выбрать сертификат [Enc] ППУ ресурсный сервер Песочница "certificateOID"=1.2.643.6.57.42.13.2.3.1	
30	Сервер ресурсов → СПИУ	Отправка ответа от конечной точки ресурса	• response_status_code 201 CREATED • response_headers • response_body	
31	СПИУ	Парсинг ответа	Объекта ответа account_requests_endpoint_response • headers • body	
32	СПИУ	Проверка наличия в заголовке значения x-fapi-interaction-id	Наличие x-fapi-interaction-id в заголовке ответа	
33	СПИУ	Получение сертификата сервера с JWKS по "jwks_uri" для проверки подписи полезной нагрузки	Выбрать сертификат [Enc] ППУ ресурсный сервер Песочница "certificateOID"=1.2.643.6.57.42.13.2.3.1 "x5c" = тело сертификата.	
34	СПИУ	Проверка подписи полезной нагрузки	x-jws-signature - корректна	
35	СПИУ	Сохранение идентификатора ресурса	consentId	
Запрос авторизации				
36	СПИУ	Создание запроса к конечной точке авторизации	• client_id • redirect_uri (/callback)	

			• scope: openid accounts	
37	СПИУ	Добавление в запрос параметра (claims) userinfo, содержащего openbanking_intent_id	"openbanking_intent_id" = consentId	
38	СПИУ	Добавление в запрос наименования класса контекста запрашиваемой аутентификации пользователя acr в параметре (claims) id_token в привязке к openbanking_intent_id для определения требуемого способа аутентификации.	"openbanking_intent_id" = consentId "urn:rubanking:sca" - требуется строгая аутентификация "urn:rubanking:ca" - не требуется строгая аутентификация	
39	СПИУ	Генерация заявленного свойства state	requested_state_length state	
40	СПИУ	Генерация заявленного свойства nonce	requested_nonce_length nonce	
41	СПИУ	Создание объекта запроса, добавление в запрос других обязательных параметров:	• response_type: code id_token • exp • aud • iss • client_id • redirect_uri	
42	СПИУ	Подписание объекта запроса	Для подписания объекта запроса использовать сертификат [Sign] СПИУ/СППУ Песочница (сертификат подписи для авторизации)	
43	СПИУ → Пользователь	Создание редиректа на конечную точку авторизации (отправка объекта запроса)	HTTP 302 (Found) redirect_to_authorization_endpoint Подписанный объект запроса передается в виде JWS параметра request в запросе к конечной точке авторизации	

44	Пользователь → Сервер авторизации	Установка соединения TLS 1.2		
45	Пользователь → Сервер авторизации	HTTP GET - запрос на конечную точку авторизации от имени пользователя	redirect_to_authorization_endpoint	
46	Сервер авторизации	Проверка соответствия параметров объекта запроса	Проверка соответствия параметров: • scope = accounts openid • response_type: code id_token • redirect-uri • значение client_Id передается как часть URI запроса	
47	Сервер авторизации	Получение сертификата клиента с JWKS по client_id и kid	Выбрать сертификат [Sign] СПИУ/СППУ Песочница (сертификат подписи для авторизации): "certificateOID"= 1.2.643.6.57.42.21.3.2.1 "x5c" = тело сертификата	
48	Сервер авторизации	Проверка подписи объекта запроса	request	
49	Сервер авторизации	Проверка наличия всех авторизационных claims в объекте запроса	• userinfo: openbanking_intent_id = consentId • id_token: openbanking_intent_id = consentId • id_token: acr	
50	Сервер авторизации	Проверка того, что consentId из запроса принадлежит client_id, инициировавшему запрос		
51	Пользователь → Сервер авторизации	Аутентификация Пользователя в соответствии с требуемым классом контекста аутентификации, указанным в объекте запроса (acr)		

52	Пользователь → Сервер авторизации	Выбор счета(ов), определение кластеров данных доступа (данные для Согласия/Разрешения)		
53	Сервер авторизации	Генерация кода авторизации (code)	code - имеет короткое время жизни (например 5 минут) и достаточную энтропию	
54	Сервер авторизации	Генерация c_hash и включение его в id_token	c_hash = хэш-коду значения кода авторизации (code)	
55	Сервер авторизации	Включение в id_token параметра nonce из запроса аутентификации	nonce = значению параметра <nonce> в запросе аутентификации	
56	Сервер авторизации	Генерация s_hash и включение его в id_token	s_hash = хэш-коду значения параметра <state> в запросе аутентификации	
57	Сервер авторизации	Генерация токена идентификации (id_token)	id_tocken содержит следующие параметры: "nbf":, "exp": , "iss": , "aud": , "iat": , "nonce": , "c_hash": , "s_hash": , "sub": , - идентификатор пользователя на сервере авторизации "auth_time": , - время аутентификации пользователя "name": - имя пользователя , "openbanking_intent_id": = consentid, "amr": ["password"] - как был авторизован пользователь	
58	Сервер авторизации	Подписание id_tocken (JWS)	Для подписания используется сертификат	

			[Sign] ППУ сервер авторизации Песочница 1.2.643.6.57.42.13.1.2.1
			В заголовок JWS id_token добавляется идентификатор ключа kid:
			{
			"alg": "RS256",
			"kid":
			"typ": "JWT"
			}	

59	Сервер авторизации	Связывание счетов, доступ к которым авторизован Пользователем в результате аутентификационной сессии, с идентификаторами consentId/RetrievalGrantId	Способы передачи на Сервер ресурсов и хранения consentId/RetrievalGrantId ППУ определяет самостоятельно	
60	Сервер авторизации → Сервер ресурсов	Создание суб-ресурса RetrievalGrantId, привязанного к consentId		
61	Сервер авторизации → Сервер ресурсов	Обновление статуса ресурса consentId на "Authorised"	Способы передачи на Сервер ресурсов и хранения consentId/RetrievalGrantId ППУ определяет самостоятельно	
62	Сервер ресурсов → Сервер авторизации	Статус обновлен	Способы передачи на Сервер ресурсов и хранения consentId/RetrievalGrantId ППУопределяет самостоятельно	
63	Сервер авторизации → Пользователь	HTTP 302 (Found); Location: redirect-uri (authorization-code, id_token, state)		
64	Пользователь → СПИУ	HTTP GET redirect-uri (authorization-code, id_token, state)	Значения параметров передается в виде fragment	
Проверка ответа авторизации				

65	СПИУ	Проверка URL query на отсутствие кода авторизации	Код авторизации не представлен в возвращенном URL query	
66	СПИУ	Проверка URL query на отсутствие 'error'	'error' не представлен в возвращенном URL query	
67	СПИУ	Проверка на наличие ошибок с конечной точки авторизации		
68	СПИУ	Проверка состава параметров ответа авторизации	Представлены только: code, id_token, state, session_state, scope	
69	СПИУ	Проверка корректности параметра state	Значение state в authorization_response соответствует ранее сгенерированному	
70	СПИУ	Проверка наличия кода авторизации	Наличие authorization code	
71	СПИУ	Проверка длины кода авторизации	Длина authorization code: actual >= required	
72	СПИУ	Проверка энтропии кода авторизации	Энтропия authorization code: actual >= expected	
73	СПИУ	Извлечение id_token из ответа авторизации и проверка его в соответствии с ФАПИ.СЕК 5.4.2.17. Проверка ID токена клиентом.	Проверка ID токена клиентом должна выполняться следующим образом: 1. Если ID токен зашифрован, следует расшифровать его, используя ключи и алгоритмы, которые сервер авторизации должен был использовать для шифрования ID токена как JWE; 2. Идентификатор эмитента (сервера авторизации), полученный при регистрации клиента, должен точно соответствовать значению параметра <iss> ID токена; 3. Клиент должен убедиться, что значение параметра <aud> содержит значение <client_id>, полу-	

			ченное клиентом от сервера авторизации при регистрации. ID токен должен быть отклонен, если <aud> не содержит значение <client_id> клиента; 4. Если ID токен содержит несколько компонент в составе параметра <aud>, клиент должен проверить наличие параметра <azp> и при его наличии проверить, что значение равно <client_id>; 5. Клиент должен проверить подпись структуры JWS ID токена, применяя алгоритм, указанный в параметре <alg>, используя ключ, предоставленный сервером авторизации. Если ID токен получен посредством прямого обмена данными между клиентом и конечной точкой токена, проверка сообщений протокола TLS от сервера может использоваться для проверки эмитента вместо проверки подписи токена; 6. Значение параметра <alg> должно быть значением по умолчанию или алгоритмом, зарегистрированным клиентом в значении параметра <id_token_signed_response_alg> во время регистрации; 7. Если в значении параметра <alg> указан алгоритм MAC, то в качестве ключа должны использоваться октеты UTF-8 представления <client_secret>, соответствующего <client_id> клиента; 8. Текущее время проверки должно быть раньше времени, указанного в значении параметра <exp>; 9. Для отзыва токенов, выпущенных слишком давно, может использоваться параметр <iat>, который ограничивает время, необходимое для хранения значений параметра <nonce>. Приемлемый диапазон определяется при разработке сервера авторизации;	

			10. Если в запросе аутентификации, отосланном клиентом в адрес сервера авторизации, указано значение параметра <nonce>, в структуре ID токена также должно присутствовать значение параметра <nonce>. Следует убедиться, что эти два значения совпадают; 11. Если запрашивалось значение параметра <auth_time> либо посредством специального запроса этого параметра, либо указанием параметра <max_age> в запросе аутентификации, клиенту следует проверить значение параметра <auth_time> на допустимость диапазону <max_age> и сделать запрос повторной аутентификации, если он определяет, что с момента последней аутентификации конечного пользователя прошло слишком много времени.	
74	СПИУ	Проверка значения s_hash	s_hash соответствует state из запроса	
75	СПИУ	Проверка значения c_hash	значение c_hash соответствует code	
76	СПИУ	Проверка значения openbanking_intent_id на соответствие идентификатору авторизируемого ресурса	openbanking_intent_id соответствует идентификатору авторизуемого ресурса	
77	СПИУ	Получение сертификата сервера с JWKS по "jwks_uri" для проверки подписи полезной нагрузки	Выбрать сертификат по представленному kid из заголовка JWS "x5c" = тело сертификата.	
78	СПИУ	Проверка подписи токена идентификации	Подпись id_token актуальна и поставлена с использованием представленного значения kid	
Создание запроса к конечной точке токена для обмена кода авторизации (Authorization Code Grant)				
79	СПИУ	Создание и подписание client_assertion	Создается client_assertion, где	

			• "sub" = client_id • "iss" = client_id • "aud" = "token_endpoint" Сервера авторизации Для подписания client_assertion использовать сертификат [Sign] СПИУ/СППУ Песочница (сертификат подписи для авторизации)	
80	СПИУ	Создание запроса к конечной точке токена	Создание параметров запроса к конечной точке токена • request_headers • request_body • client_assertion	
81	СПИУ	Включение заявленных свойств клиента в запрос	Присутствуют элементы: • grant_type • code • redirect_uri • client_assertion • client_assertion_type	
82	СПИУ → Сервер авторизации	Установка соединения TLS 1.2 МА		
83	СПИУ → Сервер авторизации	Вызов к конечной точке токена	HTTP request: • request_uri • request_method • request_headers • request_body • request_mutual_tls	
84	Сервер авторизации	Загрузка и проверка TLS сертификата клиента	Определение параметров сертификата: • CN • x5t#S256	

85	Сервер авторизации	Проверка параметров представленного запроса	• grant_type = "authorization_code" • client_assertion = "представлено" • client_assertion_type = "urn:ietf:params:oauth:client-assertion-type:jwt-bearer" • code = представлен	
86	Сервер авторизации	Проверка структуры client_assertion Аутентификация клиента механизмом private_key_jwt	• "sub" = client_id • "iss" = client_id • "aud" = "token_endpoint" Сервера авторизации • "exp" - имеет актуальное значение (не просрочен) • "jti" - имеет уникальное значение, ранее не предъявлялось	
87	Сервер авторизации	Получение сертификата клиента с JWKS по client_id	Выбрать сертификат [Sign] СПИУ/СППУ Песочница (сертификат подписи для авторизации): "certificateOID"= 1.2.643.6.57.42.21.3.2.1 "x5c" = тело сертификата.	
88	Сервер авторизации	Актуальность подписи client_assertion	• "client_assertion" • "x5c"	
89	Сервер авторизации	Представленный TLS сертификат принадлежит Клиенту с client_id из запроса	Связка CN TLS сертификата с client_id	
90	Сервер авторизации	Проверка code	код авторизации был выдан аутентифицированному клиенту (client_id); код авторизации актуален; код авторизации ранее не использовался;	
91	Сервер авторизации	Генерация токена доступа (access-token)	Генерации токена доступа в привязке к TLS сертификату клиента.	

			"client_id": "cnf": {
			"x5t#S256":
			}	

92	Сервер авторизации	Генерация id_tocken	• параметр <nonce> обязательный; • обязательный параметр <at_hash>: хэш-значение токена доступа; вычисляется сервером авторизации и проверяется клиентом как Base64url кодирование левой половины хэш-кода октетов ASCII представления значения <access_token>	
93	Сервер авторизации	Генерация refresh_token	refresh_token привязан к client_id и consentid	
94	Сервер авторизации → СПИУ	Ответ от конечной точке токена	• response_status_code 200 OK • response_status_text OK • response_headers • response_body	
95	СПИУ	Извлечение тела из HTTP ответа	Получено тело token_endpoint_response	
96	СПИУ	Парсинг token_endpoint_response	Присутствуют элементы: • access_token • refresh_token • id_token • token_type = Bearer • expires_in - актуален	
97	СПИУ	Проверка значения токена обновления	Длина refresh_token: actual >= required	
98	СПИУ	Проверка энтропии токена обновления	Энтропия refresh_token: actual >= expected	
99	СПИУ	Извлечение id_tocken из ответа от конечной точки токена	id_token содержит требуемые claim	

100	СПИУ	Проверка структуры id_token	ID токен включает следующие параметры: • <iss>: (обязательный) идентификатор эмитента (сервера авторизации), источника ответа; регистрозависимый URL-адрес, использующий схему https; содержит схему, хост и опционально компоненты номера порта и пути, но не компоненты запроса или фрагмента; • <sub>: (обязательный) уникальный идентификатор субъекта, выданный сервером авторизации конечному пользователю; регистрозависимая строка длиной не более 255 символов ASCII; • <aud>: (обязательный) аудитория, для которой предназначен данный ID токен; должна содержать идентификатор <client_id>; в общем случае значение <aud> представляет собой массив чувствительных к регистру строк; если есть только один элемент, значением <aud> может быть единственная строка; • <exp>: (обязательный) время завершения срока действия, при наступлении и после наступления которого ID токен не должен приниматься к обработке; значением является число в формате JSON, представляющее количество секунд от 1970–01-01T0:0:0Z UTC до соответствующей даты/времени; • <iat>: (обязательный) время, когда был выпущен токен; значением является число в формате JSON, представляющее количество секунд от 1970–01-01T0:0:0Z UTC до соответствующей даты/времени; • <auth_time>: (обязательный, если в запросе аутентификации присутствует значение параметра <max_age>) время, когда выполнена	

			аутентификация конечного пользователя; значением является число в формате JSON, представляющее количество секунд от 1970–01-01T0:0:0Z UTC до соответствующей даты/времени; • <nonce>: строковое значение; равно значению параметра <nonce>, в запросе аутентификации; серверу авторизации следует включать этот параметр, если значение параметра <nonce> присутствует в соответствующем запросе аутентификации; • <azp>: (опциональный) идентификатор клиента стороны, для которого был выпущен ID токен. • добавлен обязательный параметр <at_hash>: хэш-значение токена доступа; вычисляется сервером авторизации и проверяется клиентом как Base64url кодирование левой половины хэш-кода октетов ASCII представления значения <access_token>;	
101	СПИУ	Проверка заявленного свойства nonce	Значение nonce в id_token соответствует ранее сгенерированному	
102	СПИУ	Проверка заявленного свойства nonce	Значение acr в id_token соответствует или одно из значений из запроса (actual = requested или actual in requested)	
103	СПИУ	Проверка значения openbanking_intent_id на соответствие идентификатору авторизуемого ресурса	openbanking_intent_id = consentId	
104	СПИУ	Проверка подписи токена идентификации	Подпись id_token актуальна	
105	СПИУ	Проверка значения ключа подписи токена идентификации	Подпись id_token поставлена с использованием представленного значения kid	

106	СПИУ	Проверка допустимости алгоритма подписи токена идентификации	Подпись id_token поставлена с использованием допустимого алгоритма alg = permitted_algorithm	
107	СПИУ	Проверка значения at_hash	значение at_hash соответствует access-token	
Проверка доступа к серверу ресурсов				
108	СПИУ	Генерация заголовка запроса	Сгенерированный заголовок HTTP запроса к ресурсу	
109	СПИУ	Генерация fapi_interaction_id	fapi_interaction_id	
110	СПИУ → Сервер ресурсов	Установка соединения TLS 1.2 MA		
111	СПИУ → Сервер ресурсов	Запрос к конечной точки ресурса с Bearer токен и заполненным заголовком	• request_method: GET • request_headers • request_body (empty) • request_mutual_tls ("cert": "key":"ca":)	
112	Сервер ресурсов	Представленный TLS сертификат принадлежит Клиенту с client_id из запроса	Связка CN TLS сертификата с client_id	

113	Сервер ресурсов	Проверка токена доступа (access-token)	• Токен представлен в заголовке запроса: Authorization = “Bearer” ‖ <token>
			• Токен доступа привязан к TLS сертификату клиента.
			"client_id": "4ba3b98a4c6b4731a08bcb91229d1250",
			"cnf": {
			"x5t#S256": "_zdkTcZq4tjnsu8GJ73evA9API-oYzlNOpMa4wqZkh9k"
			}
			• Токен доступа принадлежит client_id
			• Токен доступа связан с consentid	

114	Сервер ресурсов → СПИУ	Получение ответа от конечной точки ресурсов	• response_status_code 200 OK • response_status_text Created • response_headers • response_body	
115	СПИУ	Извлечение ответа		
116	СПИУ	Проверка даты и времени из ответа	Значение date актуально (проверка на разницу с текущим временем) openid-financial-api-part-1-ID2.html#rfc.section.6.2.1-11	
117	СПИУ	Проверка наличия в заголовке значения x-fapi-interaction-id	Наличие x-fapi-interaction-id	
118	СПИУ	Проверка соответствия значения x-fapi-interaction-id значению запроса	x-fapi-interaction-id соответствует значению из HTTP request	
119	СПИУ	Кодовая страница ответа должна быть UTF-8	HTTP response charset = UTF-8 openid-financial-api-part-1-ID2.html#rfc.section.6.2.1-10,9	
120	СПИУ	Тип структуры данных ответа должен быть JSON	HTTP response type= JSON	
121	СПИУ	Проверка подписи полезной нагрузки	x-jws-signature - корректна.	
122		Парсинг ответа и получение информации о счете	Объекта ответа account_requests_endpoint_response • headers • body openid-financial-api-part-1-ID2.html#rfc.section.6.2.1	